

WebSphere. Lombardi Edition Version 7.1

Authoring Environment User Guide



Table of Contents

Planning Lombardi projects	1
What is business process modeling?	1
How are processes developed in Lombardi and who is involved?	1
Lombardi product components	4
Lombardi architecture	4
Lombardi key terms and concepts	6
Understanding process development in Lombardi	7
Understanding the Process Center	8
Re-using items in Lombardi	9
Versioning Lombardi items	10
Planning for process deployment and installation	10
Starting Lombardi Authoring Environment	12
Logging in	12
Navigating initial views	12
Accessing and using Lombardi Authoring Environment interfaces	14
Creating your first Lombardi project	15
Authoring Environment tips and shortcuts	15
Lombardi tasks	19
Managing the Process Center repository	22
Overview	22
Where to perform tasks	23
To learn more	23
Managing process applications, workspaces, and snapshots	24
Overview	24
Creating and maintaining high-level library items	24
Managing process applications	25
Creating new process applications in the Process Center Console	25
Cloning process applications in the Process Center Console	26
Copying or moving library items from one process application to another in the Designer view	26
Archiving process applications in the Process Center Console	26
Importing and exporting process applications from the Process Center Console	27
Managing and using toolkits	29
About Lombardi System Data toolkit	29
Creating toolkits in the Process Center Console	29
Cloning toolkits in the Process Center Console	30
Moving or copying library items to a toolkit in the Designer view	31
Creating a toolkit dependency in the Designer view	31
Updating a toolkit dependency in the Designer view	32
Deleting a toolkit dependency in the Designer view	33
Archiving toolkits in the Process Center Console	34
Importing and exporting toolkits from the Process Center Console	34
Managing workspaces	36
Enabling workspaces in the Process Center Console	36
Creating new workspaces in the Process Center Console	37
Editing workspaces in the Process Center Console	38
Setting the default workspace in the Process Center Console	39
Archiving workspaces in the Process Center Console	40
Managing snapshots	40
Creating new snapshots in the Process Center Console	41
Creating new snapshots in the Designer view	42

Comparing snapshots in the Designer view	42
Creating snapshots from the revision history in the Designer view	43
Activating snapshots in the Process Center Console	44
Archiving snapshots in the Process Center Console	45
Managing access to the Process Center repository	46
Granting administrative access to the Process Center repository	47
Adding users and groups	47
Managing access to process applications and toolkits	48
Removing users and groups	50
Managing Lombardi servers	50
Monitoring installed snapshots on each Process Server from the Process Center Console	51
Configuring Lombardi Process Servers from the Process Center Console	52
Adding offline servers to the Process Center Console	53
Removing offline servers from the Process Center Console	53
Configuring installed snapshots	54
Managing library items in the Designer view	56
Navigating the library	56
Creating favorites	57
Tagging library items	59
Organizing library items in smart folders	60
Copying or moving library items	62
Reverting to a previous version of a library item	63
Copying a library item from a snapshot	64
Understanding concurrent editing	65
Subscribing to Blueprint processes	67
Subscribing to Blueprint processes in the Designer view	67
Opening subscribed processes in Blueprint	68
Updating Blueprint processes in the Designer view	68
Removing Blueprint subscriptions from the Designer view	69
Managing external files	70
Adding managed files	70
Adding managed files using drag and drop	71
Updating managed files	71
Replacing a managed file	71
Using managed files	72
Deleting managed files	72
Importing files from previous versions of Lombardi	72
Modeling processes	75
Building processes in Lombardi	75
Using the Designer in Lombardi Authoring Environment	75
Understanding process components	76
Basic modeling tasks	78
Creating a BPD	78
Adding lanes to a BPD	80
Assigning participant groups to lanes	81
Adding activities and other process components to a BPD	82
Establishing process flow with sequence lines	84
Using gateways	85
Implementing activities	88
Adding process variables to a BPD	89
Adding events to a BPD	93
Setting environment variables	94
Validating processes	95

Configuring BPDs	97
Exposing BPDs	98
Setting the work schedule for a BPD	99
Setting the name and due date for BPD instances	100
Advanced modeling tasks	100
Building services	101
Building Coaches	122
Using nested processes	157
Using embedded JavaScript	160
Using external activities	161
Integrating with other systems	164
Modeling events	191
Managing and mapping variables	201
Handling exceptions	217
Creating loops	220
Helpful reference information	222
Creating a participant group	222
Creating a user attribute definition	224
Routing activities	225
Example gateways	230
Lombardi naming conventions	234
Running and installing processes	235
Overview	235
To learn more	236
Running and debugging processes with the Inspector	236
Managing process instances	237
Stepping through a process	239
Debugging a process	244
Resolving errors	247
Inspector reference	249
Releasing and installing processes	252
Developing a release and installment strategy	253
Building installation services	253
Installing process applications: online Process Servers	254
Installing process applications: offline Process Servers	255
Migrating instances	257
Completing post-installation tasks	257
Troubleshooting installations	258
Customizing process application installations on offline Process Servers	260
Configuring KPIs and SLAs	264
Using KPIs	264
Using SLAs	264
Creating custom KPIs	265
Associating KPIs with activities	266
Creating SLAs	267
Creating and configuring reports	269
Reporting options	269
How reporting works in Lombardi	269
How Lombardi transfers tracked data	271
Determining which reporting option meets your needs	272
Using out of the box scoreboards	273
My Performance	273
My Team Performance	274
Process Performance	275

SLA Overview	276
Tracking Lombardi performance data	277
Tracking options	277
About autotracking	278
About tracking groups	278
About timing intervals	279
Sending tracking definitions	279
Supported data types	279
Naming tracking groups	280
Tracking data across processes and process applications	280
Working with versioned data	280
Creating a quick custom report	280
Configuring autotracking	281
Creating a quick report using the ad-hoc wizard	282
Creating a basic custom report	285
Creating a timing interval	285
Creating a basic report that uses the timing interval	288
Creating a more advanced custom report	291
Creating a tracking group	291
Steps to create a more advanced custom report	293
Creating an Integration service that contains a query	294
Creating the report	295
Creating the scoreboard	300
Adding a filter	301
Creating a third-party report	303
Using autotracked data	304
Using a third-party tool	304
Performance Data Warehouse database architecture	307
Tracking Group views	308
SNAPSHOTS view	309
TASKS view	309
TRACKINGGROUPS view	310
TIMINGINTERVALS view	310
TIMINGINTERVALVALUE view	310
TRACKEDFIELDS view	311
TRACKEDFIELDUSE view	311
TRACKINGPOINTS view	312
TRACKINGPOINTVALUE view	312
PROCESSFLOWS view	313
SLA STATUS view	314
SLATHRESHOLDTRAVERSALS view	315
Simulating and optimizing processes	316
Configuration requirements for simulation	317
Setting up simulation profiles	317
Setting simulation properties for participant groups	320
Creating simulation analysis scenarios	321
Configuration requirements for optimization	322
Optional configuration for optimization	323
Tracking performance data for the Optimizer	324
Creating historical analysis scenarios	325
Analyzing data from Performance Data Warehouses in runtime environments	327
Generating historical data	327
Running simulations, historical analyses, and comparisons	331
Before you begin	331

Running scenarios	332
Reviewing results	334
Understanding heat maps	335
Understanding live reports	335
Reviewing recommendations	337
Using the Smart Start view	339
Sample simulations	341
Running a quick simulation	341
Taking advantage of simulation profiles and scenarios	347
Sample historical analyses and comparisons	352
Running an historical analysis	352
Using the guided optimization wizard	355
Running a Simulation vs. Historical comparison	359
Notices and Trademarks	362
Notices	362
Trademarks and service marks	364

Planning Lombardi projects

Efficiency and cost-effectiveness are the ultimate goals of all Business Process Management (BPM) initiatives. IBM WebSphere Lombardi Edition provides a complete platform for designing, developing, and delivering applications to streamline your business processes. With Lombardi, you can build everything you need in one place, including process models, forms, rules, and services.

What is business process modeling?

Business process models are diagrams that depict the steps in a process. The Business Process Modeling Notation (BPMN) is a graphical notation that standardizes the depiction of those steps. Lombardi supports the BPMN standard, which enables you to leverage process diagrams created in other BPMN-compliant applications. (For more information about BPMN, see <http://www.bpmn.org/>.) The BPMN standard maps directly to business process execution languages, and because Lombardi is compliant with these languages, it provides powerful interoperability with other process modeling tools.

Types of business processes that normally require automation and ongoing maintenance and management include:

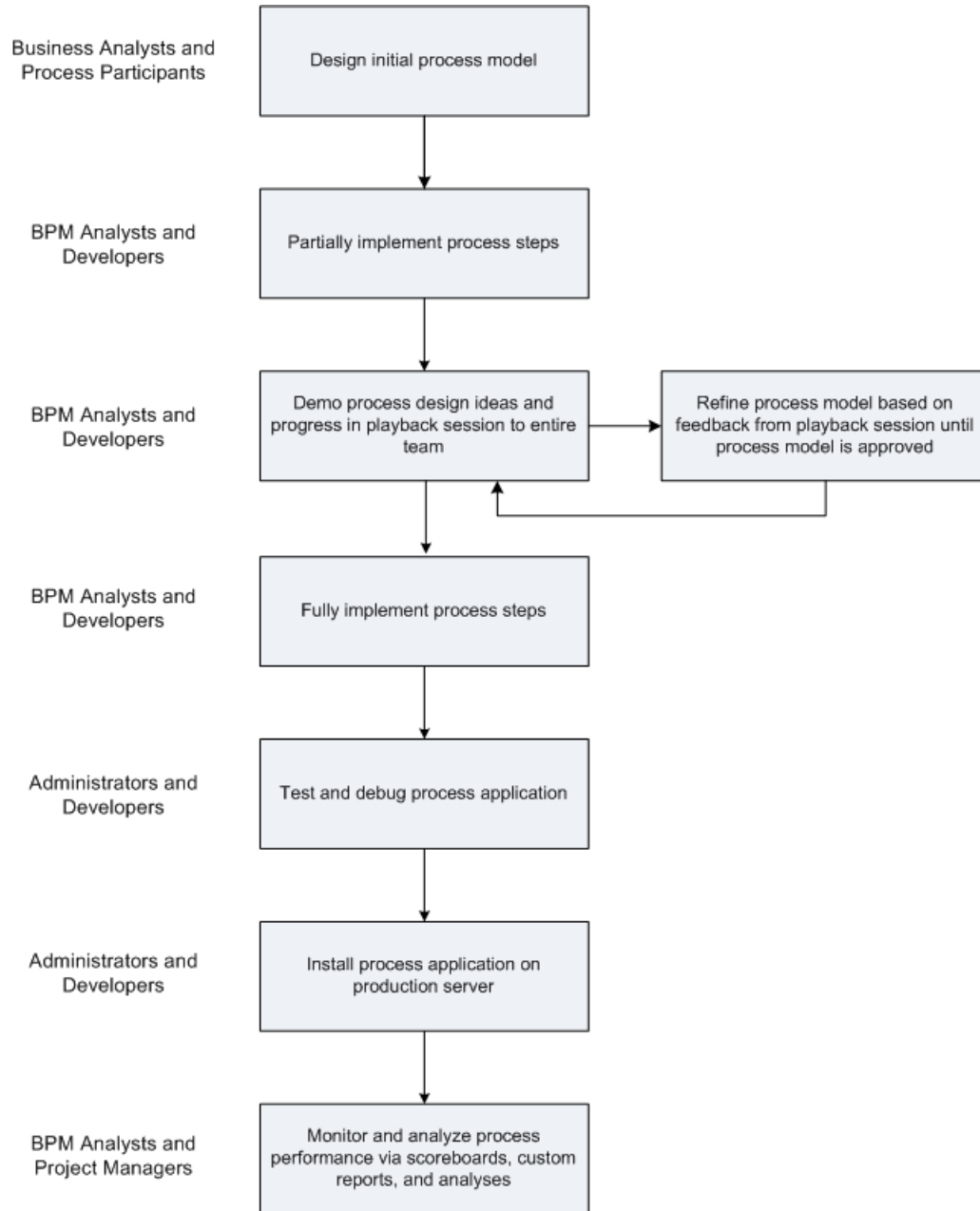
- System to system
- System to human
- Human to human

Lombardi enables you to design and automate the preceding types of processes because it fully supports both inbound and outbound integration with external systems and also enables you to easily develop interfaces to collect data from and otherwise interact with end users.

Lombardi provides the capability to diagram a process step by step and then implement each step in the process model even if you need to pass data from one external system to another and then on to end users who are vital to completion of the process. Lombardi gives you the ability to develop a fully functional process application, providing easy-to-use tools for each role involved.

How are processes developed in Lombardi and who is involved?

The following figure illustrates a typical process development effort in Lombardi:



Several different types of individuals are normally involved in the development of a process in Lombardi as outlined in the following table:

Role	Responsibilities
Business consultants	<ul style="list-style-type: none"> Collect input from all process participants to plan process models Create initial process models
BPM analysts	<ul style="list-style-type: none"> Refine initial process models in Lombardi Authoring Environment Define a data model for each process to identify the data that is passed from one step to the next

Role	Responsibilities
	<ul style="list-style-type: none"> • Coordinate with developers to plan programming of end-user interfaces, integrations with external systems, and variables required for defined data model • Demonstrate process design ideas and progress in iterative playback sessions with management and process participants • Run simulations to identify potential issues and refine process models • Coordinate with project managers to identify business variables to track for reporting purposes
Developers	<ul style="list-style-type: none"> • Coordinate with BPM analysts to understand steps in process models and research options for implementing those steps in Lombardi • Create end-user interfaces and implement integrations with external systems • Create variable types and variables to support data models for processes • Participate in iterative playback sessions with management and process participants to collect feedback to help improve interfaces and integrations • Collect information from IT administrators to facilitate integration with systems external to Lombardi
Project managers	<ul style="list-style-type: none"> • Coordinate with BPM analysts to identify business variables to track for reporting purposes • Run analyses and write custom reports to use tracked data to measure process performance • Train process participants to design and create reports for the processes in which they participate • Using data from analyses and reports, work with BPM analysts to improve process models
Process participants	<ul style="list-style-type: none"> • Provide input to BPM analysts and business consultants so that they can first diagram and then plan the implementation of business processes • Attend iterative playback sessions to determine if process models under development meet the requirements and goals of your team • Work with project managers to learn how to write Lombardi reports that provide insight into vital areas of your business
Administrators	<ul style="list-style-type: none"> • Plan Lombardi installation and install necessary Lombardi servers (typically staging, test, and production servers) • Coordinate with developers to facilitate Lombardi integration with external systems (such as databases, LDAP servers, inventory tracking systems, etc.) • Coordinate with developers to create a versioning strategy for process applications developed in Lombardi • Coordinate with developers to build a Lombardi installation service to handle deployment when process applications are ready to move from the Lombardi development environment to test or production servers

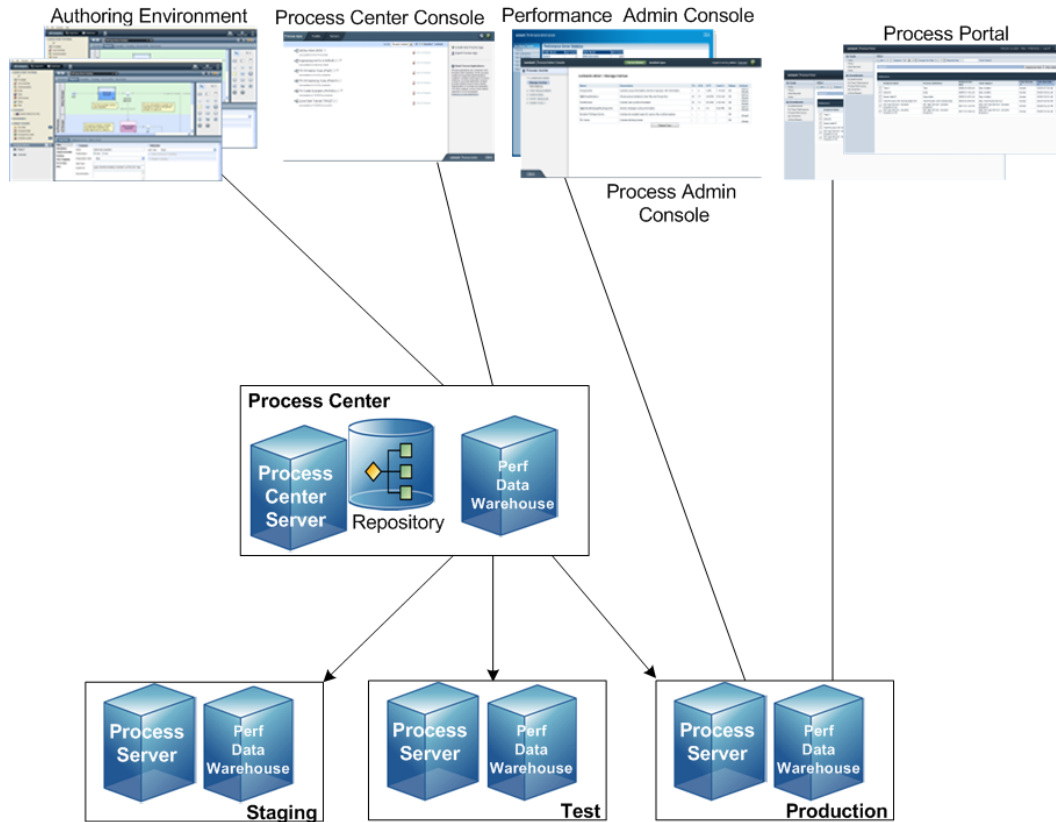
Lombardi product components

IBM WebSphere Lombardi Edition includes the following components. The following diagram illustrates how these components are commonly configured.

Component	Function
Process Center	Provides a central development environment and repository for multiple process authors working in the Process Center Console and other interfaces in Lombardi Authoring Environment. The Process Center includes a Process Center Server and a Performance Data Warehouse, allowing you to build and run process applications and also store performance data for testing and playback purposes during development efforts.
Process Server	Executes the processes and services built in Lombardi Authoring Environment, stored in the Process Center repository, and then installed in a runtime environment.
Performance Data Warehouse	Collects and aggregates process data according to tracking requirements established in Lombardi Authoring Environment.
Process Center Console	Enables you to manage and maintain the Lombardi repository, including management of process applications, workspaces, and snapshots. Also enables installation of process applications on Process Servers in runtime environments.
Authoring Environment	Lombardi Authoring Environment consists of several interfaces to enable process authors to model, implement, simulate, and inspect business processes.
Process Portal	Provides an interface that enables process participants to perform assigned tasks, view the history of tasks, and view the performance of their processes and teams. Using Process Portal, process participants can connect to the Process Center Server or a Process Server in any configured runtime environment, such as test or production environments.
Process Admin Console	Provides an interface that enables administrators to configure and maintain Lombardi Process Servers in any configured runtime environment, such as test or production environments. Also enables administrators to configure and maintain the Process Center Server.
Performance Admin Console	Provides an interface that enables administrators to configure and maintain Lombardi Performance Data Warehouses in any configured runtime environment, such as test or production environments. Also enables administrators to configure and maintain the Performance Data Warehouse included in the Process Center.

Lombardi architecture

The following diagram illustrates a typical IBM® WebSphere® Lombardi Edition configuration:



- From Lombardi Authoring Environment, multiple users connect to the Process Center.
- In Lombardi Authoring Environment, users create process models and supporting implementations (process applications) and store those applications and associated items in the Process Center repository. Authoring Environment users connected to the Process Center can share items.
- The Process Center includes a Process Center Server and Performance Data Warehouse, allowing users working in Lombardi Authoring Environment to run their process applications and store performance data for testing and playback purposes during development efforts.
- From the Process Center Console, administrators install process applications that are ready for staging, testing, or production on the Process Servers in those environments.
- From the Process Center Console, administrators manage running instances of process applications in all configured environments.
- From the Process Portal, end users perform assigned tasks. The Process Center Server and Process Servers in configured runtime environments can run the process applications that create the assigned tasks.
- Using the Process Portal, process participants can connect to the Process Center Server or a Process Server in any configured runtime environment, depending on whether a process is being developed, tested, or has been released to a production environment.
- Lombardi Performance Data Warehouse retrieves tracked data from the Process Server or Process Center Server at regular intervals. Users can create and view reports that leverage this data in Lombardi Authoring Environment and Process Portal.

- From the Process Admin Console and Performance Admin Console, administrators can manage and maintain all Lombardi servers.

Lombardi key terms and concepts

Before using IBM® WebSphere® Lombardi Edition, you should be familiar with the following terms and concepts:

Term or concept	Definition
Business Process Definition (BPD)	When you model a process in Lombardi Authoring Environment, you are creating a Business Process Definition (BPD). A BPD is a reusable model of a process, defining what is common to all run-time instances of that process model.
Pools and Lanes	Each process that you model in Lombardi Authoring Environment includes the default Lombardi pool, which consists of lanes that you designate. Lanes typically represent departments within a business organization. Plus, the lanes in a process model are containers for the activities and events that take place during process execution. For example, a Call Center lane would include all activities to be handled by Call Center personnel during process execution.
Activities	An activity represents a logical unit of work that can be completed by a human or a system during process execution.
Sequence lines	Sequence lines control the sequence of activities and events during process execution.
Services	Services are similar to programs, which you create in Lombardi Authoring Environment to implement activities or to perform one-time or recurring system tasks.
Gateways	Gateways control the divergence and convergence of sequence lines, determining branching, forking, merging, and joining of paths that a process can take during execution.
Events	When modeling processes, you can use events to trigger an action based on a timer, a message arriving from an external system, or some other occurrence such as a run-time exception. Events enable you to control or alter process flow during execution.
Variables	Variables represent the data that passes from one step to another in a process. For example, if you create a process to automate escalation of customer issues, you need to create variables to hold information such as the customer's name and the issue ID. With these variables, each person involved in the process receives information necessary for completing her work.
Coaches	Coaches are user interfaces that you create in Lombardi to collect user input required for an underlying service.
Process Center Console	Interface to the Process Center repository where administrators can create and manage process applications, manage user access to library items, install snapshots on test or production servers, and perform other tasks.
Designer	Authoring Environment interface where you can create process models and supporting implementations.
Inspector	Authoring Environment interface that enables you to step through processes during playbacks and makes it easy to inspect, troubleshoot, and debug running processes and services.
Optimizer	Authoring Environment interface that enables you to simulate process performance during development and then analyze process performance after processes are up and running.
Process applications	Containers in the Process Center repository for the process models and supporting implementations. Ordinarily, a process application includes process models, also called the Business Process Definitions (BPDs), the services to handle implementation of activities and integration with other systems, and any other items required to run the processes. Each process application can include one or more workspaces.

Term or concept	Definition
Toolkits	A collection of library items that can be used across numerous process applications in Lombardi Authoring Environment.
Workspaces	Optional subdivisions in a process application based on team tasks and/or process application versions. When enabled, workspaces allow parallel development to occur with isolation from changes in other workspaces. For example, workspaces enable one team to fix the current version of a process while another team builds a completely new version based on new external systems and a new corporate identity.
Snapshots	You can capture and save the items in a process application at a specific point in time. Usually snapshots represent a milestone or are used for playbacks or for installation.
Global assets	These assets are individual library items that are available to the entire process application in which they reside. For example, if you set environment variables for a process application, those variables are global assets and they can be called from any implementation. The installation service for a process application is also a global asset.
Managed files	Library items created outside of Lombardi Authoring Environment that are part of a process application. For example, you might need an image or Cascading Style Sheet (CSS) for a Lombardi coach. You can create such assets using other tools and store the necessary files in the Process Center repository. Doing so ensures that all required files are available and installed when a project is ready for testing or production.
Installation service	Lombardi service that you can build to handle specific requirements for the installation of a process application on the Process Servers in your test and production environments. An installation service is created by default when a process application is created. The installation service for a process application is a global asset.
Tags	You can mark library items in Lombardi Authoring Environment with custom tags for easy access. For example, you can tag items with your initials so that you can search for and retrieve each item that you worked on.
Favorites	You can mark library items in Lombardi Authoring Environment as favorites for easy access. For example, if you are working on a particular set of services that span several processes, you can mark them as favorites so that you can quickly access them each time you start Lombardi Authoring Environment.
Environment variables	Environment-specific variables that you can set for each process application. These variables are necessary to provide values for each type of environment in which a process will run (development, test, and production). For example, the Process Server host will likely be different for each environment. As global assets, you can call these variables from Java™ Scripts and other implementations in BPDs.

Understanding process development in Lombardi

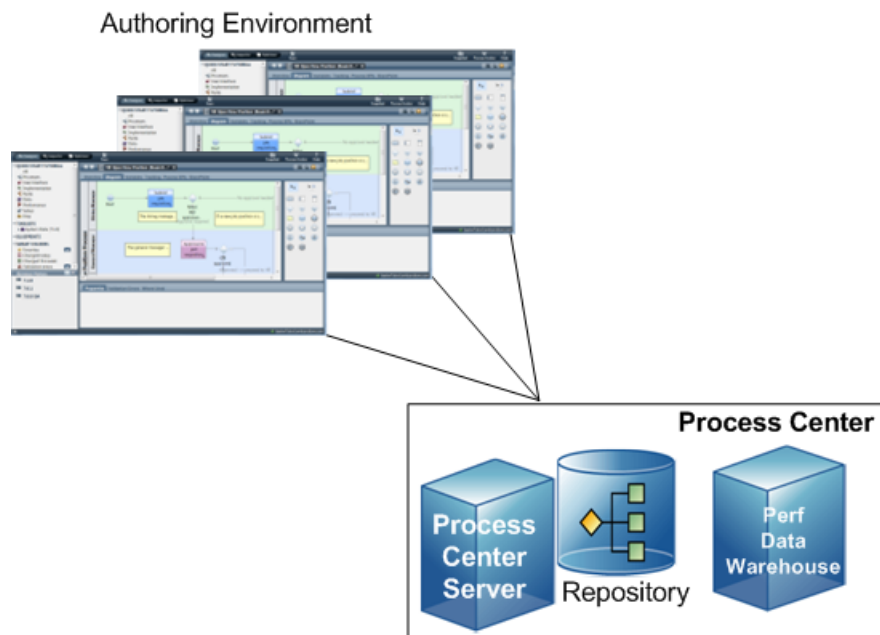
Designing and developing processes in IBM WebSphere Lombardi Edition is easy for large or small teams of developers and BPM analysts because of the following features:

Feature	Benefits
Process Center	<ul style="list-style-type: none"> Provides a central repository for all items (like BPDs and services) that you create in Lombardi Authoring Environment. Multiple Authoring Environment clients can connect to a single Process Center, which enables you to share items. You can see edits or updates made by other users as they occur.
Toolkits	<ul style="list-style-type: none"> Users in Lombardi Authoring Environment can create dependencies on Toolkits in order to re-use the items within.

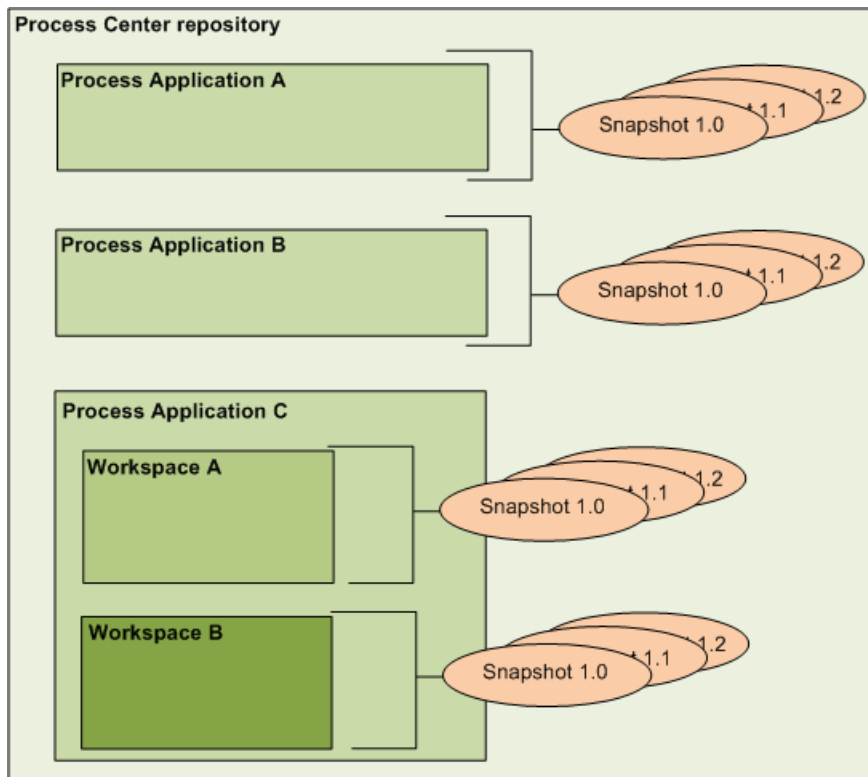
Feature	Benefits
	<ul style="list-style-type: none"> • When Toolkit items are updated, existing dependencies show that updates are available. • Team members with the required permissions can create new Toolkits as projects grow and additional items for re-use are identified.
Snapshots	<ul style="list-style-type: none"> • Enable Lombardi Authoring Environment users to capture and save the items within their process applications at specific points in time. • Snapshots are important in terms of saving key milestones for future reference such as playback versions of your process application, versions that you submit for review, and so on. • You can compare one snapshot to another to determine what has changed in different versions of your process applications. • You can install snapshots of your process applications on the Process Servers in your test and production environments. • You can view previous snapshots of your project assets; you can also run processes or services to compare previous implementations to your current status. • You can choose to copy an older version of an asset to your current project or you can simply revert to an older version of a particular asset if the previous implementation matches your current needs.

Understanding the Process Center

The Process Center serves as a central repository for all project assets created in Lombardi Authoring Environment. When multiple Authoring Environment clients connect to the Process Center as shown in the following image, users can share items (like processes and services) and can also see changes made by other users as they happen:



When you're developing processes in Lombardi, there's a hierarchy available in the Process Center repository which is designed to help you manage your projects. The following image provides a conceptual overview of the repository hierarchy:



As you can see from the preceding diagram, the Process Center repository includes the following:

Process Applications	Containers for the process models and supporting implementations that BPM analysts and developers create in the Designer in Lombardi Authoring Environment.
Workspaces	Optional subdivisions in a process application based on team tasks or process application versions. When enabled, workspaces allow parallel development to occur. Administrators determine if additional workspaces are necessary and, thus, enabled for each process application.
Snapshots	Record the state of the items within a process application or workspace at a specific point in time. Usually snapshots represent a milestone or are used for playbacks or installations. You can compare snapshots and revert to previous snapshots. If an administrator enables workspaces for a process application, he uses a snapshot as the basis for a new workspace.

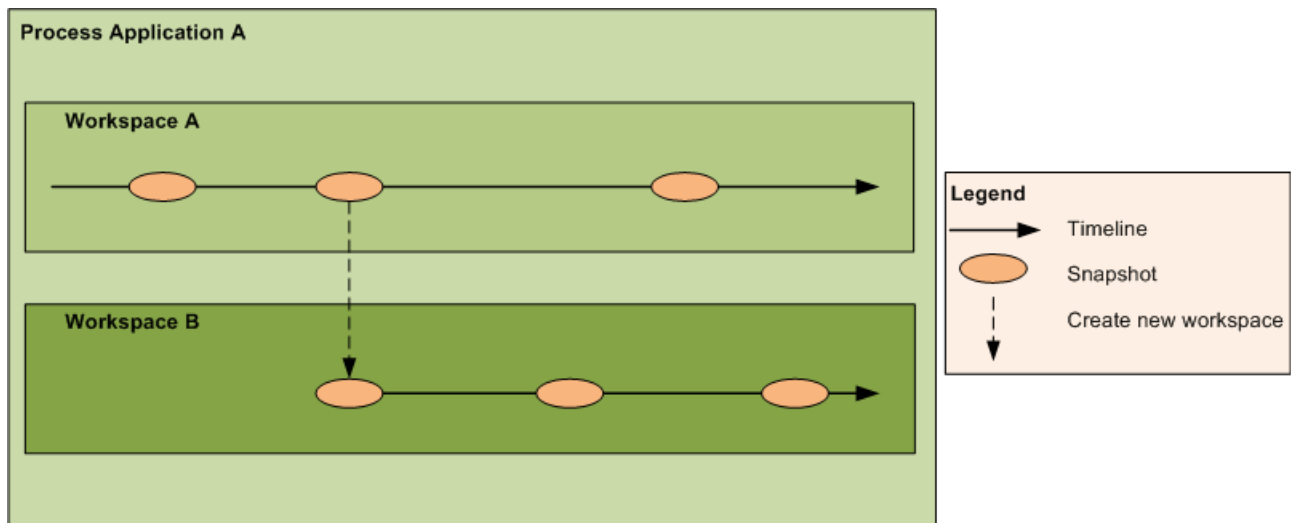
Re-using items in Lombardi

Lombardi enables process developers to re-use existing items both within and across process applications. For example, if you know several services already exist that include Coaches and other library items that you and other developers need, you can access and re-use those items by including them in a toolkit. Then, from your process application, you can add a dependency to the toolkit in which the library items reside. This enables you to pick one of the existing services when choosing the implementation for an activity. The items in the toolkit can also be used by other developers working in different process applications. To learn more about toolkits, see [Managing and using toolkits](#).

Versioning Lombardi items

To version items stored in the Process Center repository, you can save and name snapshots. Doing so enables you to compare one snapshot to another to find differences. For example, if a developer fixed a problem with a service and took a snapshot at that point, and then a different developer made several additional changes to the same service and took a new snapshot, the project manager could compare the two snapshots to determine which changes were made when and by whom. If the project manager decided that the additional changes to the service were not worthwhile, he could revert back to the snapshot of the original fix.

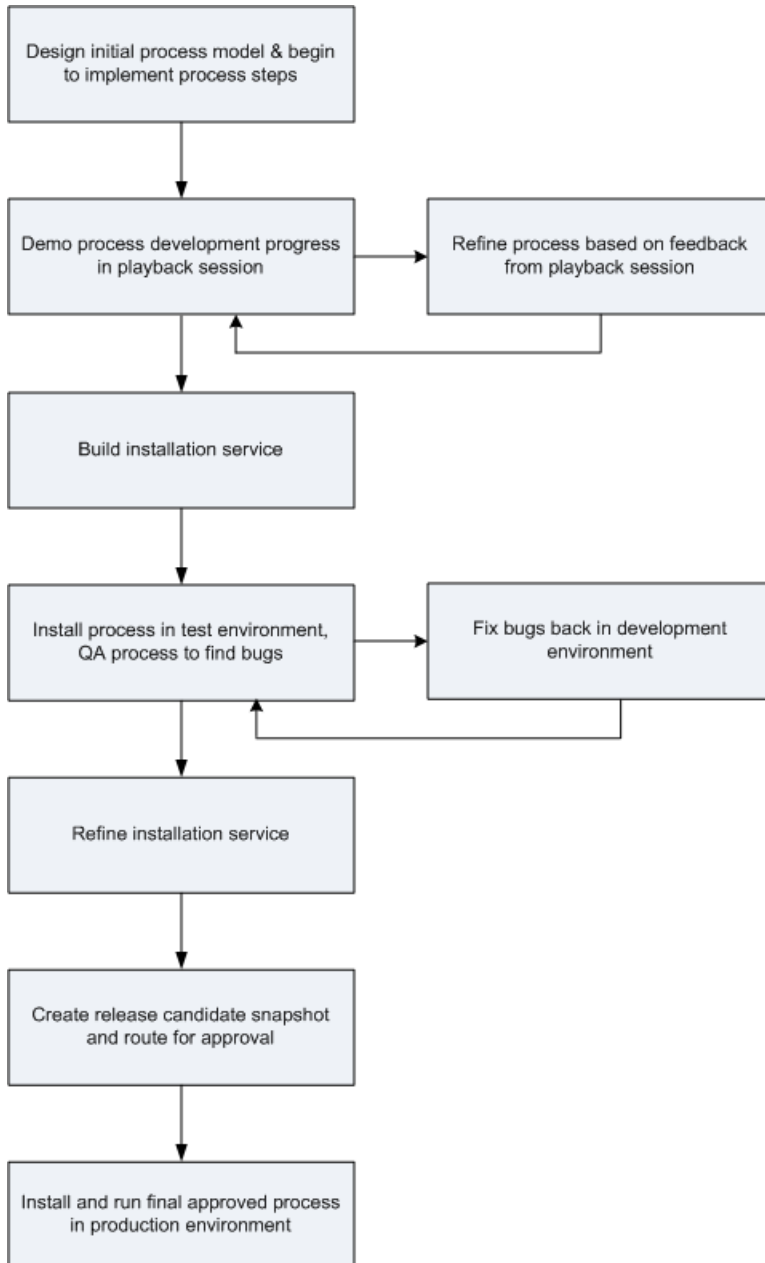
The following diagram illustrates how snapshots capture points in time and how administrators can use a snapshot to create a new workspace if additional workspaces become necessary:



To learn more about snapshots, see [Managing snapshots](#).

Planning for process deployment and installation

When developing processes in Lombardi, you need to plan for the eventual installation of your process applications on servers in your test and production environments. The following diagram illustrates the lifecycle of a typical process development effort:



Initially, you can work exclusively in your development environment. But, as you can see from the preceding flow chart, you need to configure Process Servers and Performance Data Warehouses for both your test and production environments. The preceding flow chart also includes steps for building and refining an installation service so that you can easily install your process applications on the Process Servers in each configured environment. For more information about customizing installation services and installing your process applications, see [Running and installing processes](#).

Starting Lombardi Authoring Environment

All process development takes place in Lombardi Authoring Environment. The following topics provide more details about logging in, navigating the initial views, and accessing each available interface in Lombardi Authoring Environment.

After starting Lombardi Authoring Environment as outlined in the following sections, you can run a sample process by following the instructions in *Lombardi Quick Start Tutorial*. The tutorial provides everything you need to start using all Lombardi Authoring Environment features.

Logging in

Start Lombardi Authoring Environment one of the following ways:

- Double-click the **Lombardi Authoring Environment** Windows® desktop shortcut
- Choose **Start > IBM WebSphere Lombardi Edition > Lombardi Authoring Environment** from the Windows desktop
- Go to [Lombardi_home] \Lombardi Authoring Environment and run **eclipse.exe**

When the Log In dialog opens, provide the following information:



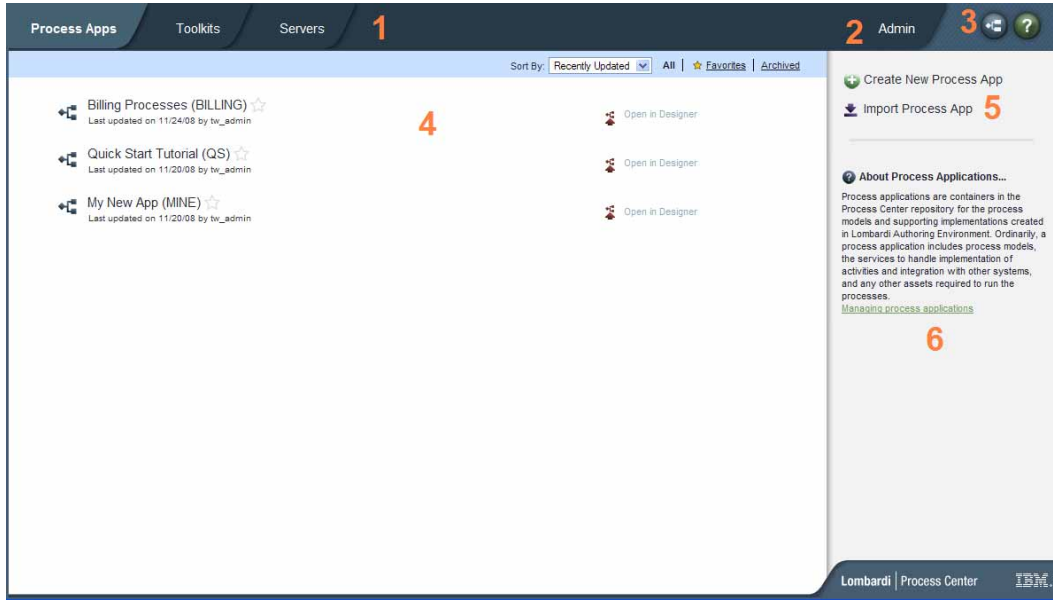
Contact your Lombardi administrator if you do not already have a user account.

User name	Your Lombardi user name.
Password	Your Lombardi password.

You are connected to the Process Center designated during installation of Lombardi Authoring Environment.

Navigating initial views

The first time you start Lombardi Authoring Environment, it opens to the Process Center Console:



The Process Center Console enables you to create and manage process applications, install snapshots on test and production servers and perform other tasks. The following table describes each numbered area in the preceding image of the Process Center Console:

1	Click one of the available tabs to pick the items you want to create or manage. Clicking Process Apps takes you to the page shown in the preceding image where you can create and manage process applications. Clicking Toolkits takes you to a similar page for managing toolkits. Click Servers to manage the servers configured in your environment.
2	Click Admin to manage access to the Process Center repository.
3	Clicking the Designer button takes you directly to the Designer interface in Lombardi Authoring Environment. To learn more about the Designer and other interfaces in Lombardi Authoring Environment, see the following section. Clicking the Help button opens the online help for Lombardi Authoring Environment.
4	This is the main area of the Process Center Console where the items that you are currently managing, such as process applications, snapshots, or servers, are displayed. You can click the All, Favorites, or Archived options to filter the items displayed. Click an item to view and manage its settings. In the example, you can click one of the listed process applications to view and manage its snapshots, history, and general settings. To open a specific process application in the Designer, click the Open in Designer option for the process application that you want to access.
5	Use these options to either create a new process application or import one.
6	Informational dialogs such as this one are available throughout the Process Center Console. You can click the link shown to learn more about the subject, which in this case is process applications.

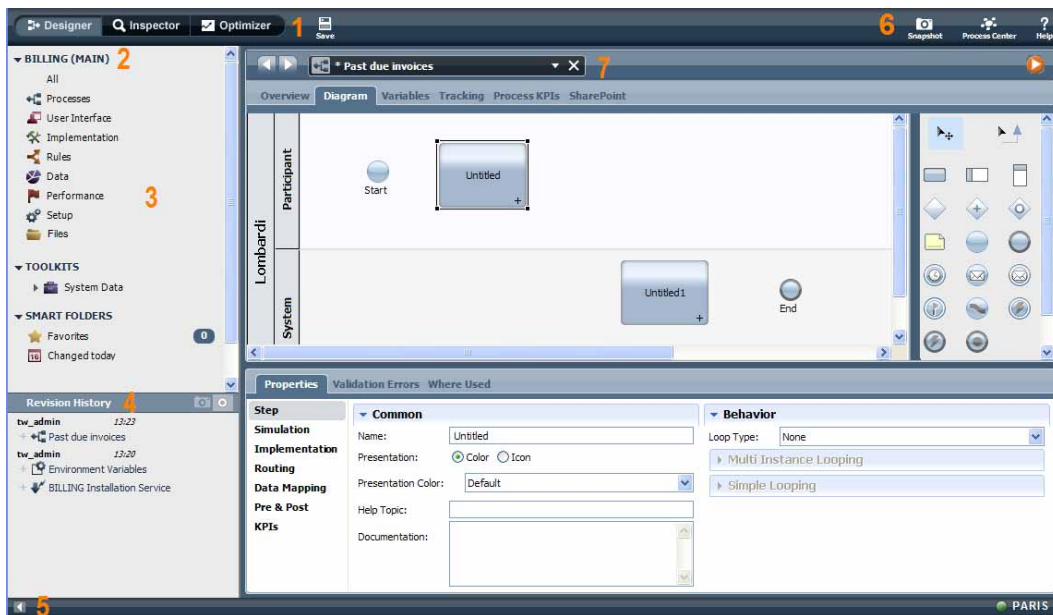
- To create a process application and get started developing processes in Lombardi, see [Creating your first Lombardi project](#).
- To learn about all of the administrative tasks you can perform in the Process Center Console, see [Managing the Process Center repository](#).



You can also access the Process Center Console by opening your Web browser to the following location: `http://[host_name]:[port]/ProcessCenter`. You can log in using your Lombardi user name and password. When accessing the Process Center Console from a browser, you cannot select library items such as process applications and immediately open them in the Designer view. To work interactively with the Designer and other available interfaces, you must start the Authoring Environment.

Accessing and using Lombardi Authoring Environment interfaces

From the Process Center Console, you can click the **Designer** button or an **Open in Designer** option to begin developing processes in Lombardi Authoring Environment. The following image shows the Designer interface and each functional area:



You can use the Designer interface to develop process models and their underlying implementations, such as services. The following table describes each numbered area in the preceding image of the Designer interface in Lombardi Authoring Environment:

1	Click the appropriate button to open the interface that you want in Lombardi Authoring Environment, including the Optimizer and Inspector views.
2	Shows the process application currently open. In this sample, the Billing process application is open.
3	Shows the types of library items included in the currently open process application. Click a category, such as Processes, to see the processes that you can open and alter.
4	Shows the revision history for the currently open process application. In this sample, a user has recently added items to the open process.
5	Click this icon to hide the library and revision history. The icon enables you to toggle back and forth if you want to alternately hide and view the library and revision history. This toggle control is available in all Lombardi Authoring Environment interfaces: Designer, Optimizer, and Inspector.
6	Use the icons to create snapshots, access the Process Center Console, or access online assistance.

7	Shows the library item currently open for editing in the Designer. In this sample, the user has a process open and is working in the diagram, palette, and properties to create the steps of the process.
---	---

To learn more about the tasks that you can perform in each available interface in Lombardi Authoring Environment, including the Process Center Console, see [Lombardi tasks](#).

Creating your first Lombardi project

When you're ready to start building processes in the Designer view, you first need to create a process application for the project using the Process Center Console. Each time a new project begins, you can create a new process application for the new project as outlined in the following table.

Before you get started, you must have access to the Process Center repository to complete the tasks outlined in the following table. See [Managing access to the Process Center repository](#) for more information.

The following steps list the basic tasks for getting started:

Task	See...
1. Add the users and groups who need to develop processes and implementations in the repository. (Requires administrative access to the repository.)	Adding users and groups
2. Create a new process application.	Creating new process applications in the Process Center Console
3. Grant access to those users who need to work in this new process application.	Managing access to process applications and toolkits
4. Add dependencies to any toolkits that your developers need.	Creating a toolkit dependency in the Designer view
5. Create snapshots of the process application for milestones such as playbacks and reviews.	Creating new snapshots in the Process Center Console

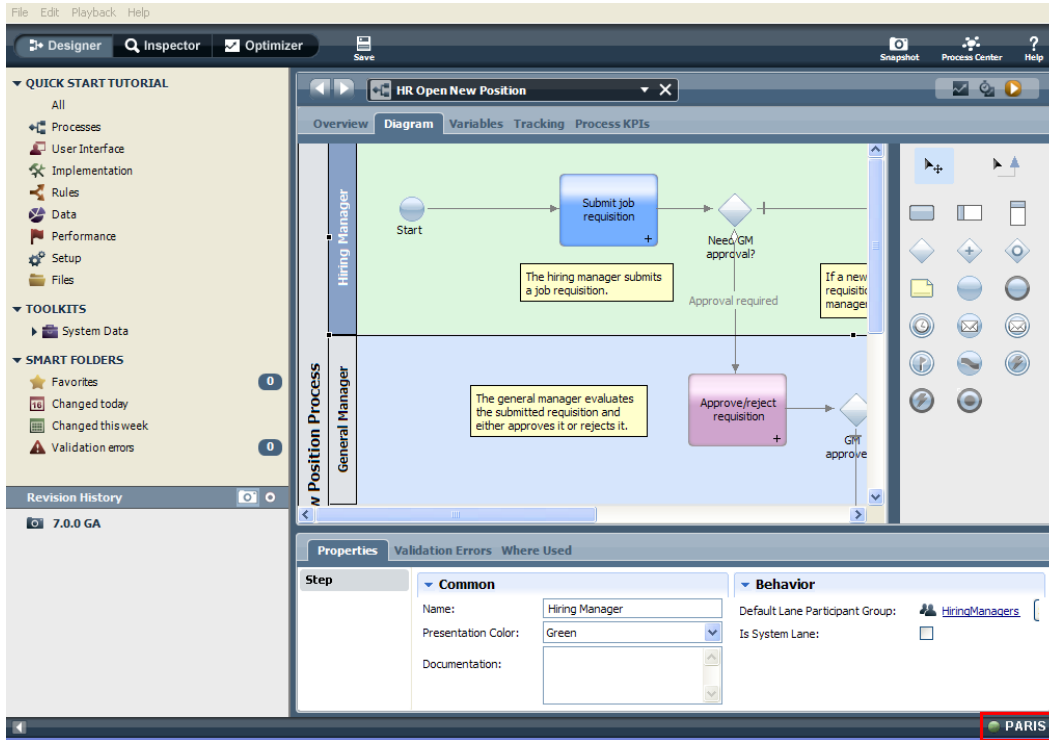


You can optionally enable and create workspaces in your process applications as described in [Enabling workspaces in the Process Center Console](#) and [Creating new workspaces in the Process Center Console](#).

Authoring Environment tips and shortcuts

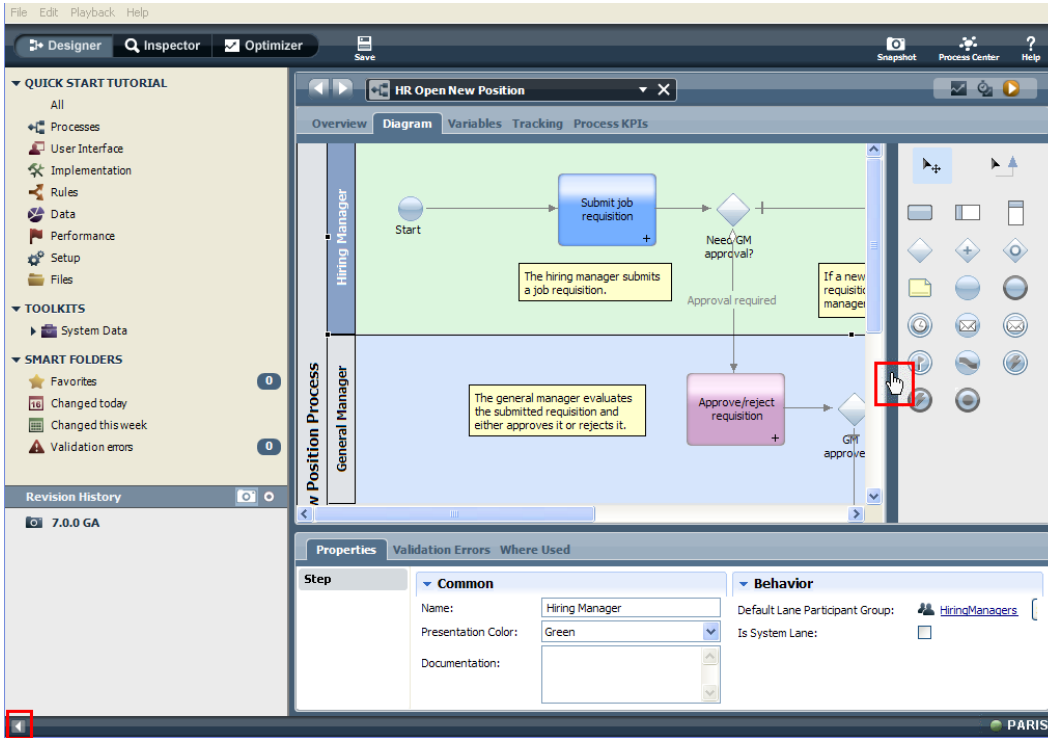
When you start using the Designer interface in Lombardi Authoring Environment, keep the following tips in mind:

- To determine your connection status, check the lower right corner of the Authoring Environment as shown in the following image:

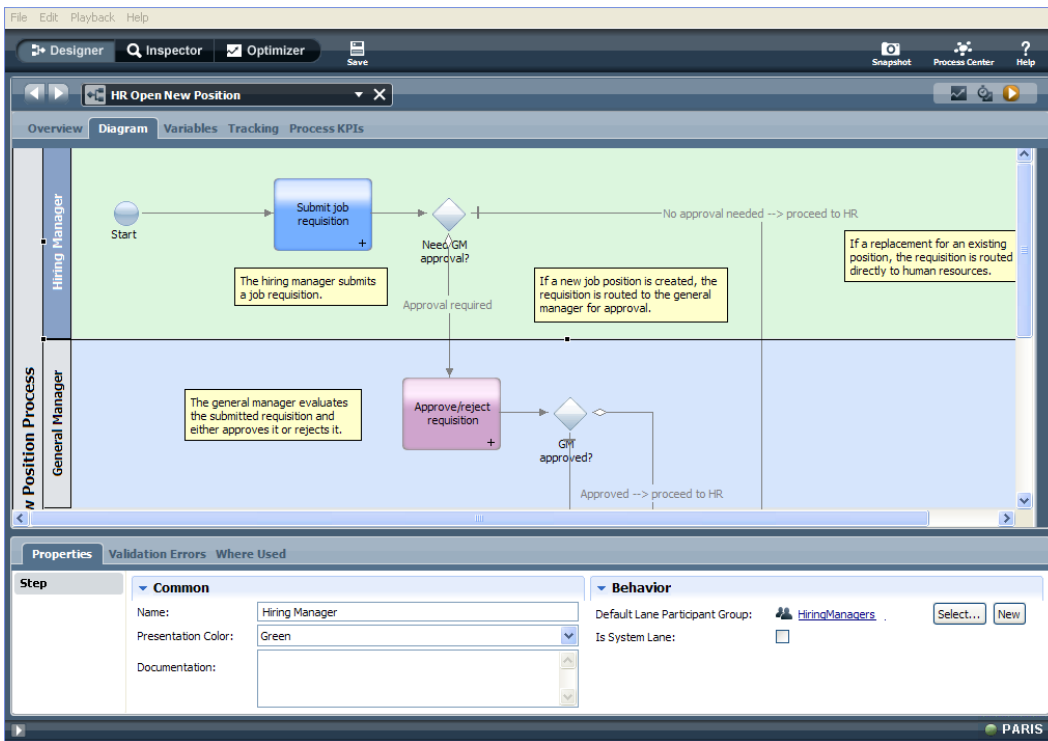


Indicator color	Connection to Process Center Server status
Green	Good connection
Yellow	Slow connection which could cause issues with concurrent edits
Orange	Even slower connection and more potential issues with concurrent editing
Red	Connection has been lost; check with your Lombardi administrator to ensure the Process Center Server is up and running

- To maximize the space available for your process diagram, you can hide the library by clicking the toggle at the bottom of the Revision History. Then click the left margin of the palette as shown in the following image:

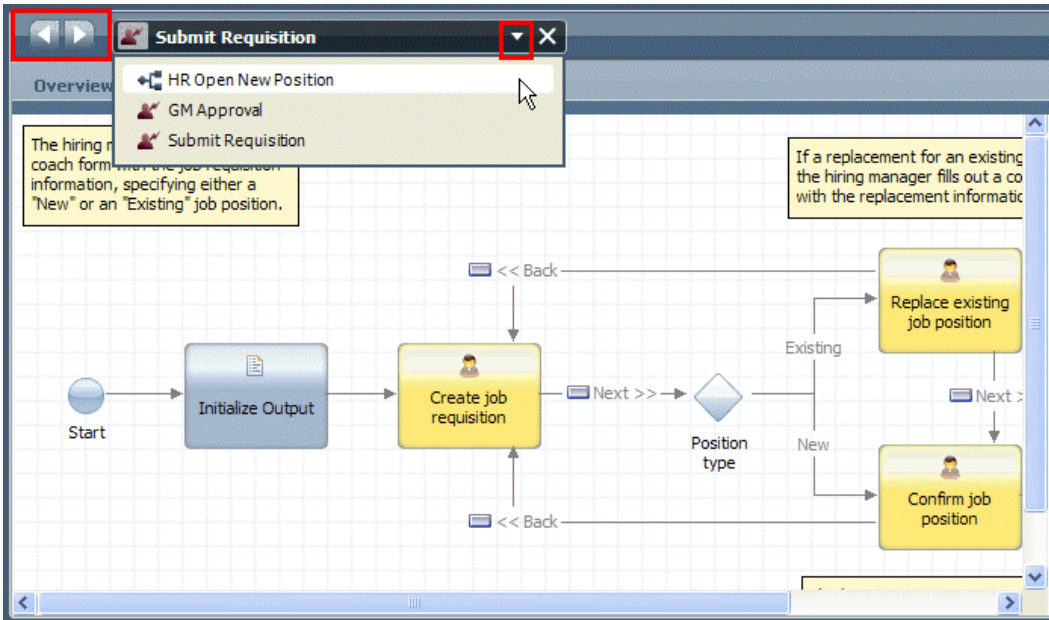


When you do, the space available for your diagram is maximized as shown in the following example:

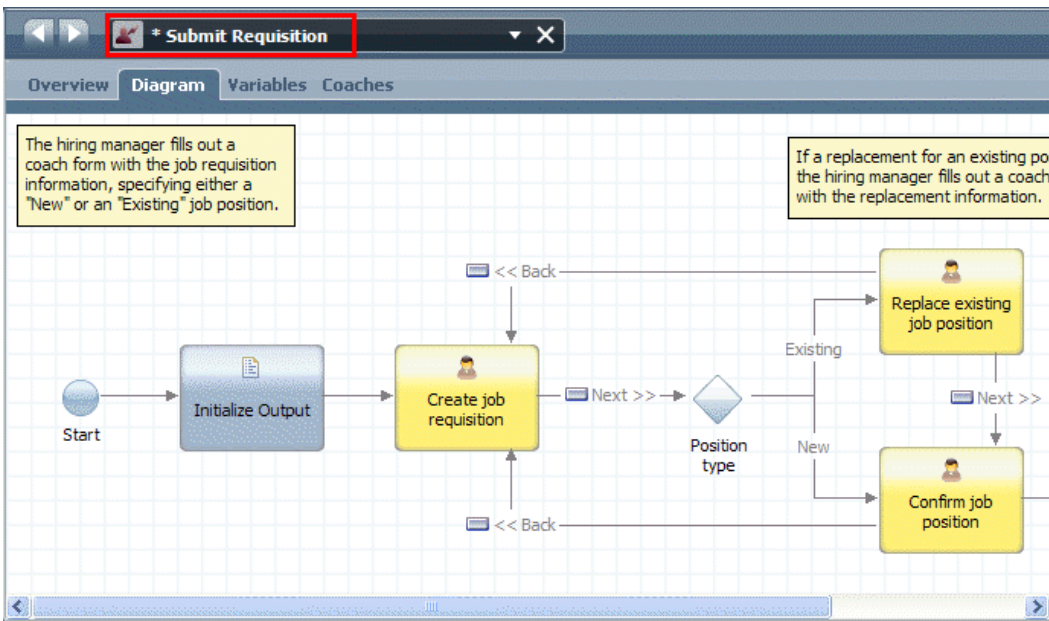


Click the toggle icon and the palette margin to restore the library and the palette, respectively.

- To move from one open library item to another in the Designer, click the arrow keys or the drop-down menu shown in the following image:

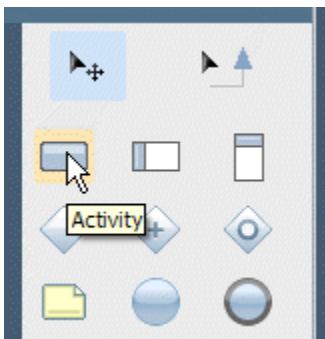


- When you make changes to a library item that have not been saved, the Designer displays an asterisk with the item name as shown in the following example:



- To create a new library item while working in the Designer, press **Ctrl+Shift+N**.
- To open an existing library item while working in the Designer, press **Ctrl+Shift+O**.
- To undo changes made in the diagram for a process or service, press **Ctrl+Z**. To get back a change, press **Ctrl+Y**.

- To zoom in on a process or service diagram, press `Ctrl` and the `+` key. To zoom out, press `Ctrl` and the `-` key. You can also press `Ctrl` and move your mouse wheel up to zoom in or press `Ctrl` and move your mouse wheel down to zoom out.
- You can hold your mouse over each component in the palette to see a description as shown in the following image:



For a complete description of each process component, see [Understanding process components](#). For a complete description of each service component, see [Understanding service components](#).

- You can capture your development progress in snapshots as described in [Creating new snapshots in the Designer view](#).
- You can revert to a previous snapshot (version) of a library item as described in [Reverting to a previous version of a library item](#).
- You can copy the previous snapshot (version) of a library item to your current project as described in [Copying a library item from a snapshot](#).
- You can add a dependency to a toolkit to use the library items from that toolkit as described in [Creating a toolkit dependency in the Designer view](#).
- You can see updates made by other users as described in [Understanding concurrent editing](#).
- For quick and easy access of particular library items, you can create favorites as described in [Creating favorites](#).
- To group library items for easy access, follow the instructions in [Tagging library items](#).
- To create smart folders of library items, follow the instructions in [Organizing library items in smart folders](#).
- To move or copy library items from one process application to another, follow the instructions in [Copying or moving library items](#).
- To add and manage external files as part of your Lombardi project, see [Managing external files](#).

Lombardi tasks

The following table lists the types of tasks that you can perform in the Process Center Console and Authoring Environment and provides links to topics that provide detailed instructions:

Task	Authoring Environment interface	Description	See...
Creating and managing process applications	Process Center Console	Create containers where BPM analysts and developers can create process models and underlying implementations.	Managing process applications
Creating and managing workspaces	Process Center Console	Optionally create subdivisions in process applications to allow parallel development to occur in Lombardi Authoring Environment.	Managing workspaces
Creating and managing toolkits	Process Center Console and Designer	Create special containers to enable users in Lombardi Authoring Environment to share library items across process applications.	Managing and using toolkits
Managing user access	Process Center Console	Establish who can access process applications and toolkits.	Managing access to the Process Center repository
Creating and managing snapshots	Process Center Console and Designer	Capture and save the items within a process application at specific points in time.	Managing snapshots
Creating process models	Designer	Read about the building blocks available and how to use them to create a process model in Lombardi.	Modeling processes
Implementing process steps (activities)	Designer	Read about the implementation options for activities in your process and determine the best option for each step.	Advanced modeling tasks
Running and inspecting processes	Inspector	Use the Inspector to debug and then playback (demonstrate) your processes.	Running and debugging processes with the Inspector
Creating reports	Designer	Define the data to track and then analyze that data using Lombardi reports. You can create custom reports to analyze data specific to your processes.	Creating and configuring reports
Running simulations of your processes	Optimizer	Simulate your processes during development to test and refine process designs before deployment and installation.	Configuration requirements for simulation
Optimizing your processes	Optimizer	Analyze your processes after they're up and running using tracked data stored in the Performance Data Warehouse. Running historical analyses using Lombardi Optimizer enables you to measure and then improve the efficiency of your processes.	Configuration requirements for optimization
Building installation services	Designer	Build the special Lombardi service that handles installation of your process applications on the Process Server in each configured environment.	Building installation services
Installing process applications	Process Center Console	When a process application is ready for testing or production, install a snapshot of the application on the	Releasing and installing processes

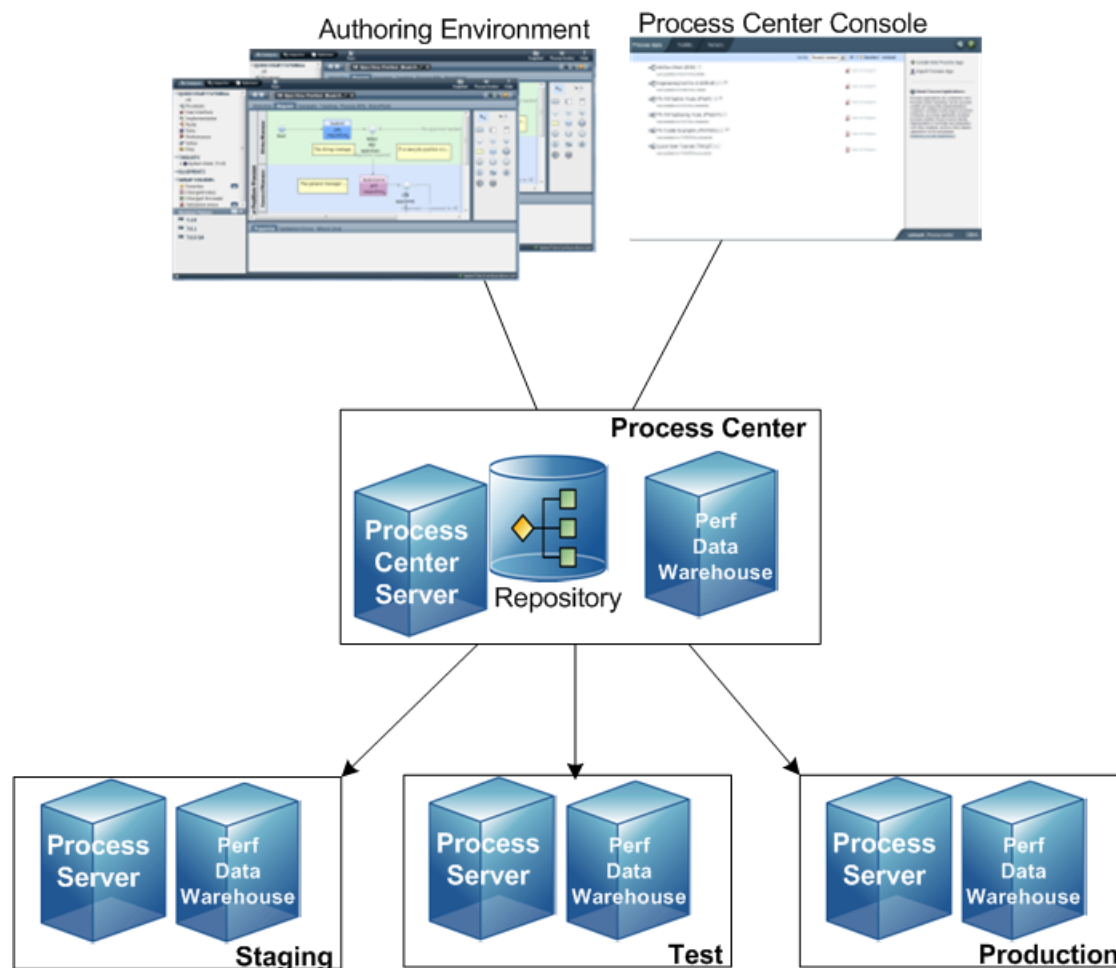
Task	Authoring Environment interface	Description	See...
		Process Server in the appropriate environment.	

Managing the Process Center repository

The Process Center includes a repository for all processes, services, and other assets created in Lombardi Authoring Environment. The following sections provide an introduction to the types of tasks involved in managing and maintaining the Process Center repository.

Overview

The Process Center Console provides the tools that you need to maintain the repository. The following figure illustrates how the Process Center Console provides access to the Process Center:



- From the Process Center Console, you can create process applications and toolkits and grant other users access to those process applications and toolkits.
- Users in Lombardi Authoring Environment create process models, services, and other assets within process applications.
- The Process Center includes a Process Center Server and Performance Data Warehouse, allowing users working in Lombardi Authoring Environment to run their processes and store performance data for testing and playback purposes.

- From the Process Center Console, administrators install process applications that are ready for testing or production on the Process Servers in those environments.
- From the Process Center Console, administrators manage running instances of process applications in configured environments.

Where to perform tasks


The Process Center Console provides a convenient location for users to create and maintain high-level library items such as process applications and toolkits. For users who are primarily administrators and do not actively work in the Designer view, the Process Center Console enables you to provide a framework in which BPM analysts and developers can build their processes and underlying implementations. Another primary task for administrators is managing access to the Process Center repository by setting up the appropriate authorization for users and groups.

Those users with appropriate authorization can perform some administrative tasks directly in the Designer view in Lombardi Authoring Environment. For example, if a developer wants to capture the state of all project assets at a particular milestone, with write access to the process application, he can create a snapshot while working in the Designer view.

The procedures in the following sections provide instructions for users working in the Process Center Console as well as the Designer view.

To learn more

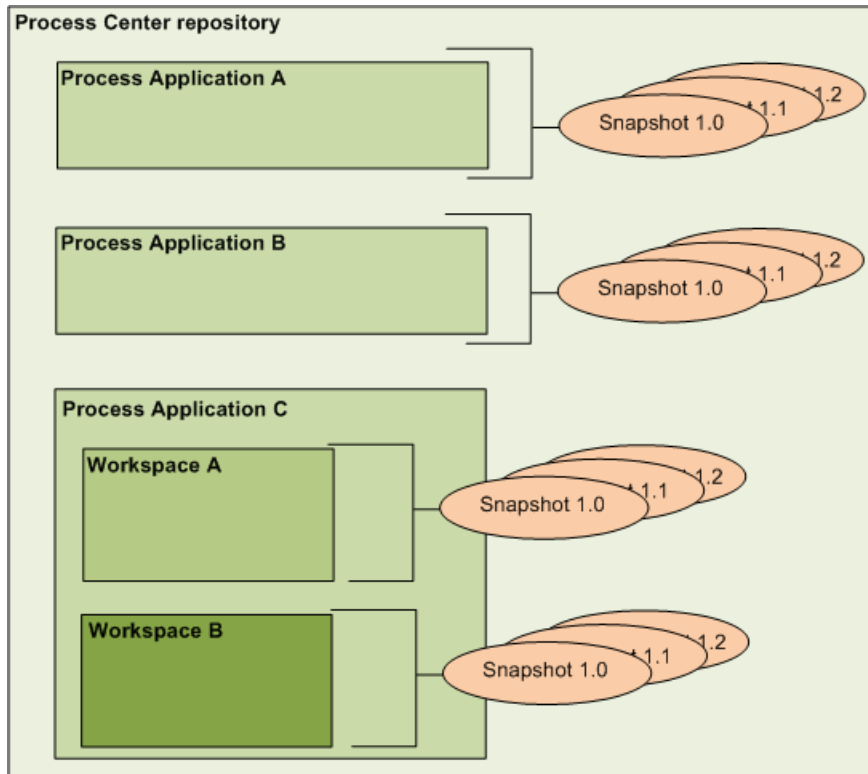
See the following topics to learn more about managing the Process Center repository:

To learn how to...	See..
Create, edit, and archive process applications, workspaces, and snapshots in the Process Center Console. Also learn how to create and compare snapshots in the Designer view.	Managing process applications, workspaces, and snapshots
Create and archive toolkits in the Process Center Console. Also learn how to create and update toolkit dependencies in the Designer view.	Managing and using toolkits
<p>Monitor and configure installed snapshots on each connected Process Server.</p>  <p>To learn how to install snapshots of process applications to connected Process Servers, see Releasing and installing processes.</p>	Managing Lombardi servers
Manage access to the overall repository as well as access to individual process applications and toolkits.	Managing access to the Process Center repository
Open and edit library items in the Designer view, as well as create favorites, tag specific items, organize items in smart folders, and revert to a previous snapshot of a library item.	Managing library items in the Designer view
Manage images, style sheets, JAR files, and other assets developed outside of Lombardi within the Designer view.	Managing external files

Managing process applications, workspaces, and snapshots

Overview

The Process Center repository provides a structured hierarchy to help you manage multiple process development efforts:



As you can see from the preceding figure, the Process Center repository includes the following:

Process applications	Containers for the process models and supporting implementations that BPM analysts and developers create in Lombardi Authoring Environment. You should create a process application for each business process to be automated using Lombardi.
Workspaces	Optional subdivisions in a process applications based on team tasks or process application versions. You can determine if additional workspaces are necessary for each process application and, if so, enable them at any time.
Snapshots	Record the state of the items within a process application or workspace at a specific point in time. From the Process Center Console, you can create snapshots of your process applications and you can also install particular snapshots of your process applications on the Process Servers in staging, test, and production environments.

Creating and maintaining high-level library items

See the following topics to learn how to create and maintain process applications, workspaces, and snapshots:

Task	Description	See...
Creating, importing, and archiving process applications	Create and maintain containers where BPM analysts and developers can develop process models and underlying implementations.	Managing process applications
Enabling, creating, editing, and archiving workspaces	Create subdivisions in process applications to enable development of separate versions of the same application in Lombardi Authoring Environment.	Managing workspaces
Creating, comparing, and archiving snapshots	Capture and save the items within a process application or workspace at specific points in time.	Managing snapshots



To learn how to create and maintain toolkits, see [Managing and using toolkits](#).

Managing process applications

The following topics describe how to create, clone, import, and perform other maintenance tasks for process applications.

Before performing any of the following tasks, you should:

- Start Lombardi Authoring Environment and open the appropriate view as explained in [Starting Lombardi Authoring Environment](#).
- To create and import process applications, you must have access to the Process Center repository. See [Managing access to the Process Center repository](#) for more information.

Creating new process applications in the Process Center Console

1. Select the **Process Apps** tab.
2. Click the **Create New Process App** option shown in the following image:

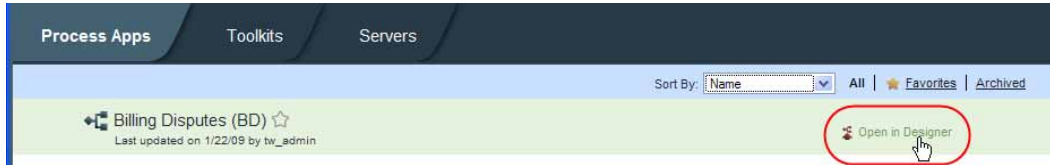


3. In the Create New Process App dialog, enter a name and an acronym for your process application.

The acronym for a process application must be unique and is limited to seven characters. Lombardi uses the acronym as an identifier for this process application and the library items that it contains. For example, when manipulating the items within the process application using the Lombardi JavaScript API, you can use the acronym to specify the namespace of the items.

Providing a description is optional. When you enter a description, you can view it in the Process Center Console by clicking the question mark next to the process application name.

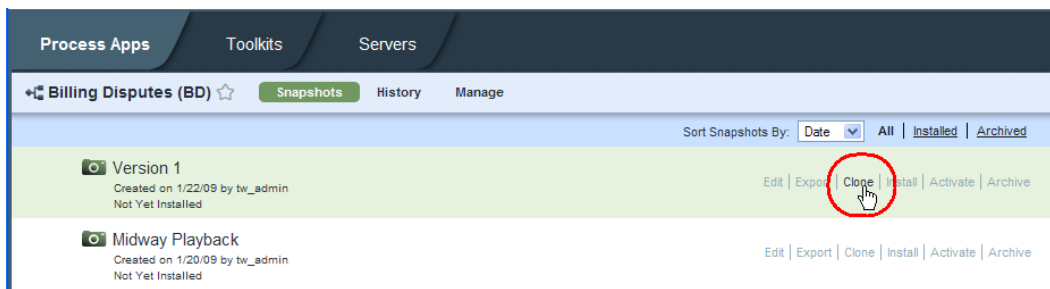
4. To create library items in the process application or perform other edits, click the **Open in Designer** option shown in the following image:



If you want to take advantage of workspaces in this process application, see [Enabling workspaces in the Process Center Console](#).

Cloning process applications in the Process Center Console

1. Select the **Process Apps** tab.
2. In the list of process applications, click on the process application that you want to clone.
3. Click the **Clone** option next to the snapshot that you want to use as the basis for your new process application as shown in the following image:



The Process Center Console opens the Process Apps tab and displays the cloned application with **COFY** added to the end of the original name and the number 2 added to the end of the original acronym.

To change the name and acronym of the cloned application, click the application to open it, click the **Manage** option, and then edit the text in the appropriate fields.

To create library items in the process application or perform other edits, click the **Open in Designer** option.



If you want to take advantage of workspaces in this process application, see [Enabling workspaces in the Process Center Console](#).

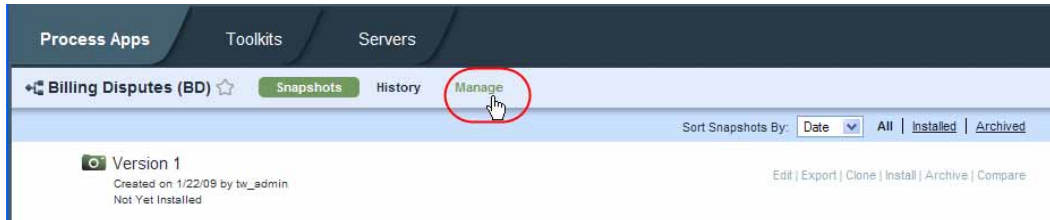
Copying or moving library items from one process application to another in the Designer view

You can copy or move library items to an existing or new process application as described in [Copying or moving library items](#).

Archiving process applications in the Process Center Console

If a process application is no longer used, you can archive it. When you archive a process application, it no longer appears in the list of all process applications in the Process Center Console and you must restore it before you can open it in the Designer view.

1. Select the **Process Apps** tab.
2. In the list of process applications, click on the process application that you want to archive.
3. Click the **Manage** option shown in the following image:



4. Click the **Archive Process App** option as shown in the following image:



5. When prompted, click the **Archive** button to confirm that you want to archive this process application.
6. To view or restore archived process applications, click the **Archived** filter in the Process Apps tab as shown in the following image:



Importing and exporting process applications from the Process Center Console

You can import process applications from other Process Center repositories and you can also export process applications.



To import assets from previous versions of Lombardi, see [Importing files from previous versions of Lombardi](#).

To import process applications:

1. Select the **Process Apps** tab.
2. Click the **Import Process App** option shown in the following image:



- In the Import Process App dialog, click the **Browse** button to locate the Lombardi export (.twx) file that you want to import.



Process applications that you import should have unique acronyms. If an acronym is not unique, import completes with a warning, but attempts to install snapshots of the process application on test and productions servers will fail with an error.

- Click the **Next** button to import the selected .twx file.

In the Import Process App dialog, click to expand the sections that show the snapshots that will be imported and the snapshots that are already available (and will not be imported).

- Click the **Import** button.

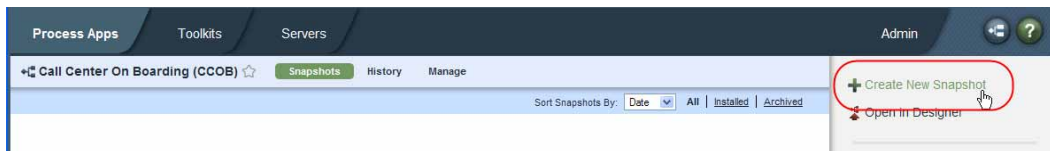
When the import is complete, the imported process application is included in the list in the Process Apps tab. You can grant access to other users as described in [Managing access to process applications and toolkits](#).

Repository administrators can see a log of all imports by clicking the **Admin** option at the top-right of the Process Center Console and then clicking **Import and Export Log**.

To export process applications:

- Select the **Process Apps** tab.
- In the list of process applications, click on the application that you want to export.
- Find the snapshot that you want to export.

If a snapshot does not exist, create one by clicking **Create New Snapshot** as shown in the following image:



- Click the **Export** option for the snapshot as shown in the following image:



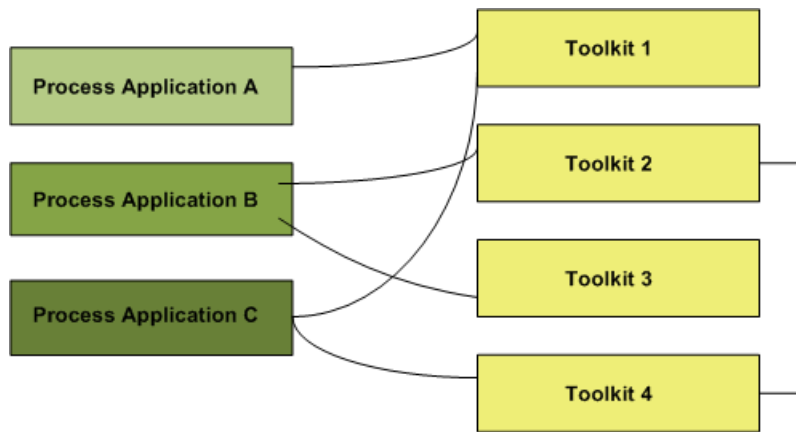
- Locate the directory to which you want to save the export (.twx) file, name the file, and then save it.

The exported file can be imported into any Process Center repository.

Repository administrators can see a log of all exports by clicking the **Admin** option at the top-right of the Process Center Console and then clicking **Import and Export Log**.

Managing and using toolkits

You can create toolkits to enable Authoring Environment users to share library items across process applications. The following figure shows how process applications can share library items from one or more toolkits:



In the preceding figure notice that toolkits can also share library items from other toolkits.

If a user has access to a toolkit, he can create a dependency on the toolkit and use the library items within it for his process development efforts. See the following sections to learn how to create and maintain toolkits, as well as how to use them during process development.

Before performing any of the following tasks, you should:

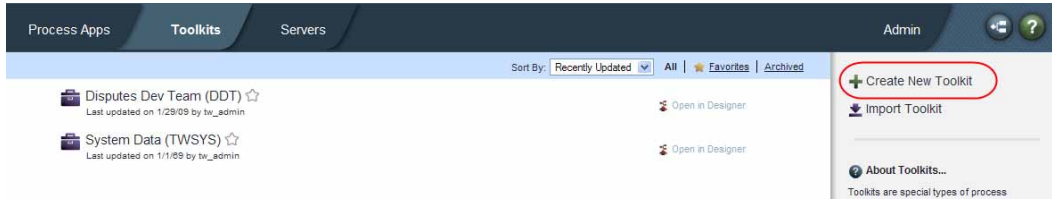
- Start Lombardi Authoring Environment and open the appropriate view as explained in [Starting Lombardi Authoring Environment](#).
- To create and import toolkits, you must have access to the Process Center repository. See [Managing access to the Process Center repository](#) for more information.

About Lombardi System Data toolkit

During Lombardi installation, the System Data toolkit is imported into the Process Center repository. Each process application and toolkit that you create automatically includes a System Data toolkit dependency so that you have access to the assets that all Lombardi projects require, such as standard variable types, standard charts for reports, and so on. You cannot edit or change the library items in the System Data toolkit, but you can open the toolkit and view the items within it as described in [Variable types in Lombardi](#) and [Using Lombardi SQL Integration services](#).

Creating toolkits in the Process Center Console

1. Select the **Toolkits** tab.
2. Click the **Create New Toolkit** option shown in the following image:



3. In the Create New Toolkit dialog, enter a name and an acronym for your toolkit.

The acronym for a toolkit must be unique and is limited to seven characters. Lombardi uses the acronym as an identifier for this toolkit and the library items that it contains. For example, when manipulating the items within the toolkit using the Lombardi JavaScript API, you can use the acronym to specify the namespace of the items.

Providing a description is optional. When you enter a description, you can view it in the Process Center Console by clicking the question mark next to the toolkit name.

4. To create library items in the toolkit or perform other edits, click the **Open in Designer** option shown in the following image:



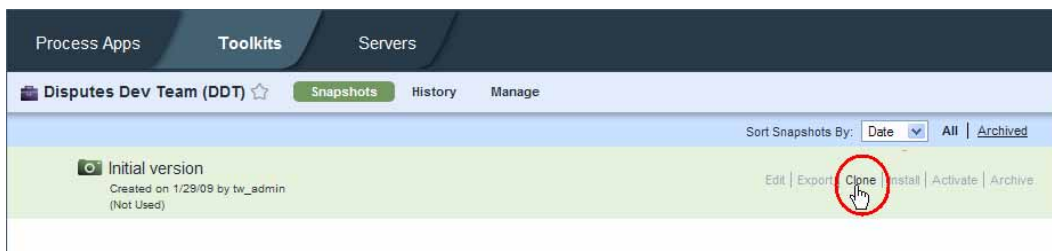
To move or copy library items from another process application or toolkit into this toolkit, see [Copying or moving library items](#).



If you want to take advantage of workspaces in this toolkit, see [Enabling workspaces in the Process Center Console](#).

Cloning toolkits in the Process Center Console

1. Select the **Toolkits** tab.
2. In the list of toolkits, click on the toolkit that you want to clone.
3. Click the **Clone** option next to the snapshot that you want to use as the basis for your new toolkit as shown in the following image:



The Process Center Console opens the Toolkits tab and displays the cloned toolkit with `COPY` added to the end of the original name and the number 2 added to the end of the original acronym.

To change the name and acronym of the cloned toolkit, click the toolkit to open it, click the **Manage** option, and then edit the text in the appropriate fields.

To create library items in the toolkit or perform other edits, click the **Open in Designer** option.



If you want to take advantage of workspaces in this toolkit, see [Enabling workspaces in the Process Center Console](#).

Moving or copying library items to a toolkit in the Designer view

You can copy or move library items to an existing or new toolkit as described in [Copying or moving library items](#).



Imported toolkits are immutable, which means that no one can change the items within an imported toolkit. For more information, see [Importing and exporting toolkits from the Process Center Console](#).

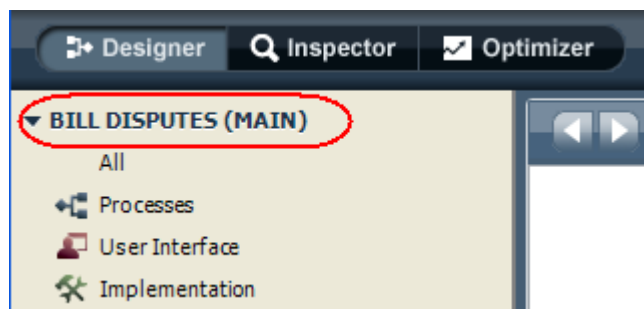
Creating a toolkit dependency in the Designer view

When you create a dependency on a toolkit, you can use the library items from that toolkit for the implementation of the process steps you are building in your current project. For example, after creating a dependency on a toolkit that includes several services, the Designer view automatically makes those services available when a developer is choosing the implementation for an activity.



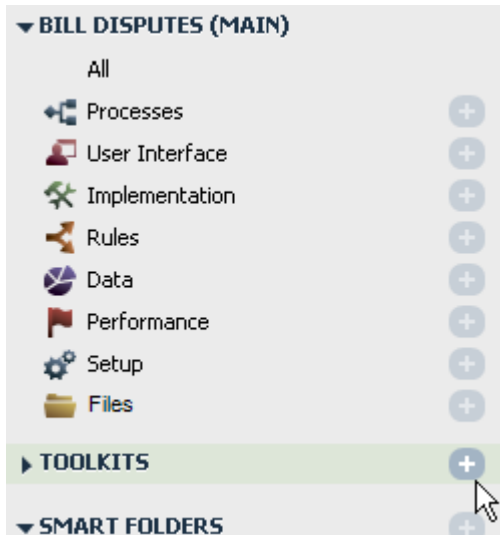
To create a dependency on a toolkit, one or more snapshots of that toolkit must exist. If not, the library items within the toolkit are not available for re-use.

1. Make sure the process application or toolkit for which you want to create a toolkit dependency is open in the Designer view. You can check the currently open process application or toolkit as shown in the following image:



If workspaces are enabled, the workspace name is displayed in parentheses after the process application or toolkit. In the preceding example, the currently open process application is Bill Disputes and the open workspace is Main.

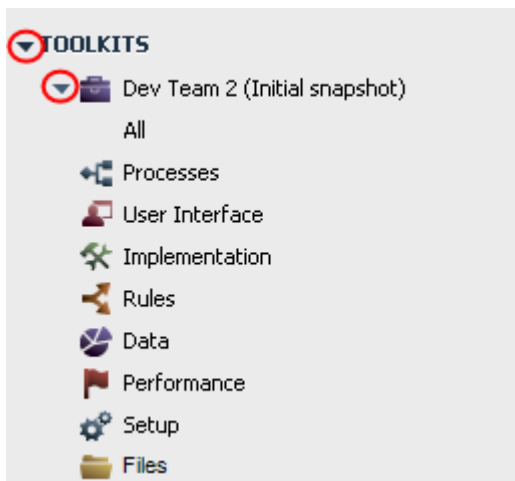
2. Click the plus sign next to **Toolkits** in the library as shown in the following image:



3. From the **Add Dependency** dialog, click to select the snapshot of the toolkit that you want.

You should choose the snapshot that includes the version of the library items that you need to re-use in your current project.

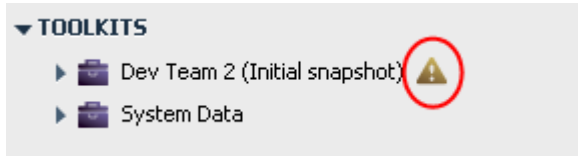
4. Now when you expand the Toolkits entry in the library, you should see the toolkit snapshot that you added. You can then click to expand the toolkit to see the library items within it as shown in the following image:



Double-clicking an item category, such as **Processes**, shows the processes included. The library items that you see are automatically made available for re-use throughout your current project.

Updating a toolkit dependency in the Designer view

1. If you create a toolkit dependency and that toolkit is subsequently updated, the Designer view displays the icon shown in the following image, letting you know that a more recent snapshot of the toolkit is available:



2. Click the icon and select one of the following options:

Upgrade dependency to <i>current_snapshot_name</i>	Select this option to ensure that you have the most recent version of the library items within the toolkit.
Ignore this new version of dependency	Select this option if you know you do not want to use the updated versions of the library items in this toolkit.

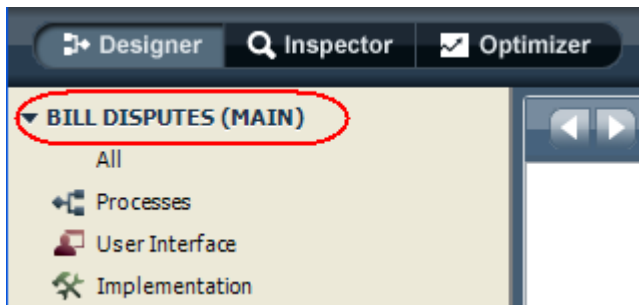


You can change the version of a toolkit dependency at any time by right-clicking the toolkit in the library and choosing **Change version of dependency** from the pop-up menu. In the Change Dependency dialog, select the snapshot that you want.

When you update or change the version of a toolkit dependency, library items from the toolkit that are currently in use are automatically updated. For example, if a service from the toolkit is the implementation for an activity and that service changes from one snapshot to the next, the changes are automatically reflected in your implementation. The same is true if you change a dependency to an older version of a toolkit.

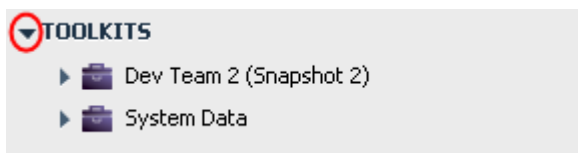
Deleting a toolkit dependency in the Designer view

1. Make sure the process application or toolkit that contains the dependency you want to remove is open in the Designer view. You can check the currently open process application or toolkit as shown in the following image:



If workspaces are enabled, the workspace name is displayed in parentheses after the process application or toolkit. In the preceding example, the currently open process application is Bill Disputes and the open workspace is Main.

2. Expand the Toolkits entry in the library:



3. Right-click the toolkit in the library and choose **Remove dependency** from the pop-up menu.



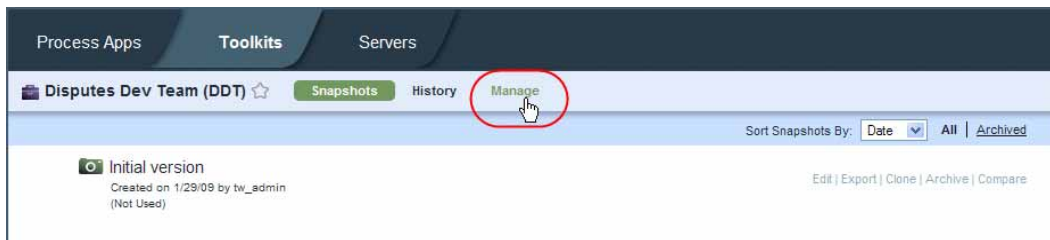
When you delete a toolkit dependency, you must be sure to update implementations of library items from the toolkit. For example, if a service from the toolkit is the implementation for an activity, the implementation for that activity is missing or broken as soon as you remove the toolkit dependency. Missing implementations are marked with alert icons in the properties for an affected activity.

Archiving toolkits in the Process Center Console



If you archive a toolkit and that toolkit has existing dependencies in the Designer, those dependencies remain in tact, including implementations that rely on library items within the toolkit. However, you should not use archived toolkits when creating toolkit dependencies in the Designer view since archived items are not considered part of the active Lombardi library.

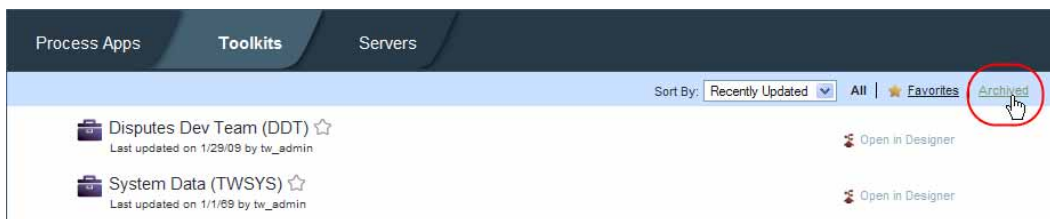
1. Select the **Toolkits** tab.
2. In the list of toolkits, click on the toolkit that you want to archive.
3. Click the **Manage** option shown in the following image:



4. Click the **Archive Toolkit** option as shown in the following image:



5. When prompted, click the **Archive** button to confirm that you want to archive this toolkit.
6. To view or restore archived toolkits, click the **Archived** filter in the Toolkits tab as shown in the following image:

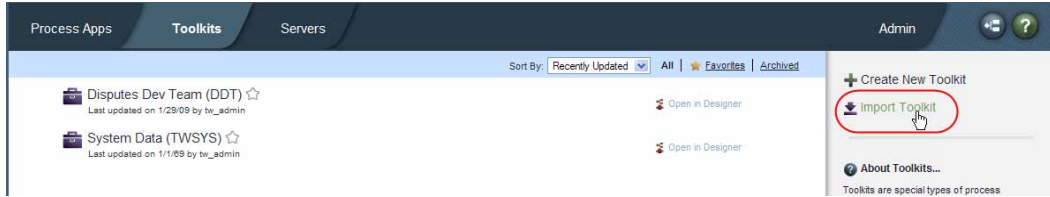


Importing and exporting toolkits from the Process Center Console

You can import toolkits from other Lombardi libraries and you can also export toolkits from the Process Center Console.

To import toolkits:

1. Select the **Toolkits** tab.
2. Click the **Import Toolkit** option shown in the following image:



3. In the Import Toolkit dialog, click the **Browse** button to locate the Lombardi export (.twx) file that you want to import.



Toolkits that you import should have unique acronyms. If an acronym is not unique, import completes with a warning. However, attempts to install snapshots that include references to a toolkit with a non-unique acronym will fail with an error.

4. Click the **Next** button to import the selected .twx file.

In the Import Toolkit dialog, click to expand the sections that show the snapshots that will be imported and the snapshots that are already available (and will not be imported).

5. Click the **Import** button.

When the import is complete, the imported toolkit is included in the list in the Toolkits tab. You can grant access to other users as described in [Managing access to process applications and toolkits](#).

Repository administrators can see a log of all imports by clicking the **Admin** option at the top-right of the Process Center Console and then clicking **Import and Export Log**.



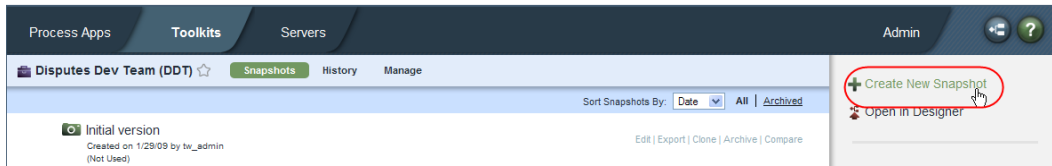
Imported toolkits are immutable, which means that no one can change the items within an imported toolkit. The user who imports a toolkit has administrative access to that toolkit. The following table describes the type of access that the toolkit administrator can grant to other users and groups.

Read	Users with Read access can clone the imported toolkit or copy items from the toolkit to a different toolkit or process application.
Write	Users with Write access have all the capabilities included with Read access plus they can import a new version of the toolkit and archive older versions of the toolkit. Write access also enables users to restore previously archived versions.
Admin	Users with Admin access have all the capabilities included with Write access plus they can grant or remove administrative access to the toolkit.

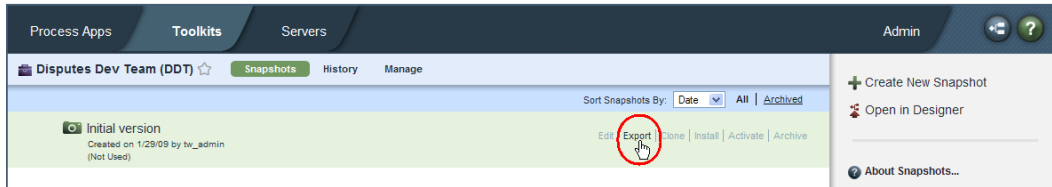
To export toolkits:

1. Select the **Toolkits** tab.
2. In the list of toolkits, click on the toolkit that you want to export.
3. Find the snapshot that you want to export.

If a snapshot does not exist, create one by clicking **Create New Snapshot** as shown in the following image:



4. Click the **Export** option for the snapshot as shown in the following image:



5. Locate the directory to which you want to save the export (.twx) file, name the file, and then save it.

The exported file can be imported into any Process Center repository.

Repository administrators can see a log of all exports by clicking the **Admin** option at the top-right of the Process Center Console and then clicking **Import and Export Log**.

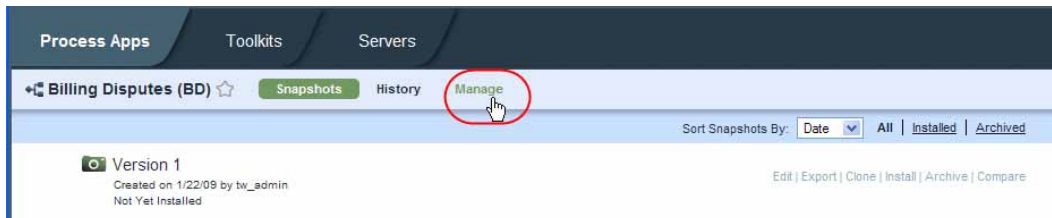
Managing workspaces

You can enable and manage workspaces for the process applications and toolkits that you create or to which you have administrative access. To learn how to manage user accounts and authorizations, see [Managing access to the Process Center repository](#).

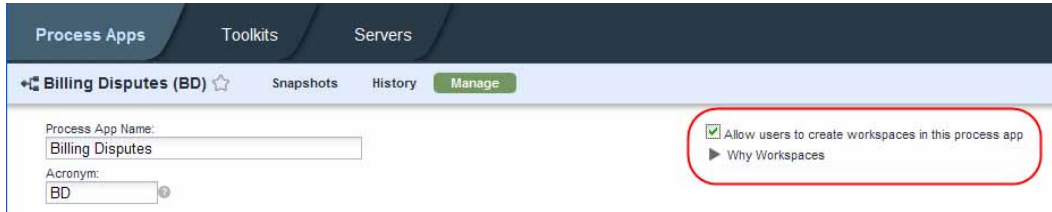
Before performing any of the following tasks, start Lombardi Authoring Environment and access the Process Center Console as explained in [Starting Lombardi Authoring Environment](#).

Enabling workspaces in the Process Center Console

1. Select the **Process Apps** or **Toolkits** tab.
2. Select the process application or toolkit for which you want to enable workspaces.
3. Click the **Manage** option shown in the following image:



4. Click the **Allow users to create workspaces in this process app** check box as shown in the following image:



For toolkits, the check box label is **Allow users to create workspaces in this toolkit**.

Administrators can now create additional workspaces in the selected process application or toolkit as described in the following procedure.

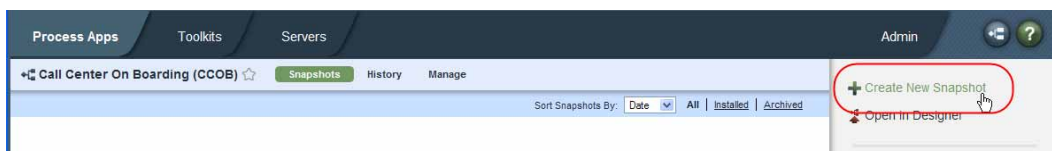
Creating new workspaces in the Process Center Console

When you create a process application as described in [Managing process applications](#) or a toolkit as described in [Managing and using toolkits](#), Lombardi creates a single default workspace named Main. After workspaces are enabled for a process application or toolkit as described in the preceding task, you can create additional workspaces.



To create a new workspace, you must enable workspaces as instructed in the preceding task and you must use a snapshot as the basis for a new workspace.

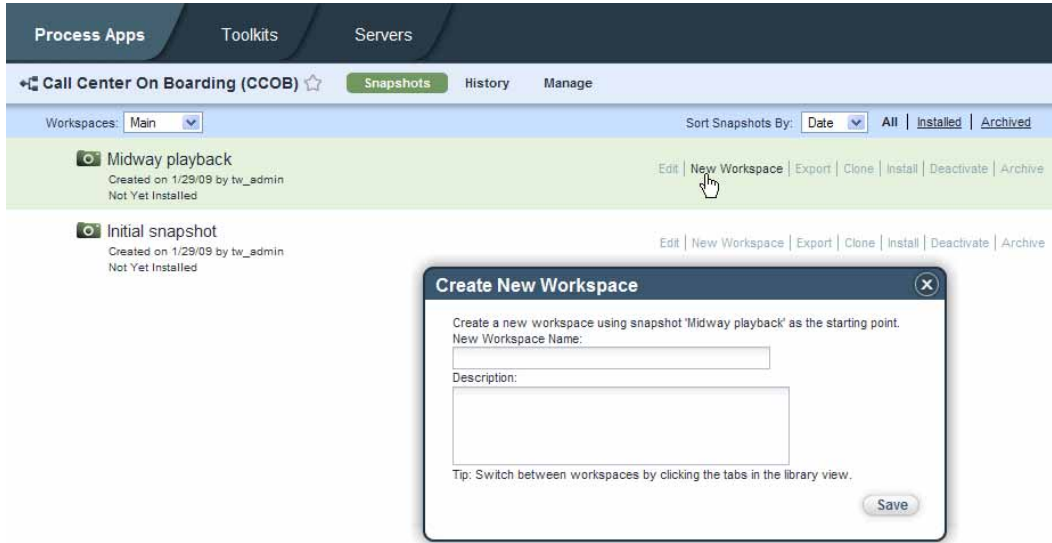
1. Select the **Process Apps** or **Toolkits** tab.
2. Select the process application or toolkit for which you want to create a new workspace.
3. If a snapshot does not exist, create one by clicking **Create New Snapshot** as shown in the following image:



4. Click the **New Workspace** option for the snapshot as shown in the following image:



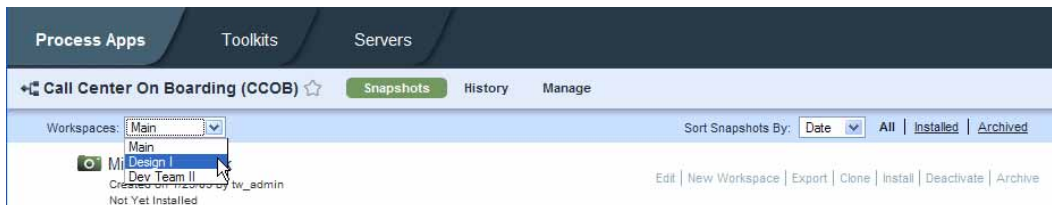
The New Workspace option is available only if you have enabled workspaces for the process application or toolkit.



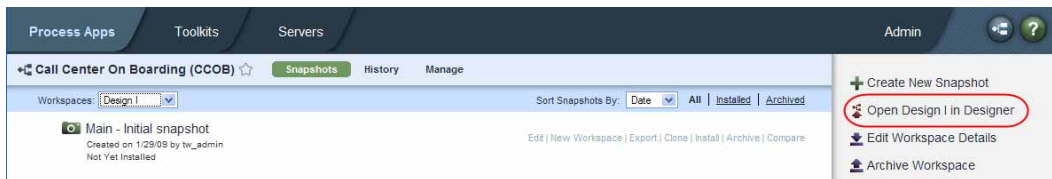
5. Enter a name and click **Save**. The description is optional.

When you create a new workspace, the Process Center Console displays a drop-down menu for the process application or toolkit that enables you to select the workspace that you want.

6. To access the new workspace, click the Workspaces drop-down menu and select the workspace that you want as shown in the following image:

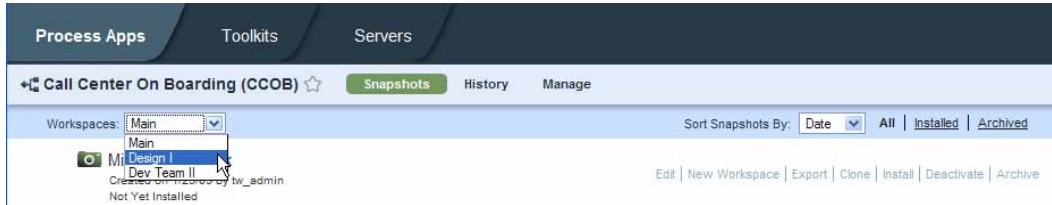


7. To open the new workspace in the Designer in Lombardi Authoring Environment, click the option highlighted in the following image:

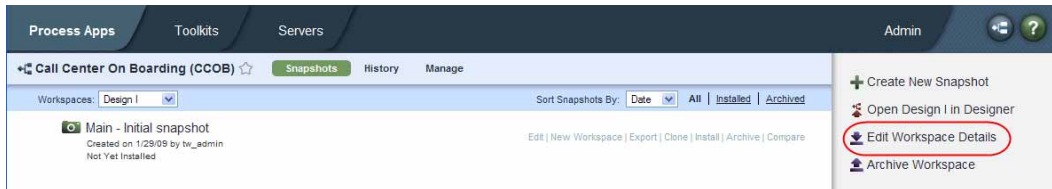


Editing workspaces in the Process Center Console

1. Select the **Process Apps** or **Toolkits** tab.
2. Select the process application or toolkit in which the workspace that you want to edit resides.
3. Click the Workspaces drop-down menu and select the workspace that you want as shown in the following image:



4. Click the **Edit Workspace Details** option as shown in the following image:

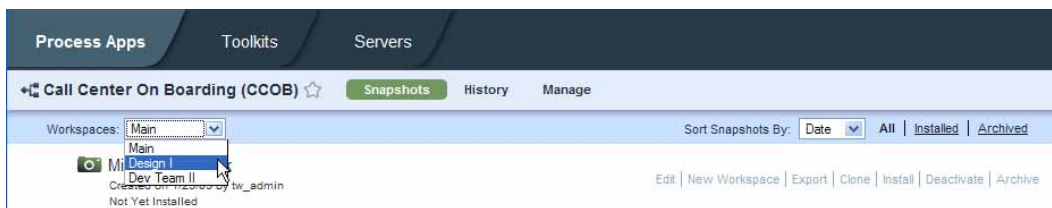


5. In the Workspace Details dialog, edit the workspace name and description and click **Save**.

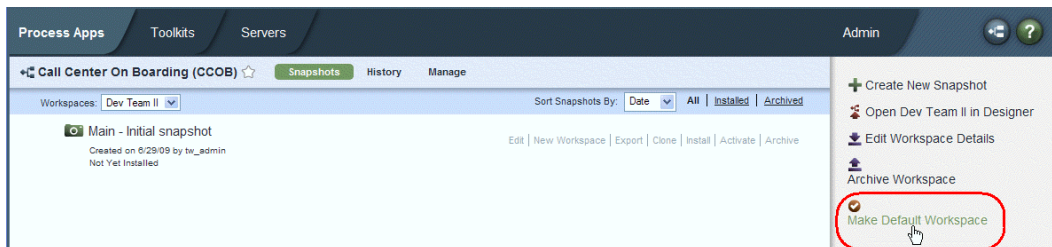
Setting the default workspace in the Process Center Console

When you create a process application as described in [Managing process applications](#) or a toolkit as described in [Managing and using toolkits](#), Lombardi creates a single default workspace named Main. If you create additional workspaces, you can configure one of the new workspaces as the default workspace. When a workspace is the default workspace, the library items within it run by default when an event or other trigger that applies to items in more than workspace is received by the Process Center Server. For example, when you are executing a service by URL and that service exists in more than one workspace, the service in the default workspace is executed.

1. Select the **Process Apps** or **Toolkits** tab.
2. Click to select the process application or toolkit that contains the workspace that you want.
3. Click the Workspaces drop-down menu and select the workspace that you want as shown in the following image:



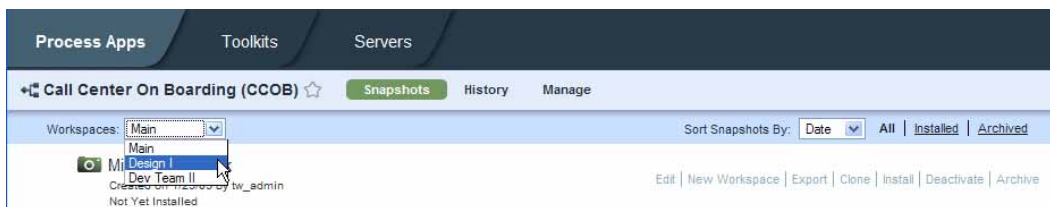
4. Select the **Make Default Workspace** option as shown in the following image:



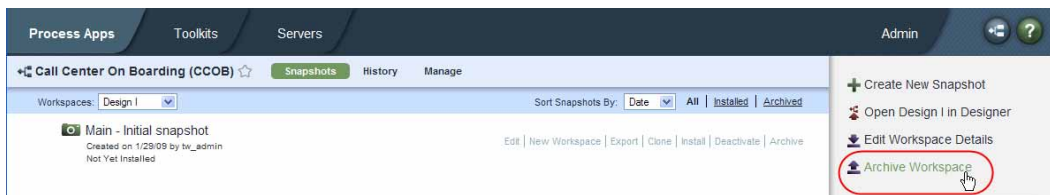
Archiving workspaces in the Process Center Console

If a workspace is no longer used, you can archive it. When you archive a workspace, the option for that workspace no longer appears in the Workspaces drop-down menu for the process application or toolkit. If only two workspaces exist for a process application or toolkit and you archive one of them, the Workspaces drop-down menu is no longer displayed. You must restore an archived workspace before you can open it in the Designer view in Lombardi Authoring Environment.

1. Select the **Process Apps** or **Toolkits** tab.
2. Select the process application or toolkit in which the workspace that you want to archive resides.
3. Click the Workspaces drop-down menu and select the workspace that you want as shown in the following image:



4. Select the **Archive Workspace** option as shown in the following image:



5. When prompted, click the **Archive** button to confirm that you want to archive this workspace.
6. To view or restore archived workspaces, click the **Manage** option for your process application or toolkit and then click the **View Archived Workspaces** option shown in the following image:



Managing snapshots

Snapshots record the state of library items within a process application or workspace at a specific point in time. You can create snapshots in the Process Center Console or in the Designer view. Snapshot management, such as deploying, exporting, and archiving, is performed in the Process Center Console.

In addition to the topics covered in this section, you can refer to the following topics for more information about managing snapshots:

To learn how to...	See...
Install snapshots of process applications	Installing process applications: online Process Servers

To learn how to...	See...
Import and export snapshots of process applications	Importing and exporting process applications from the Process Center Console
Import and export snapshots of toolkits	Importing and exporting toolkits from the Process Center Console
Archive snapshots of process applications	Archiving process applications in the Process Center Console
Archive snapshots of toolkits	Archiving toolkits in the Process Center Console

You can create snapshots of the process applications and toolkits that you have created or to which you have write or administrative access. See [Managing access to the Process Center repository](#) for more information.

Before performing any of the following tasks, start Lombardi Authoring Environment and open the appropriate view as explained in [Starting Lombardi Authoring Environment](#).



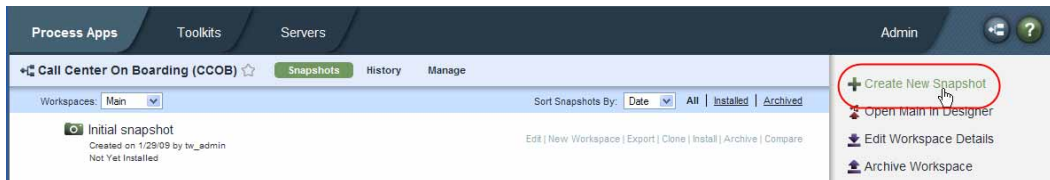
Lombardi does not require unique snapshot names. However, you cannot install a snapshot of a process application in a runtime environment if a snapshot with the same name has already been installed. This is also true for snapshots in different workspaces.

Creating new snapshots in the Process Center Console

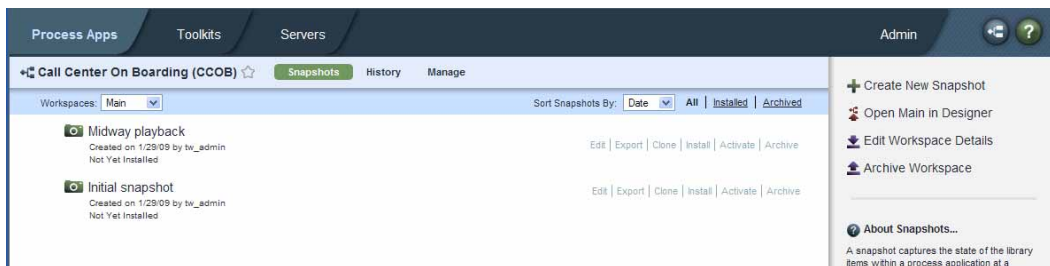
1. Select the **Process Apps** or **Toolkits** tab.
2. Select the process application or toolkit for which you want to create a snapshot.
3. Click the **Create New Snapshot** option as shown in the following image.



If multiple workspaces exist, select the workspace that you want from the drop-down menu and then click the **Create New Snapshot** option.



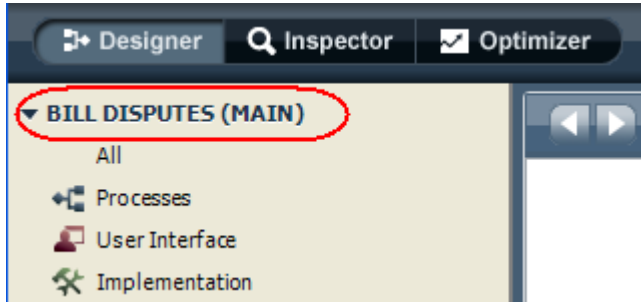
4. Enter a name for the snapshot and click **Save**. The description is optional.
5. The resulting snapshot is displayed in the Process Center Console:



The snapshot records the current state of the library items within the workspace. You can install the snapshot as described in [Releasing and installing processes](#) and you can also compare and archive snapshots as explained in the following tasks.

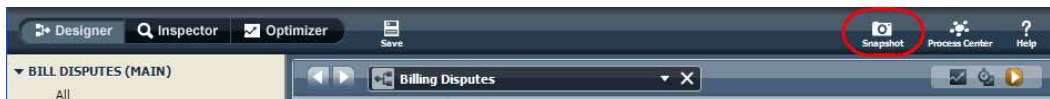
Creating new snapshots in the Designer view

1. Make sure the process application or toolkit for which you want to create a snapshot is open in the Designer view. You can check the currently open process application or toolkit as shown in the following image:



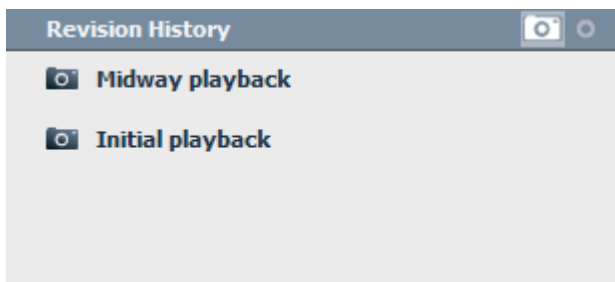
If workspaces are enabled, the workspace name is displayed in parentheses after the process application or toolkit. In the preceding example, the currently open process application is Billing Disputes and the open workspace is Main.

2. Click the snapshot icon highlighted in the following image:



3. Enter a name for the snapshot and click **OK**.

The Designer displays the newly created snapshot, named Midway Playback, in the Revision History tab as shown in the following image:

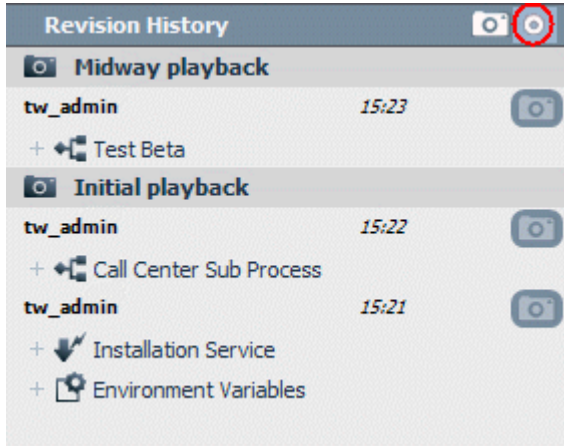


To learn how to compare snapshots in the Designer view, see the following procedure.

To learn how to create snapshots from previous points in time, see [Creating snapshots from the revision history in the Designer view](#)

Comparing snapshots in the Designer view

1. To view new or changed library items in each snapshot, click the icon shown in the following image:

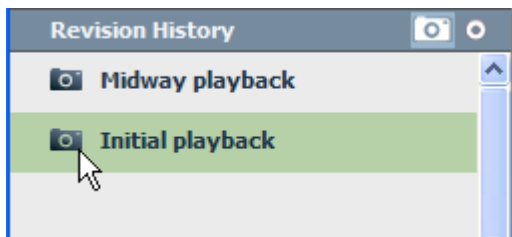


In the preceding example, one new process called Test Beta was added to the library for the Midway playback snapshot. This is the only change or addition made since the original snapshot, Initial playback.

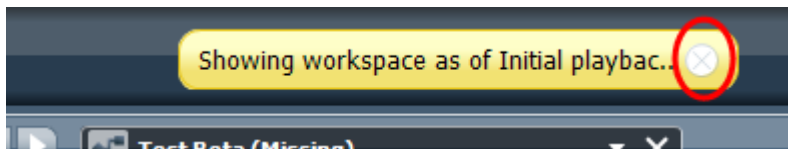


To return to the list of snapshots, click the snapshot icon in the Revision History header.

2. To view the state of an entire process application or toolkit as of a particular snapshot, click the indicator shown in the following image:



The Designer displays a message in the toolbar to indicate that it is showing the contents of the entire process application as they existed when the snapshot called Initial playback was created. While this message is displayed, you can examine the library contents, but you cannot make changes. To get back to the current state of your process application or toolkit, click the indicator shown in the following image:



When you are back in the current state, you can change and add library items.

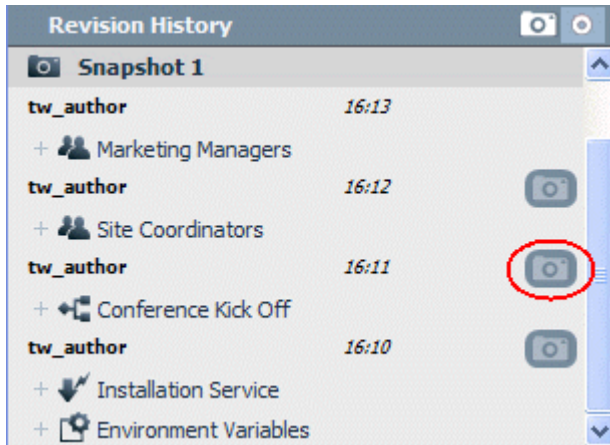
Creating snapshots from the revision history in the Designer view

In addition to capturing snapshots of your ongoing efforts in the Designer view, you can also create snapshots from previous points in time using the entries in the revision history. For example, if you need to snapshot your project as it existed before several new items were added, you can use the revision history to locate the point in time that meets your needs.

1. To view the detailed revision history, click the icon shown in the following image:



2. Find the point in time at which you want to create a snapshot and click the corresponding snapshot icon in the Revision History as shown in the following example:



3. In the Take Snapshot dialog box, type a name for the snapshot and click **OK**.

The revision history displays the new snapshot, which you can install on a test or production server, export to a different Process Center, or use in other ways as described in [Managing snapshots](#).

Activating snapshots in the Process Center Console

If you want exposed library items within particular snapshots to display in Lombardi Process Portal while those items are being developed and reside on the Process Center Server, you need to activate the snapshot that contains the version of the items that you want to display. For example, if you are developing a BPD and you want to start the BPD using the New icon in the Process Portal Inbox, you need to activate the snapshot that contains the version of the BPD that you want to start. Doing so enables you to start and run the BPD on Lombardi Process Portal for testing and other purposes. (For more information about exposing BPDs, see [Configuring BPDs](#). For information about exposing Human services, see [Exposing a Human service](#).)

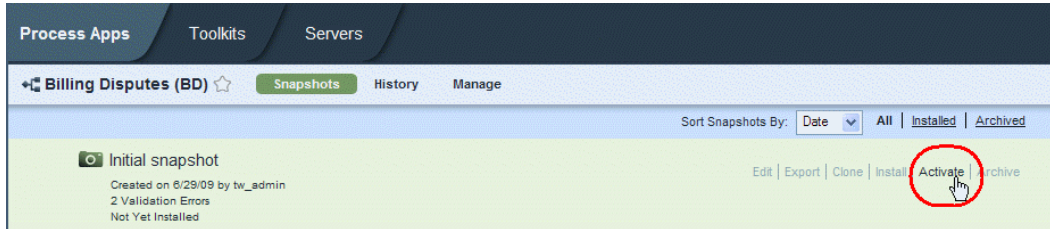


Exposed BPDs and data from the current working version (tip) are always available. Activation is required only when you want to access a snapshot version of an item or data that resides on the Process Center Server.

When you install snapshots of process applications on Process Servers in other environments, such as test and production environments, those snapshots are active by default. You can deactivate installed snapshots and perform other configuration tasks as described in [Configuring installed snapshots](#).

To activate or deactivate a snapshot, you must be a repository administrator, an administrator for the selected process application or toolkit, or the user who created the process application or toolkit.

1. In the Process Center Console, select the **Process Apps** or **Toolkits** tab.
2. Click to select the process application or toolkit for which you want to activate a snapshot.
If multiple workspaces exist, select the workspace that you want from the drop-down menu.
3. Find the snapshot that you want and click **Activate** as shown in the following image:



Now when you start Lombardi Process Portal by directing your browser to `http://[host_name]:[port]/portal`, providing the name of the host on which Lombardi Process Center Server is installed and the port designated for the server during installation, you can access the exposed library items in the activated snapshot. Using the previous example, you can start the exposed BPD using the New icon in the Process Portal Inbox.



To deactivate a snapshot, follow the preceding steps and click the Deactivate option.

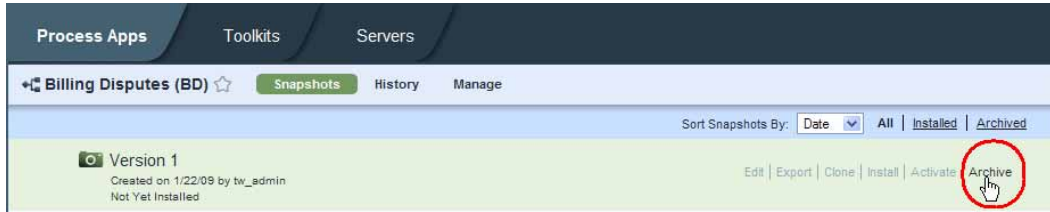
Archiving snapshots in the Process Center Console

If a snapshot is no longer used, you can archive it. When you archive a snapshot, it no longer appears in the list of snapshots for the process application or toolkit in the Process Center Console. You must restore a snapshot if you want to edit it or perform any other actions on it.

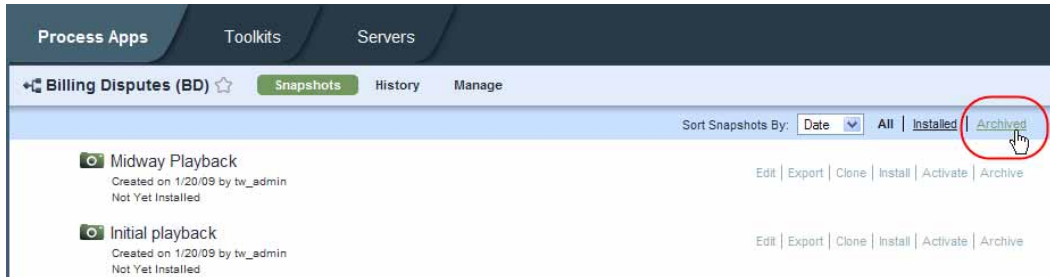


To archive a snapshot, you must have write or administrative access to the process application. For more information, see [Managing access to process applications and toolkits](#).

1. Select the **Process Apps** or **Toolkits** tab.
2. Select the process application or toolkit for which you want to archive snapshots.
If multiple workspaces exist, select the workspace that you want from the drop-down menu.
3. Find the snapshot that you want and click **Archive** as shown in the following image:



4. When prompted, click the **Archive** button to confirm your selection.
5. To view or restore archived snapshots, click the **Archived** filter as shown in the following image:



Managing access to the Process Center repository

Administrators manage user access to the Lombardi repository from the Process Center Console.

The Process Center Console displays users and groups from the following sources:

- Lombardi internal security provider. To learn how to create and manage internal Lombardi users, see the *Lombardi Administration Guide*.
- Any external security provider that was configured for use with Lombardi during installation. For more information about configuring your external security provider, see the *Lombardi Installation and Configuration Guide* for your environment.

The best way to manage access to the Process Center repository, and Lombardi in general, is by using groups. For example, the easiest way to manage access to the Process Center repository is to add preexisting groups of users from your external provider to `tw_authors`, which is a Lombardi group whose members have access to the repository by default. Then when changes are required, you can simply add or remove individual users from the groups that exist in your external security provider. This practice ensures that the security maintenance you perform in your external provider does not require additional work in Lombardi.

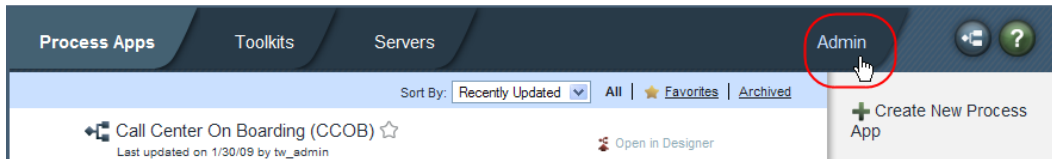
The same is true for administrative access to the Process Center repository. You can simply add preexisting groups of users from your external provider to `tw_admins`, which is a Lombardi group whose members have administrative access to the repository by default. See the *Lombardi Administration Guide* for more information about adding groups from your external security provider to Lombardi groups.

Before performing any of the following tasks, start Lombardi Authoring Environment and open the Process Center Console as explained in [Starting Lombardi Authoring Environment](#).

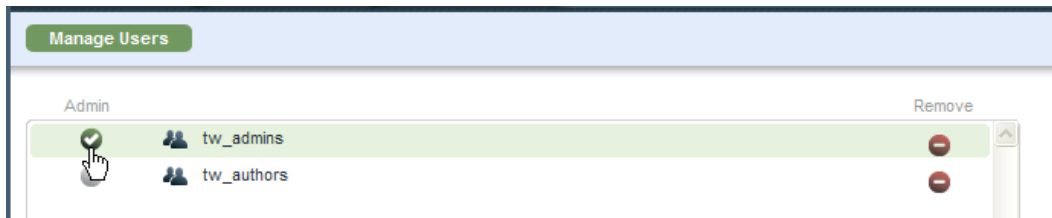
Granting administrative access to the Process Center repository

By default, Lombardi includes the tw_admin user account which provides administrative access to the Process Center repository. This default administrator can grant administrative access to other users by following these steps:

1. In the Process Center Console, click the **Admin** option as shown in the following image:



2. Select the **Manage Users** option.
3. Find the user or group to which you want to grant administrative access and then enable the **Admin** option for that user or group as shown in the following example:



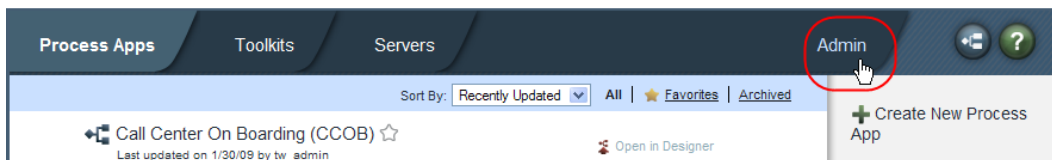
Groups and users who are designated as **Admin** in this dialog can manage user access to the entire Process Center repository as outlined in the following procedure.

Adding users and groups

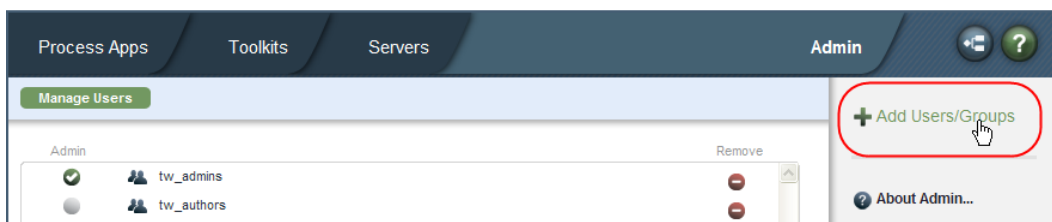


To manage user access to the Process Center repository, you must have administrative access to the repository. (Administrative access is granted as described in [Granting administrative access to the Process Center repository](#).)

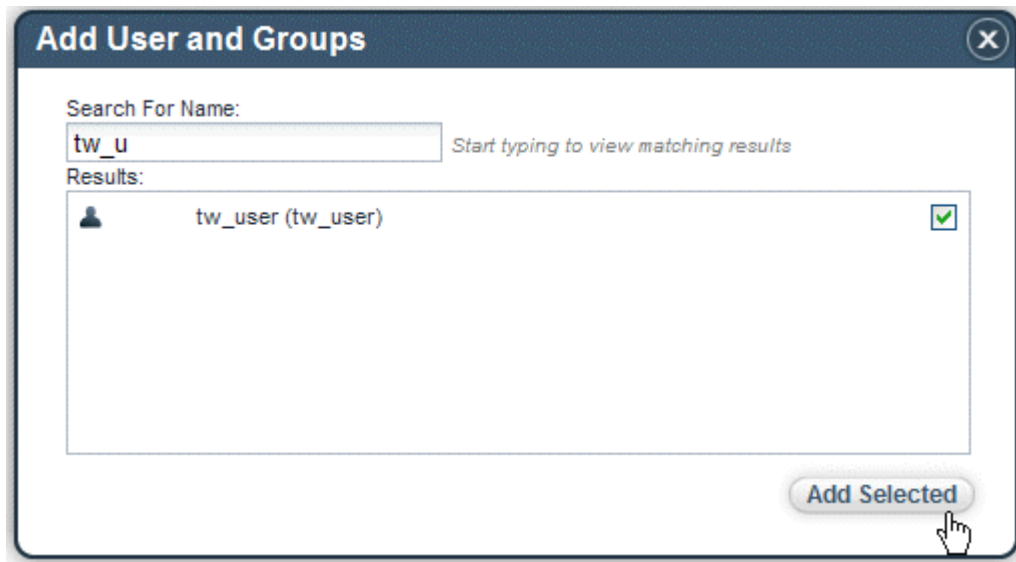
1. In the Process Center Console, click the **Admin** option as shown in the following image:



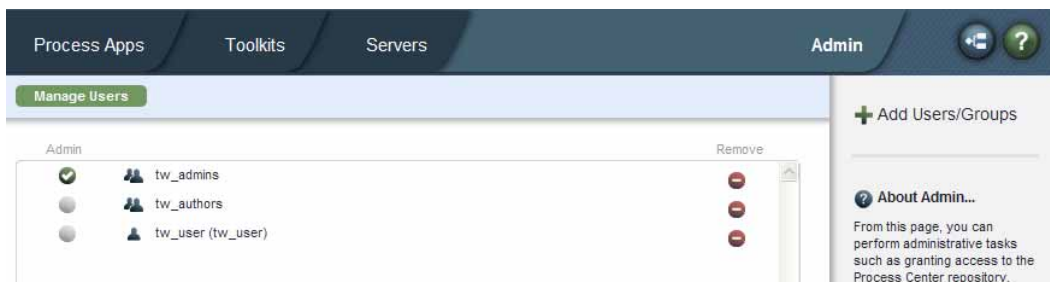
2. Select the **Manage Users** option and then click **Add Users/Groups** as shown in the following image:



3. In the **Add Users and Groups** dialog, enter the name of the user or group that you want to add in the **Search for Name** field. You can enter part of the name and Lombardi displays all accounts that match as shown in the following example.
4. Click the check box next to the users and groups that you want to add and click the **Add Selected** button as shown in the following example.



The added users and groups shown in the following image now have access to Lombardi Authoring Environment and Process Center repository, which gives them the ability to create new process applications and toolkits. When you create a process application or toolkit, you have the ability to grant access to other users. Administrators can also grant access to specific process applications and toolkits as outlined in the following procedure.



Groups and users who are designated as **Admin** in this dialog are able to manage user access to the entire Process Center repository as outlined in the preceding steps.

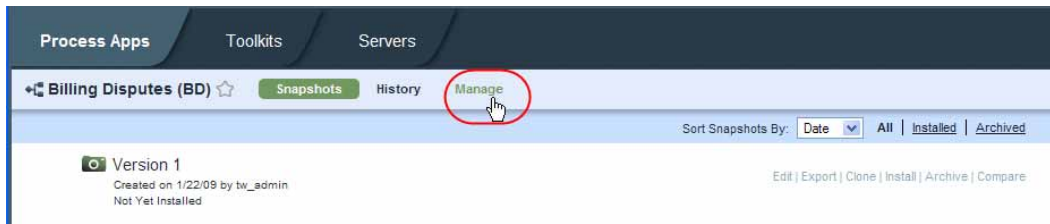
Managing access to process applications and toolkits



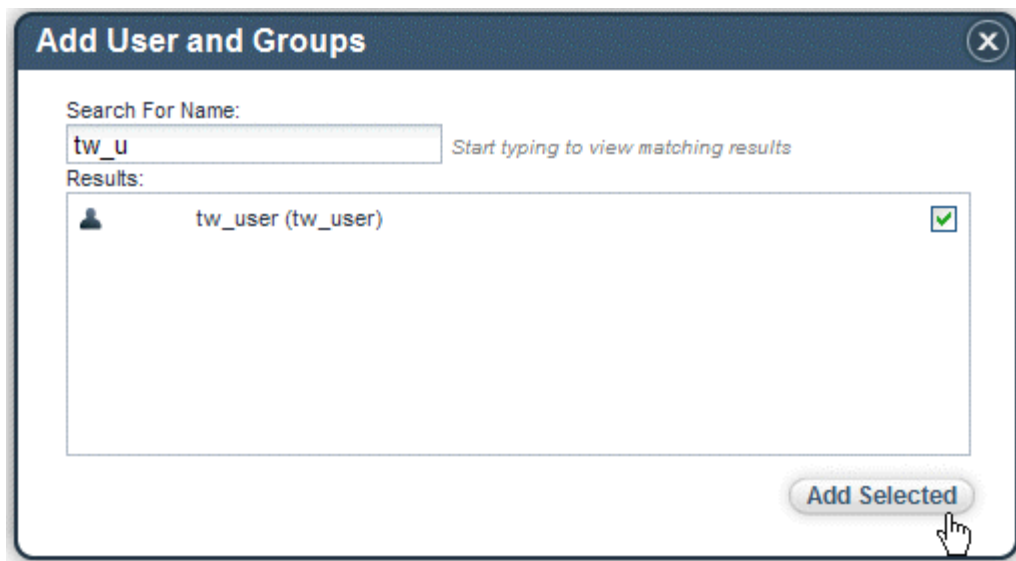
To manage access to individual process applications and toolkits, you must have administrative access to the process application or toolkit, or you must be the user who created the process application or toolkit.

1. Select the **Process Apps** or **Toolkits** tab.

2. Select the process application or toolkit for which you want to manage user access.
3. Select the **Manage** option shown in the following image:



4. Under **Manage Access to Process Library**, click the **Add Users/Groups** button.
5. In the **Add Users and Groups** dialog, enter the name of the user or group that you want to add in the **Search for Name** field. You can enter part of the name and Lombardi displays all accounts that match as shown in the following example.
6. Click the check box next to the users and groups that you want to add and click the **Add Selected** button as shown in the following example.



7. After adding the users and groups that you want, select the **Read**, **Write**, or **Admin** option for each.
The options grant access to the process application or toolkit as follows:

Read	Users with Read access can view the process application or toolkit in the Process Center Console, as well as view all library items included in the process application or toolkit in the Designer view. However, with Read access, edits are not allowed.
Write	Users with Write access can view the process application or toolkit in the Process Center Console. Plus, they can create, edit, or delete library items within the process application or toolkit in the Designer view. Users with Write access can also create and edit snapshots of the process application or toolkit either in the Process Center Console or Designer view.
Admin	Users with Admin access have all the capabilities included with Write access plus the ability to perform the following actions in the Process Center Console: edit process application or toolkit settings; create, edit, or archive workspaces; archive snapshots; and modify user access to the process application or toolkit.



To add options for a user, click the option to enable or disable it. To remove a user or group so that they no longer have access to the process application or toolkit, click the remove icon for the user or group.

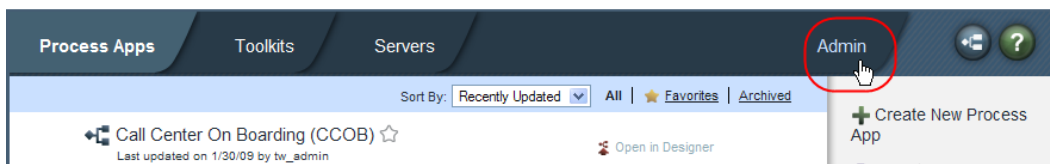
Removing users and groups

The following procedure describes how to remove users and groups so that they no longer have access to the Process Center repository.

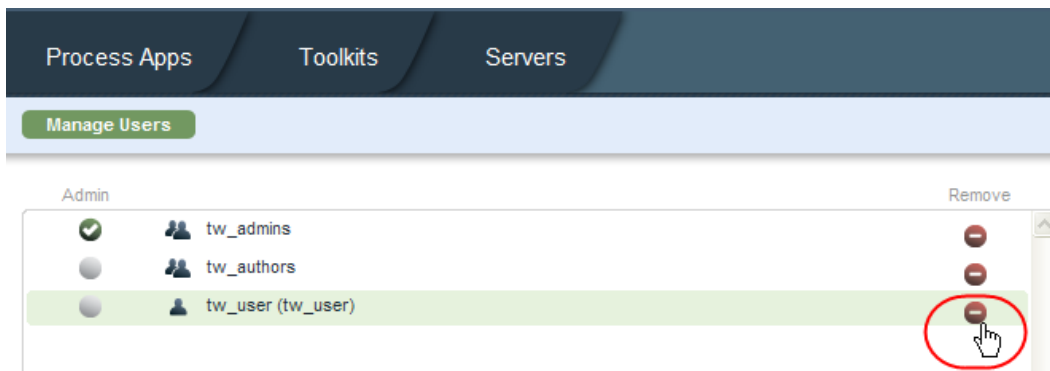


To manage user access to the Process Center repository, you must have administrative access to the repository.

1. In the Process Center Console, click the **Admin** option as shown in the following image:



2. Select the **Manage Users** option.
3. Click the remove icon for each user or group that you want to deny access as shown in the following image:



Users and groups who are removed here are automatically removed from any process applications or toolkits to which they have been granted access.

Managing Lombardi servers

From the Process Center Console, repository administrators can manage the Lombardi servers in their environment. Process Servers configured during Lombardi installation are automatically discovered and displayed in the Process Center Console. Any offline servers that have been added are also displayed in the Process Center Console.

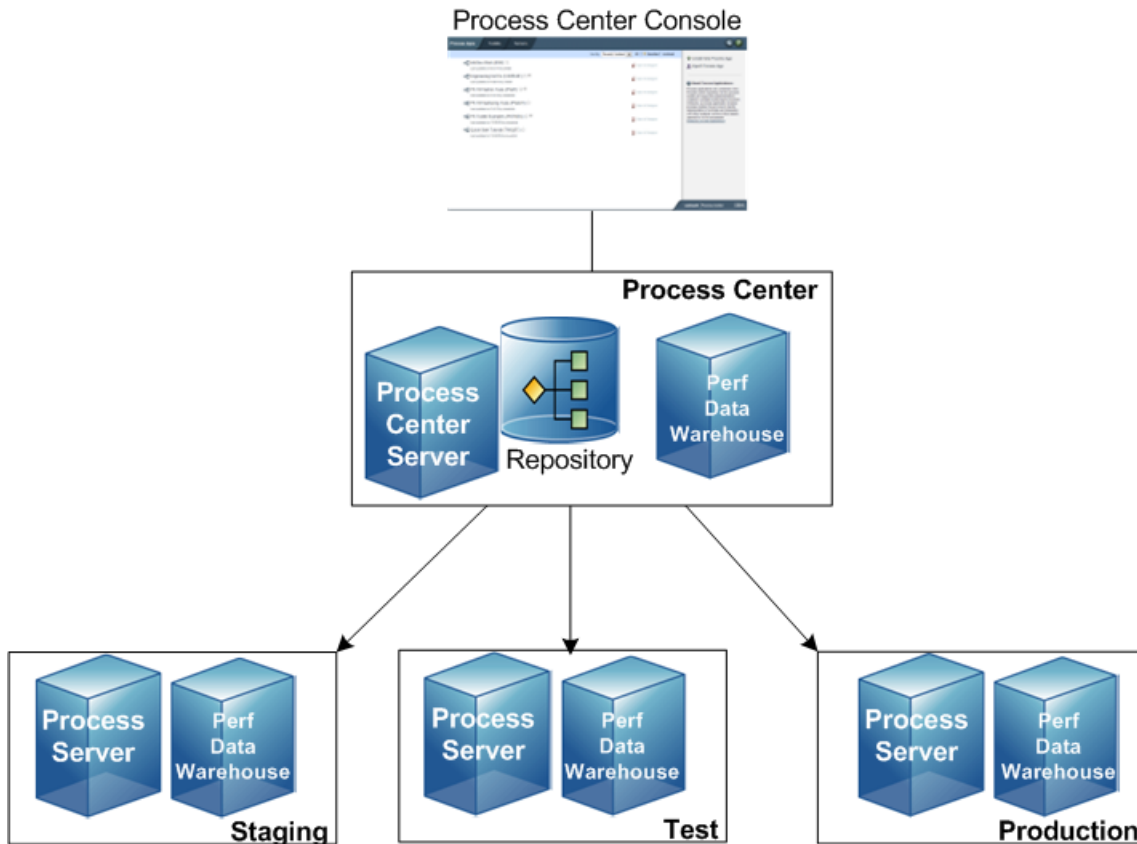
From the Process Center Console, repository administrators can monitor the snapshots of process applications installed on each connected Process Server in the Lombardi environment. For example,

administrators can monitor all snapshots installed on connected test and production servers. You can also open the Process Admin Console from the Process Center Console, which enables you to complete configuration and administrative tasks for each connected Process Server.



Use the Performance Admin Console to manage the Performance Data Warehouses in your environment. You can start the Performance Admin Console using the Windows start menu or desktop shortcuts, or you can open your favorite Web browser to the following location:
[http:// \[host_name\] : \[port\] /PerformanceAdmin](http://[host_name]:[port]/PerformanceAdmin). See the *Lombardi Administration Guide* for more information about the Performance Admin Console.

The following figure shows a sample server configuration for Lombardi where all Process Servers are connected to the Process Center:



Before performing any of the following tasks, start Lombardi Authoring Environment and open the Process Center Console as explained in [Starting Lombardi Authoring Environment](#).

Monitoring installed snapshots on each Process Server from the Process Center Console



Only users who have administrative access to the Process Center repository can monitor all installed snapshots. Other users can monitor installed snapshots of process applications to which they have access.

1. Click the Servers tab in the Process Center Console and select the Process Server that you want to monitor as shown in the following example.



If you select a Process Server that is connected to the Process Center, you can see the installed snapshots and the number of instances (for exposed BPDs) currently running on the server. If you select an offline Process Server that has been added (as described in [Adding offline servers to the Process Center Console](#)), you can see the snapshots for which an installation package has been created. For more information about installing process applications to both type of servers, see [Releasing and installing processes](#).

The snapshots are organized per process application. For example, in the following image, two different snapshots of the HR Onboarding process application have been installed on the selected Process Server:



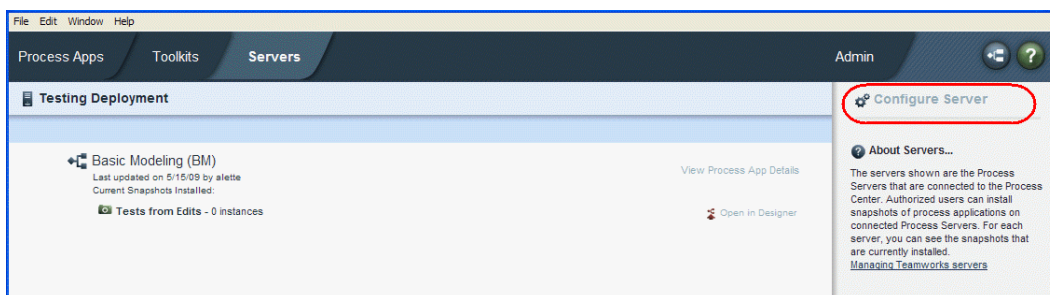
- For each snapshot, you can click the **Open in Designer** option to open the selected snapshot of the process application in the Designer view. Doing so enables you to see the state of each library item as it existed when it was installed.

Configuring Lombardi Process Servers from the Process Center Console



To configure all options for Lombardi Process Servers, you must be a Lombardi administrator. Other users can configure the parts of the server that affect the implementations that they develop in the Designer in Lombardi Authoring Environment. For example, members of the `tw_authors` group can configure the Event Manager. See the *Lombardi Administration Guide* or the online help for the Process Admin Console for more information.

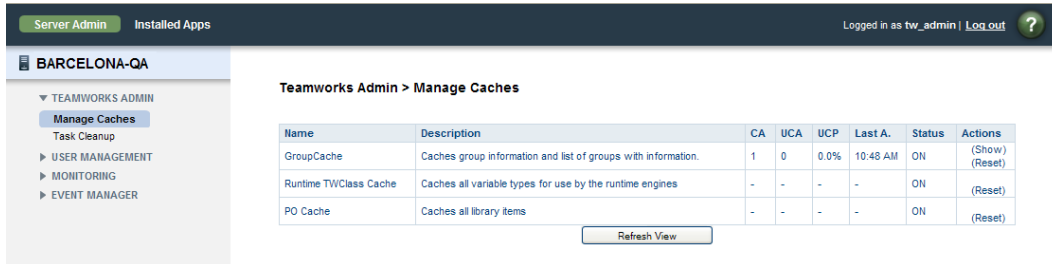
- Click the Servers tab in the Process Center Console and then select the Process Server that you want to configure.
- Click the **Configure** option shown in the following image:





The configure option is not available for offline servers.

- For the selected server, you can perform the administrative options listed in the Server Admin tab:



You can find step-by-step Instructions for these configuration tasks in the *Lombardi Administration Guide* or the online help for the Process Admin Console.

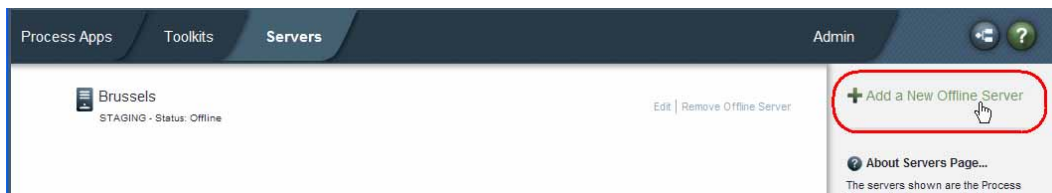
Adding offline servers to the Process Center Console

Lombardi enables you to install snapshots of process applications to Process Servers that are not connected to the Process Center Server. To do so, you must first add an offline server as described in the following procedure.



You must have administrative access to the Process Center repository to add, edit, or remove offline servers. See [Granting administrative access to the Process Center repository](#) for more information.

- Click the Servers tab in the Process Center Console.
- Click the **Add a New Offline Server** option shown in the following image:



- In the Create New Server dialog, name the sever, choose the server type from the drop-down list, provide an optional description, and then click the **Create** button.



Server names must be unique. Also, be sure to choose the appropriate server type so that the migration options made available during installation are appropriate. See [Installing process applications: offline Process Servers](#) for more information.

Lombardi displays the new server, as well as its type, in the Servers tab. You can change the name, server type, and description at any time by clicking the **Edit** option.

Removing offline servers from the Process Center Console

If an offline server has been removed from your Lombardi configuration, you can remove it from the Process Center Console as described in the following steps. You must have administrative access to the Process

Center repository to add or remove offline servers. See [Granting administrative access to the Process Center repository](#) for more information.



When you remove an offline server, all installation packages previously created for that server are also deleted. You cannot retrieve an offline server or the installation packages for that server after removing it.

1. Click the Servers tab in the Process Center Console.
2. Click the **Remove Offline Server** option.
3. When prompted, confirm that you want to remove the server.

Configuring installed snapshots

For each installed process application snapshot, you can either configure the entire snapshot (for example, select a snapshot to be the default version on the current server) or you can configure run-time settings (such as role bindings and environment variables) as described in the following tables.



To configure installed snapshots, you must have administrative access to the Process Server on which the snapshots are installed.

The options displayed in the right margin of the Process Admin Console enable you to configure the selected snapshot on the current server as follows:

Option	Description
Deactivate Application	Selected snapshot remains installed on the current server, but users cannot start new instances of the exposed processes or services. Any currently running instances run until complete.
Migrate In-flight Data	Migrates currently running instances to the version of the selected snapshot. Wherever the running instances are in the flow of the process or service, the new version is implemented for the next item or step.
Sync Settings	Copies settings from the selected snapshot. You can choose the settings that you want to copy such as environment variables, role bindings, exposed process values (EPVs), and so on.
Make Default Version	Makes the selected snapshot the default version on the current server. If a snapshot is the default, the items within it should run by default when an event or other trigger that applies to more than one version of a process or service is received. The first snapshot that you install on a server is the default version of the process application. When you install subsequent snapshots, you can use this option to ensure that the version that you want to run is actually the default.
Send Tracking Definitions	If a problem occurs during snapshot installation so that tracking definitions are not sent to the Performance Data Warehouse, you can use this option to send the definitions for the selected snapshot. Because tracking definitions are automatically sent to the Performance Data Warehouse during snapshot installation, you should only use this option when a problem occurs. For more information, see Troubleshooting installations .

Run-time configuration options include the following:

Option	Description	See...
Exposing	During development in Lombardi Authoring Environment, process authors determine which processes, services, reports, and other items are available and to which	Configuring exposed processes and services

Option	Description	See...
	<p>participant groups. After a process application is installed on a Process Server in a different environment (test or production), you may need to disable a particular exposed item within that application.</p>  <p>Items that are exposed are accessible to the designated group of users. For example, users in the designated group can start an exposed process in Lombardi Process Portal.</p>	
Role Bindings	<p>During development in Lombardi Authoring Environment, process authors create the participant groups for each process application. After a process application is installed on a Process Server in a different environment (test or production), you may need to add or remove users in those groups. For example, users that exist in the test environment may not have been available in the development environment. So, you would need to add those users once installation in your test environment is complete so that they can access and perform the tasks generated by the process.</p>	Configuring runtime participant groups
Environment Variables	<p>During development in Lombardi Authoring Environment, process authors can set environment variables for each process application. In some cases, the correct value for a particular environment (test or production) may not be known during process design. In those cases, you need to provide the value after installing the process application in the new environment.</p>	Configuring runtime environment variables

Configuring exposed processes and services

After selecting a snapshot to configure, you can manage exposed items by following these steps:

1. If not already selected, click the **Exposing** option.
2. Click the check box next to the item that you want to disable.

Clicking the check box causes the item to no longer be exposed to the selected group. When the exposure setting is disabled, the users within the group can no longer start or otherwise manipulate the process or service on the current server.



When you disable items that are not exposed to a particular participant group, such as Web services and Undercover Agents (UCAs), those items can no longer be executed on the current server.

Configuring runtime participant groups

After selecting a snapshot to configure, you can adjust the members of necessary participant groups. Follow these steps:

1. Click the **Role Bindings** option.

Lombardi lists each participant group and the members of each group. The participant groups listed are those that were created for the process application during process development in the Designer in Lombardi Authoring Environment.

- For each participant group listed, you can perform the following actions:

Action	Description
Click Add Users and Groups option	Lombardi displays the Add Users dialog where you can enter a partial or complete user name in the Retrieve text box to display the users and groups that are available on the current server. Select the check box for each user and group that you want and click the Add button.
Click the remove icon next to an existing user or group	Lombardi removes the user or group from the participant group.

Configuring runtime environment variables

After selecting a snapshot to configure, you can set environment variables to the appropriate values for the current server. Follow these steps:

- Click the **Environment Vars** option.
- For the variables listed, provide a value or ensure that the value shown is accurate for the current server.



If no variables are listed, that means that none were established during process development in Lombardi Authoring Environment.

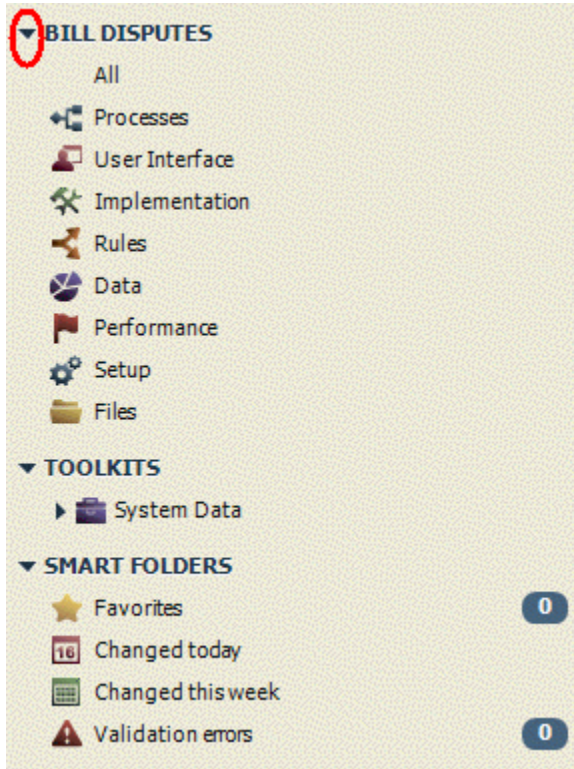
Managing library items in the Designer view

The procedures in this section provide details about how to effectively manage and maintain Lombardi library items in the Designer view. The Designer offers several tools to ensure that you can quickly and easily access items that you work with on a regular basis. Plus, the Designer enables you to move or copy items between existing or new process applications and toolkits. You can also revert to previous versions of individual library items using snapshots.

Before performing any of the following tasks, start Lombardi Authoring Environment and access the Designer view as explained in [Starting Lombardi Authoring Environment](#).

Navigating the library

When you open the Designer view in Lombardi Authoring Environment, the library is displayed on the left as shown in the following image:



You can perform the following actions in the library:

- Click the indicator highlighted in red in the preceding image to see the categories of library items in the current process application (named Bill Disputes).
- Click a category to see the individual items stored in the library. For example, click **Processes** to see a list of the business process definitions (BPDs) included in the process application named Bill Disputes. When viewing the list of individual library items, you can right-click an item to perform actions such as tagging or deleting items.
- Mouse over a category and then click the plus sign that displays next to the category to add a new library item. For example, if you click the plus sign next to Processes you can add several types of library items, including BPDs and participant groups.



When creating new library items, do not name those items `toolkit`. Doing so can cause issues when using JavaScript to refer to variables and other Lombardi objects that reside in the toolkit namespace.

- Mouse over the Toolkits category and then click the plus sign to create a dependency on a toolkit. See [Managing and using toolkits](#) for more information.
- In the Smart Folders area, you can manage favorites, tagged items, and your custom smart folders. See the following sections for more information.

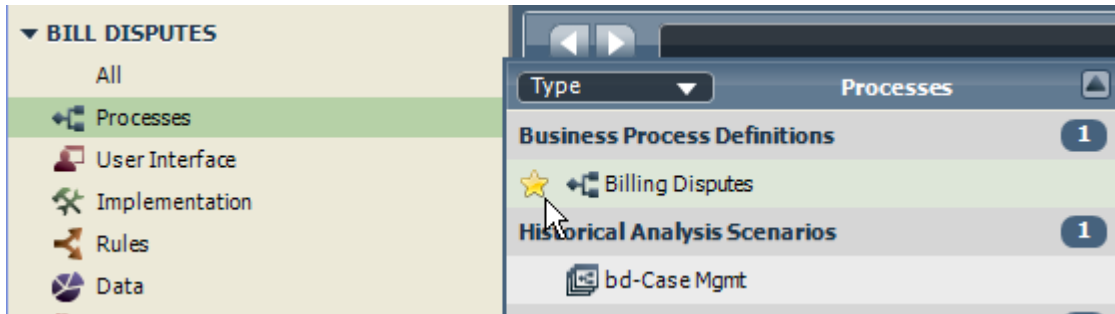
Creating favorites

You can mark library items as favorites for quick and easy access.

1. Click a category to see the individual items stored in the library.

For example, click **Processes** and then click **Business Process Definitions** (as shown in the following image) to see a list of the BPDs included in the current process application or toolkit.

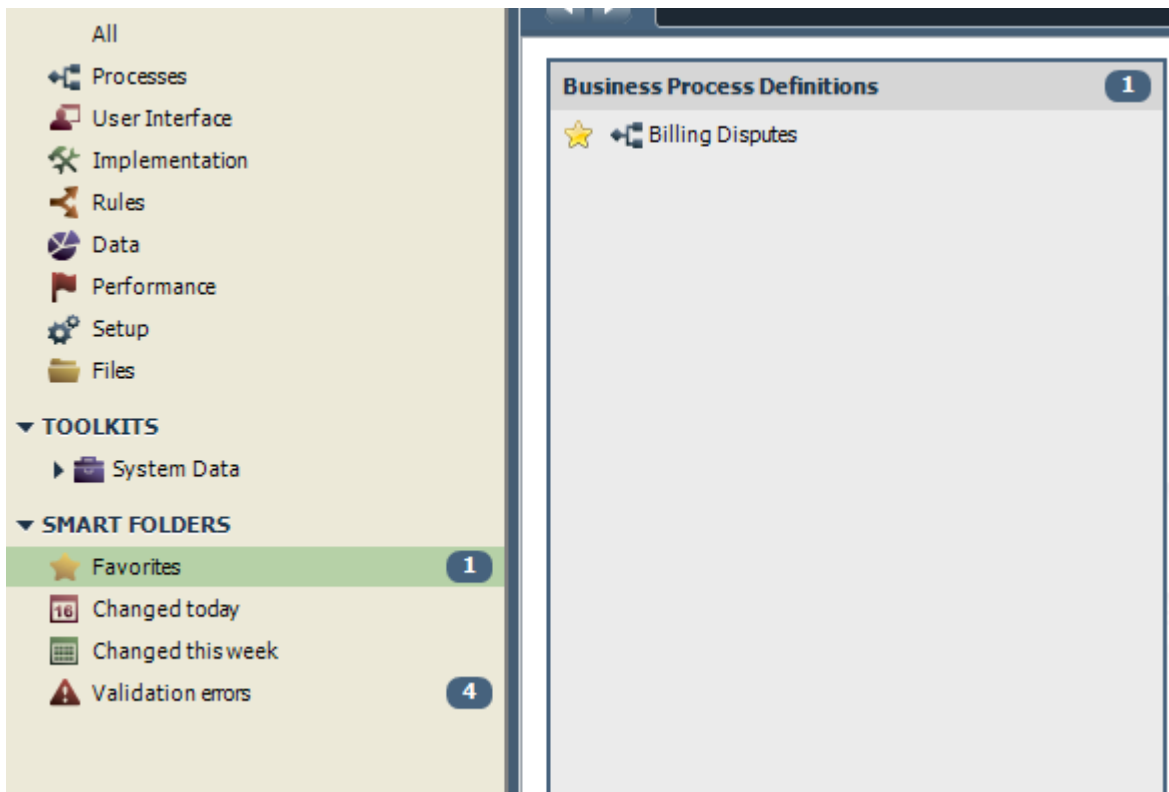
2. Move your cursor to the left of a BPD and click the star icon to highlight it as shown in the following example:



In the preceding example, the Billing Disputes BPD is now a favorite.

3. Go to the Smart Folders area, click **Favorites**, and then click a category to see all items marked as favorites.

In the following example, you can see the favorite added in the previous step:



If you want to remove a favorite, click the star icon next to the item in the Favorites list.

Tagging library items

By default you can sort library items by name or by type. If you want to sort library items using custom labels, you can attach a tag to them for quick and easy access.

1. Click a category to see the individual items stored in the library.

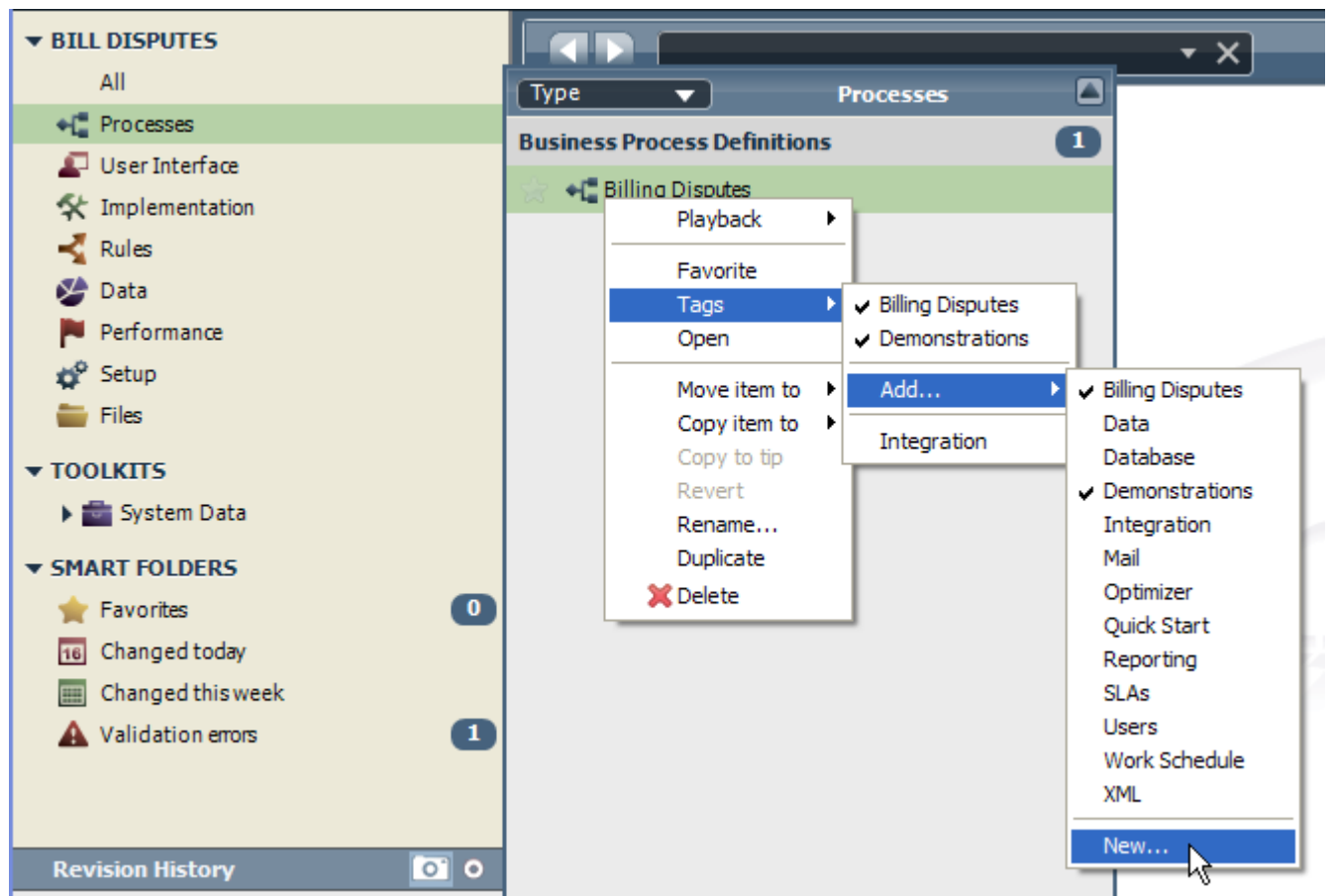
For example, click **Processes** and then click **Business Process Definitions** to see a list of the BPDs included in the current process application or toolkit.

2. Right-click a library item and select **Tags > Add** from the menu options.

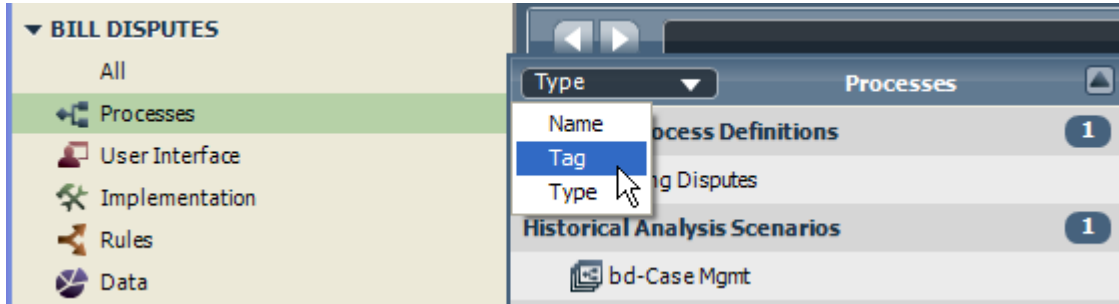


To choose multiple library items in a category, press and hold the Ctrl key and then click each item.

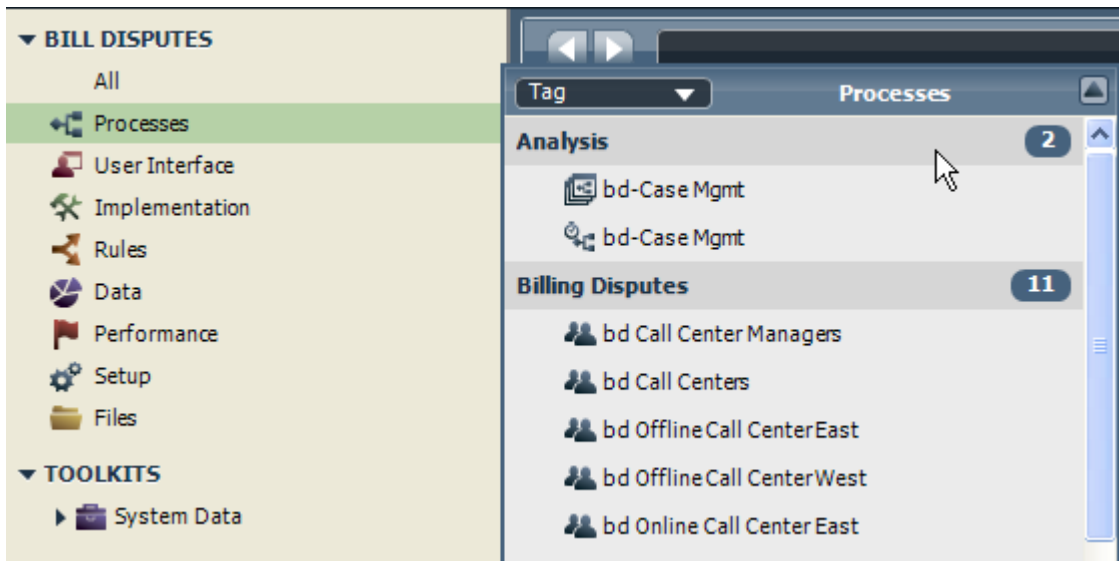
If the tag you want already exists, select it from the list. If not, select **New** from the list, enter the name for the new tag and click **OK**.



3. To view tagged items, click a category in the library and then use the drop-down list shown in the following image to switch to the **Tag** option:



Now library items are categorized according to tags as shown in the following image:



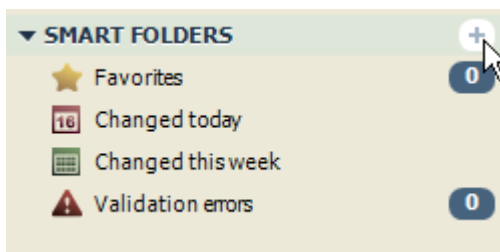
4. To remove an existing tag from a library item, right-click the library item and then select **Tags** > **[Tag_name]** from the menu options. The selected tag is no longer associated with the library item.

Organizing library items in smart folders

You can arrange library items in smart folders for quick and easy access. Lombardi Designer includes several default smart folders such as the **Changed today** folder, which includes all library items in the current process application that were changed on the current day. The **Changed today** and **Changed this week** smart folders include library items changed by all users who have access to the current process application.

In addition to the default smart folders, you can create custom smart folders as described in the following procedure. Custom smart folders can be private or shared.

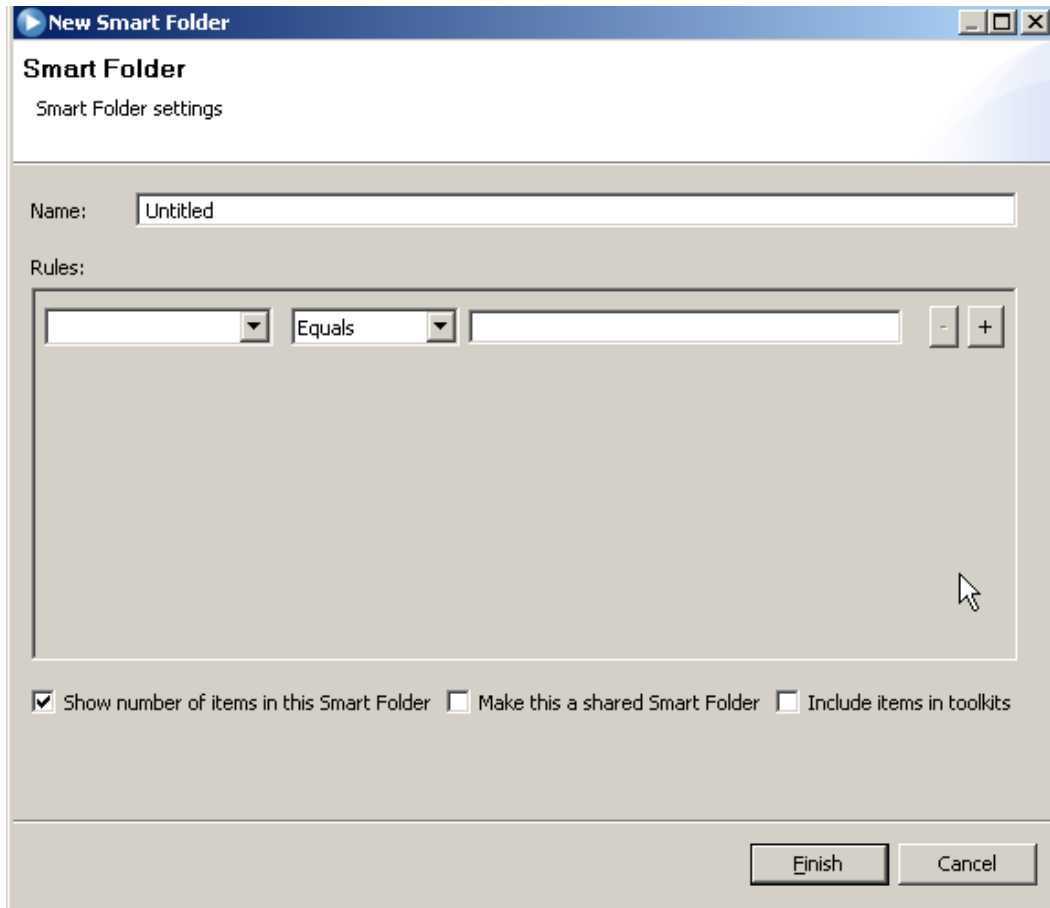
1. Click the plus sign next to **Smart Folders** in the Library as shown in the following image:



2. In the Create new smart folder dialog, enter the name for the new folder.
3. Select from the available menus to establish rules to determine which library items to include. You can include library items based on the presence of a particular tag, creation date, item type, and so on.

In the following example, the first rule establishes that we want library items that are marked as favorites.

4. Click the plus sign to add another rule. In the following example, our second rule establishes that we want library items that were modified before the selected date.



You can continue adding rules using the plus sign. When you click the plus sign next to a particular rule, the new rule is added immediately after the current rule. You can use the minus sign to remove unwanted rules.



Lombardi Designer evaluates the rules from top to bottom. If no rules match existing library items, the smart folder is empty.

5. You can optionally enable the following check boxes:

Show number of items in this smart folder	Enabling this option causes the Designer to display the number of library items in the folder when you mouse over or select the folder.
---	---

Make this a shared smart folder	If you enable this option, anyone who has access to the current process application or toolkit can view this smart folder and its contents, but cannot edit the rules for the folder.
Include items in toolkits	If you enable this option, library items within the toolkits that this process application depends upon are also included in the smart folder.

6. Click **Finish** to save the smart folder settings.

Copying or moving library items

In the library in the Designer view, you can copy or move existing library items to a new or existing process application or toolkit.



To move or copy items, you must have write access to the target process application or toolkit.

The following table describes how copy and move operations are different:

Copy a library item	Creates a new library item that is a duplicate of the original item. The newly created item is not associated with the original item from which it was copied.
Move a library item	Relocates the original item to a new or different process application or toolkit.

When you copy library items, any references to those items in the source process application or toolkit are unaffected because the original items remain and are still referenced.

When you move library items, those changes can affect existing implementations and other references. For example, if the implementation for an activity is a nested process and you move the nested process, whether the activity's implementation (reference to the nested process) is good or is broken depends upon where you move the nested process as described in the following table:

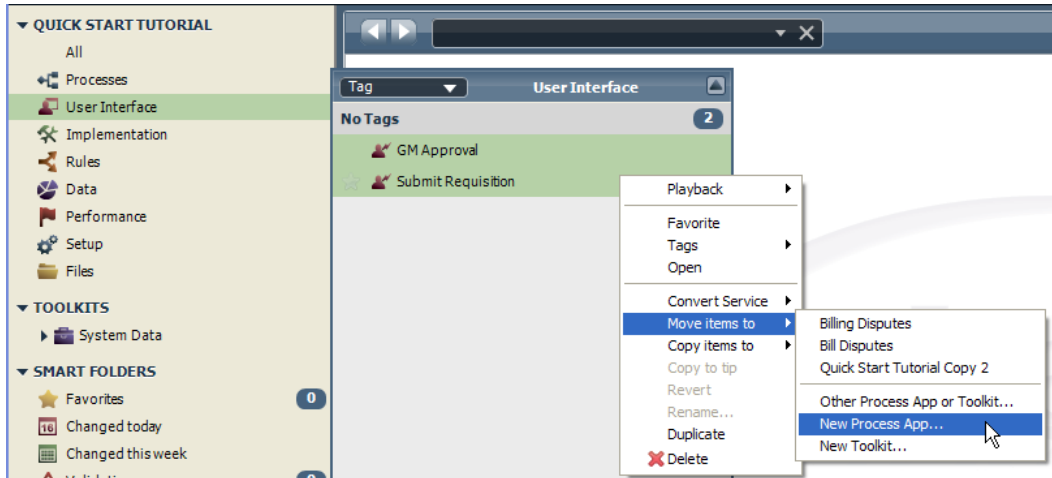
If you move the nested process to...	The reference...
A new toolkit	Is good because Lombardi automatically creates a new dependency on the new toolkit.
An existing toolkit that the source process application is <i>not</i> currently using	Is good because Lombardi automatically creates a new dependency on the existing toolkit.
An existing toolkit that the source process application is currently using	Is broken until you update the existing toolkit dependency to the new snapshot of the toolkit that Lombardi automatically creates. See Updating a toolkit dependency in the Designer view for instructions.
A new or existing process application	Is broken because process applications cannot depend upon each other. When moving library items to process applications, be sure to move all related items to avoid broken references. Lombardi automatically resolves broken references when related items are not moved simultaneously.



To ensure that you move all related library items, identify related items using the Where Used tab at the bottom of the Designer, and then tag the related items. You can then display the tagged items as described in [Tagging library items](#), which enables you to easily move or copy them as a group.

To move or copy library items, follow these steps:

1. Select the library items that you want to move or copy. To choose multiple items in a category, press and hold the Ctrl key and then click each item.
2. Right-click and select **Copy items to** or **Move items to** from the menu options as shown in the following image.



3. Select the option that you want from the pop-up menu. The top of the list displays all process applications and toolkits created by the current user. The other options include the following:

Other Process App or Toolkit...	When you select this option, the Designer displays all existing process applications and toolkits to which you have write access.
New Process Application...	When prompted, enter a name for the new process application and click OK. The acronym for the new process application is set to the first letter of each word in the process application name.
New Toolkit...	When prompted, enter a name for the new toolkit and click OK. The acronym for the new toolkit is set to the first letter of each word in the toolkit name.

The Designer moves or copies the selected library items to the designated process application or toolkit. If you copy or move a library item with the same name as an item in the target process application or toolkit, the Designer appends a number to the end of the moved or copied item. The original item in the target has no number and the copied or moved item name is followed by the number 2.

Reverting to a previous version of a library item

In the library in the Designer view, you can revert to a previous version of a library item. For example, if you realize that a service you created in a previous snapshot is closer to the implementation that you need than the current version, you can revert to the previous version of the service.

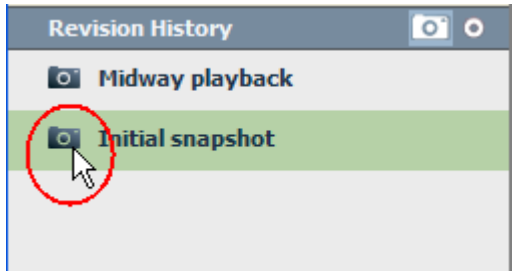


Before reverting a library item, open the item and check the **Where Used** tab at the bottom of the Designer. If the item is being used in multiple processes or services, check with the other developers on your team before making changes. If a single item is being implemented across a process application, it is best to create a toolkit, add the item to the toolkit, and then create a dependency on the toolkit.

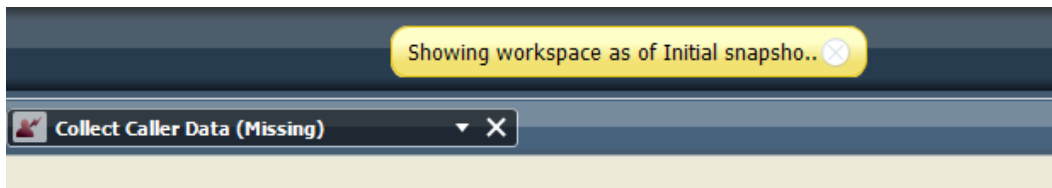
To revert to a previous version of a library item, follow these steps:

1. Go to **Revision History** in the Designer.

- Open the snapshot where the previous version of the library item that you want resides by clicking the indicator next to the snapshot as shown in the following image:



The Designer displays a message in the toolbar to indicate that it is showing all items in the process application as they existed when the snapshot called Initial snapshot was created:



- Find the item that you want, right-click it, and select **Revert**.

The current version of the same library item in your process application is replaced with this snapshot version.

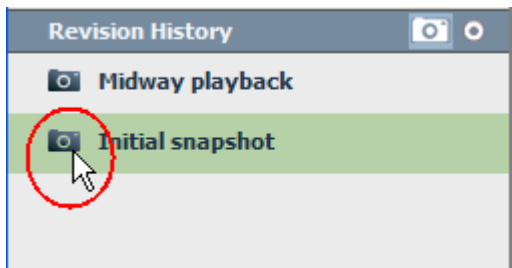
As soon as the revert operation is complete, the Designer returns to the current working version of your process application. All library items are current and the item that you chose to revert has been replaced by its previous version.

Copying a library item from a snapshot

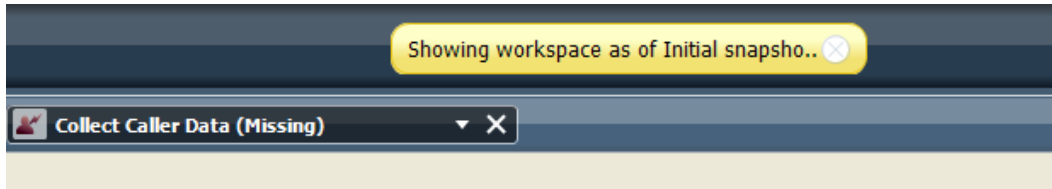
In the library in the Designer view, you can copy a library item from an older snapshot to the current version of your process application. For example, if a service was developed and then deleted, you can go back to a snapshot where the service was still present and copy it to the current working version of your process application.

To copy a library item from a previous snapshot, follow these steps:

- Go to **Revision History** in the Designer.
- Open the snapshot where the library item that you want resides by clicking the indicator next to the snapshot as shown in the following image:

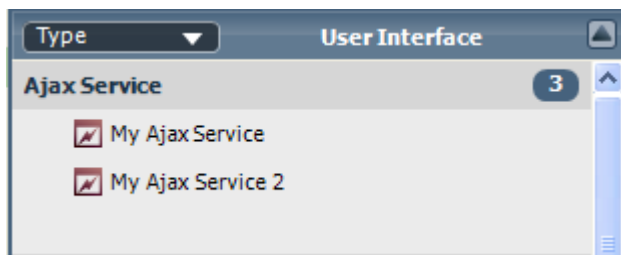


The Designer displays a message in the toolbar to indicate that it is showing all items in the process application as they existed when the snapshot called Initial snapshot was created:



3. Find the item that you want, right-click, and select **Copy to Tip**.

The selected library item is now available in the current working version of your process application. If you copy an old item with the same name as a current item, the Designer appends a number to the end of the older copied item:



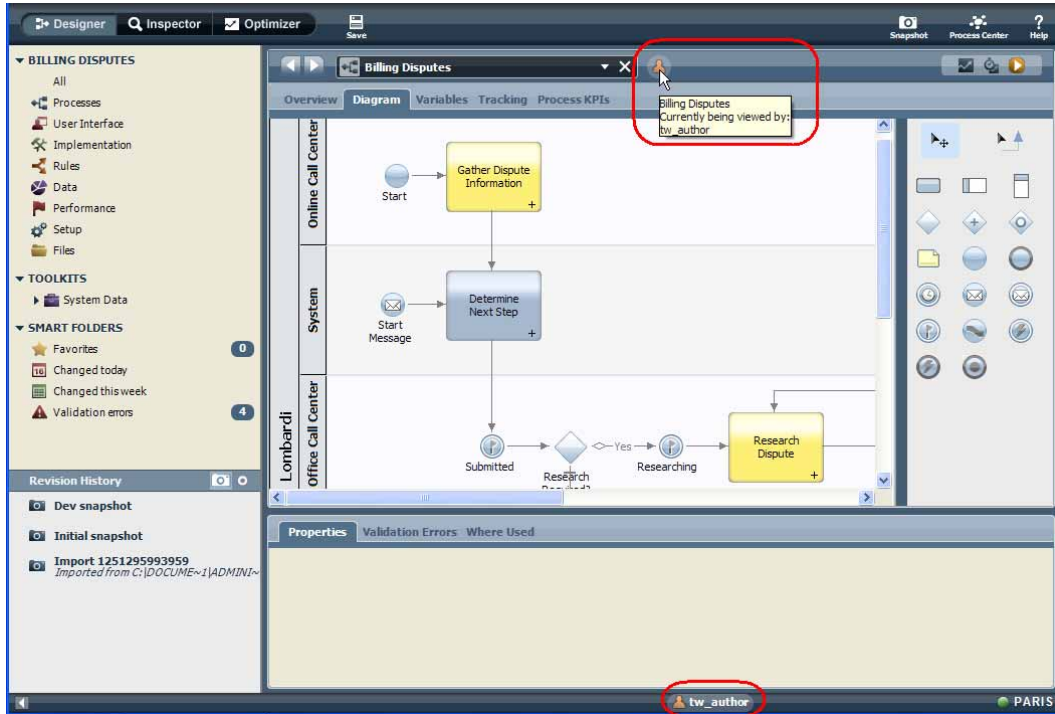
Understanding concurrent editing

Lombardi enables multiple users to simultaneously access and make changes to the library items in the Designer view.



Each user must be connected to the same Process Center and each user must have write access to the process application or toolkit where the library items reside. When editing concurrently with other users, ensure your connection status is good as described in [Authoring Environment tips and shortcuts](#).

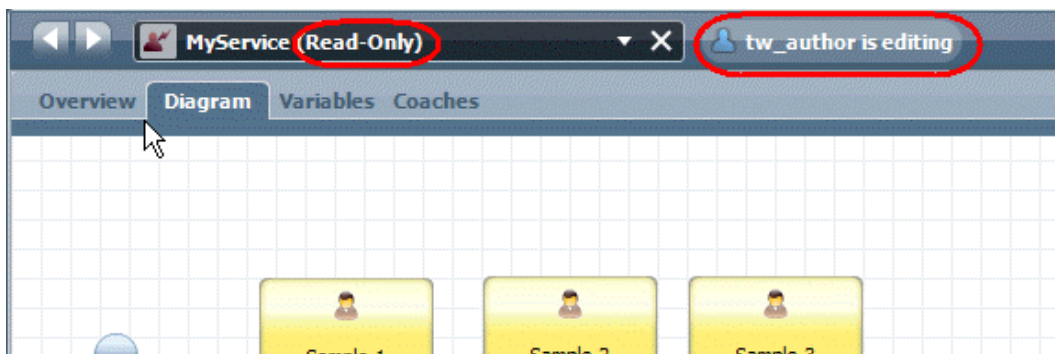
When working in the Designer view, you can see when other users are working in the same process application and library item as shown in the following image:



When multiple users are working on the same library item, such as a BPD, each user can see the changes as edits are saved. To ensure that all users are aware of the library items that are open and the changes being made, the Designer provides the following notifications:

- If you open a BPD or library item that is already opened by other users, the Designer notifies you.
- If another user starts to edit a BPD or other library item that you have open, the Designer notifies you that the edits are taking place and who is making the changes as shown in the following example.
- If you attempt to edit a library item that another user is currently editing, the Designer notifies you that you cannot make changes by displaying the text Read Only next to the item name as shown in the following example.
- If multiple users start to edit a library item at the same time, the Designer displays a warning icon to notify each user and then enables them to either immediately save their changes or discard them.

The following example shows the notification provided by the Designer when another user is editing an item that you have open:



With concurrent editing, you can collaborate with other team members to build the library items that you need for your project. For example, you can communicate about your ideas and edits using instant messaging and see the results in the Designer view as they happen.

Subscribing to Blueprint processes

If members of your team are discovering and developing processes in Blueprint, you can easily access those processes from Lombardi Authoring Environment, alter them, and then share the results with Blueprint users. To learn more about Blueprint, see <http://blueprint.lombardi.com>.

Before performing any of the following tasks, start Lombardi Authoring Environment and open the Designer view as explained in [Starting Lombardi Authoring Environment](#).

Subscribing to Blueprint processes in the Designer view

If you need to access one or more Blueprint processes from Lombardi Authoring Environment, you can create a subscription as described in the following procedure.

When you subscribe to Blueprint processes, be aware of the following:

- To subscribe to Blueprint processes, you need a Blueprint account. See <http://blueprint.lombardi.com> for more information.
- You cannot subscribe to Blueprint processes from a toolkit in the Designer in Lombardi Authoring Environment. Blueprint subscriptions are available only from existing process applications.
- You can subscribe to multiple Blueprint processes from a single process application.
- Sub-processes are included as separate BPDs within a subscription, whereas linked processes are included as a separate subscription.

1. Click the plus sign next to **Blueprints** in the library in the Designer view and select **Subscribe to Blueprint Process**:



2. Type in your Blueprint user name and password and click **Next**.



You can set the Blueprint server URL and user name in the preferences for Lombardi Authoring Environment. Select **File > Preferences** from the main menu, expand the **Lombardi** entry, and then click the **Blueprint** option. Enter the URL and user name that you would like to use for Blueprint integration.

3. Click the check box next to the processes to which you want to subscribe and click **Next**.

4. On the Summary page, verify the list of Blueprint processes shown. If you want the subscription name displayed in the Designer to be different from the Blueprint process name, type in a new name in the Subscription Name field. When you are done verifying processes and changing subscription names, click **Finish**.



Subscription names cannot exceed 64 characters.

The resulting subscriptions are displayed within the Blueprints category in the library.

If you go to the All category and click to see all library items in the current process application, Blueprint items are marked with a Blueprint icon and tag. When you subscribe to a Blueprint process, all associated library items (such as timing intervals and tracking groups) are automatically created.



If you subscribe to Blueprint processes and subsequently need to change those processes, you can open the processes in Blueprint directly from Lombardi Authoring Environment, make the required changes, and then update the process subscriptions as described in the following procedures.

Opening subscribed processes in Blueprint

If you subscribe to one or more Blueprint processes and then decide to make changes to those processes, you can open the processes in Blueprint while working in Lombardi Authoring Environment.

1. In the Blueprints category in the library, right-click the process that you want to change, and select **Open in Blueprint**.
2. Make the required changes.
3. Update the process subscription in Lombardi Authoring Environment as described in [Updating Blueprint processes in the Designer view](#).

Updating Blueprint processes in the Designer view

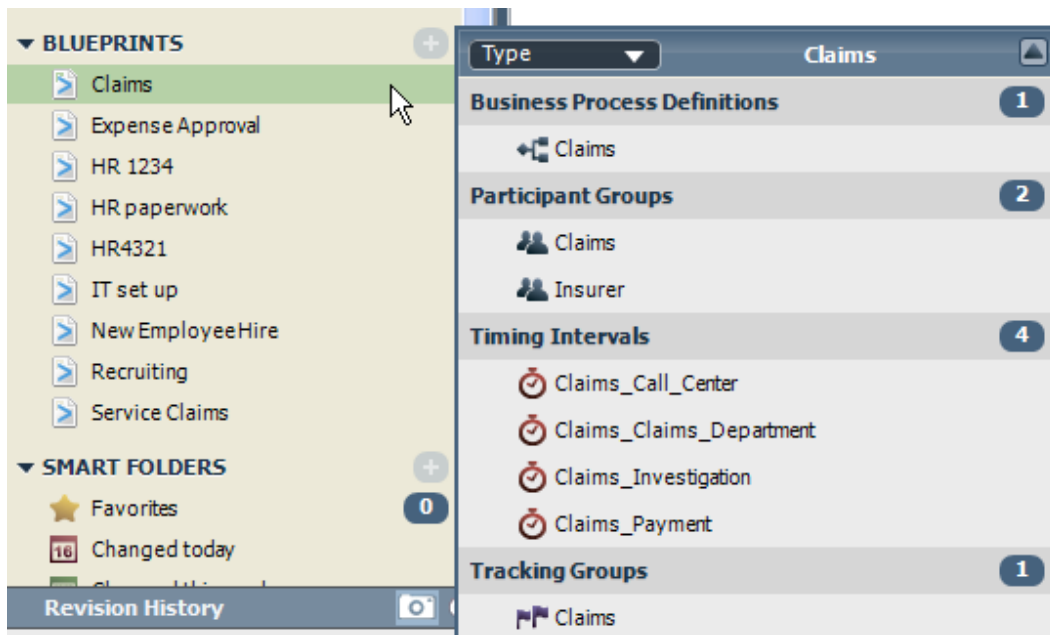
If you subscribe to Blueprint processes and then make changes to those processes in Blueprint, you can update your processes in Lombardi Authoring Environment as described in the following procedure.

When you update, be aware of the following:

- To update Blueprint process subscriptions, you need a Blueprint account. See <http://blueprint.lombardi.com> for more information.
 - You cannot update Blueprint subscriptions when viewing a historical snapshot in the Designer view in Lombardi Authoring Environment.
 - Updates made in Blueprint can be applied to subscribed Blueprint processes and supporting implementations in the Designer in Lombardi Authoring Environment. However, you cannot update your Blueprint processes with changes that you make in Lombardi Authoring Environment. So, if you have made changes in Lombardi Authoring Environment that are not reflected in the same assets in Blueprint, the Authoring Environment changes are overwritten when you update.
1. In the Blueprints category in the library, right-click the subscription that includes items that have been changed in Blueprint, and select **Update from Blueprint**.



To see all items that will be updated, you can click on a process subscription to display its library items as shown in the following example:



2. Type in your Blueprint user name and password and click **Next**.



You can set the Blueprint server URL and user name in the preferences for Lombardi Authoring Environment. Select **File > Preferences** from the main menu, expand the **Lombardi** entry, and then click the **Blueprint** option. Enter the URL and user name that you would like to use for Blueprint integration.

3. The updates from Blueprint are applied to the subscribed items in Lombardi Authoring Environment. You should snapshot the changes so that you have a record of the updates.



If Blueprint users make additional changes after you update, you need to update the process subscription again to see the most recent changes in Lombardi Authoring Environment.

Removing Blueprint subscriptions from the Designer view

You can remove existing subscriptions to Blueprint processes from the Designer view in Lombardi Authoring Environment as described in the following procedure. When you remove subscriptions, be aware of the following:

- When you remove a subscription, the library items from Blueprint remain in the process application or toolkit but are no longer tagged as Blueprint items. This enables you to manage development of all assets solely in Lombardi when the Blueprint discovery phase is over.
- When you remove a subscription, historical snapshots in the Designer view in Lombardi Authoring Environment retain their subscription details and library items.

- If you remove a subscription and then later restore to a snapshot that includes the subscription, the entire Blueprint subscription is restored.

To remove existing subscriptions to Blueprint processes, follow these steps:

1. In the Blueprints category in the library, right-click the subscription that you want to remove and select **Remove Subscription**.
2. When prompted, confirm that you want to remove the subscription.

Managing external files

External files are images, style sheets, JAR files, or other assets that are part of a Lombardi implementation, but developed outside of Lombardi. You can add these external files to your process application or toolkit in the Designer view so that all project assets are included in the Process Center repository. Adding these files to your process application ensures that all required assets are available and installed when your project is ready for testing or production.

The types of files that you can add to your Lombardi projects as managed assets include the following:

Images	.png, .gif, and .jpg
Other files	.html, .css, .js, .jar, and .zip files

Before adding managed files, be aware of the following requirements:

- You must have write access to the current process application or toolkit to add external files as assets.
- The files that you add must be 100 MB or less.



You must have write access to the current process application or toolkit to add external files as assets.

Adding managed files

To add external files to your process application or toolkit:

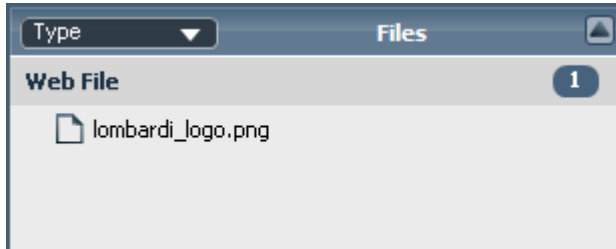
1. Click the plus sign next to **Files** in the library in the Designer view and select either Web File or Server File.

Select Web File if you want to add images, .css files, or other assets to use in your Lombardi Coaches.

Select Server File if you want to add JAR or JS files or some other type of asset to use in a Lombardi implementation.

2. In the New File dialog, click the **Browse** button to search for the file that you want.
3. Select the file that you want and then click the **Finish** button.

The external file is now displayed in the Designer view in the Files category as shown in the following image. The name of the file is the same as the original.



You can also drag and drop files or folders from your file system directly to the Files category as described in the following procedure.

Adding managed files using drag and drop

To drag and drop external files to your process application or toolkit:

1. Browse your file system and find the files or folder that you want to add to your Lombardi project.
2. Click the files or folders and then drag them to the Files category in the Designer in Lombardi Authoring Environment.

When you add folders, all the files within the folder are added as individual assets. All files retain their original names.

The external files are now displayed in the Designer view in the Files category and the files can be used throughout your current process application or toolkit.



You can also drag an image file from your file system and place it directly on a Coach that you are creating to simultaneously add the file as a managed file and design your Coach.

Updating managed files

To update managed files:

1. Change the files as needed using the appropriate application and then save the files.
2. Drag and drop the updated files (or the folder in which the files reside) from your file system to the Files category in the Designer in Lombardi Authoring Environment.

For original managed files that have the same name as the updated files, the Designer replaces all existing files with the updated versions. And, all current implementations of the files throughout your project are also automatically updated.



If you want to replace a current file with a new file, open the file in the Designer, click the Browse button, select the new or changed file, and click Open.

Replacing a managed file

To replace a current file with a new file:

1. Click the **Files** category in the library in the Designer view in Lombardi Authoring Environment.
2. Double-click the managed file to open it.

3. In the File editor, click the **Browse** button.
4. Find the file that you want in your file system, click to select it, and then click the **Open** button.

The selected file is now the managed file in your Lombardi project.

Using managed files

When you add external assets to be managed as part of your Lombardi project, you can easily use those files in your Coaches and other implementations for activities, such as services. For example, when you add a JavaScript (.js) file to be managed as part of your project, you can use the functions and variables within the JS file in the scripts and scriptlets that you develop throughout your Lombardi project. JS files that are included as managed files are able to access both dynamic library JAR files and any JAR files within the managed files themselves.

The Image and File Attachment Coach controls enable you to easily use managed files. When developing a Human service, you can drag an Image or File Attachment control onto your Coach and then select the managed file that you want from the presentation properties. Plus, for images, you can simply drag image files from the library in the Designer directly onto your Coach form. When you do, the image appears when you preview your Coach and the Coach control is automatically added. If you update or replace a managed file, that updated file is automatically used in your Coach.

You can also choose a custom CSS (previously added as a managed file) when customizing the style of a particular Coach. Managed files are also included in the list of options that you can select for the implementation of various BPD components, where appropriate. For example, when using the Java Integration component in a Integration service, you can select a JAR file previously added as a managed file.

Deleting managed files

You must have write access to the current process application or toolkit to remove managed files.

To remove managed files:

1. Click the **Files** category in the library in the Designer view in Lombardi Authoring Environment.
2. Right click the managed file that you want to remove and select **Delete**.



If the deleted file is used as the implementation for a service or other component, the implementation will be broken. To ensure that you do not delete necessary files, open the file and check the **Where Used** tab before deleting.

Importing files from previous versions of Lombardi

When you import files from previous versions of Lombardi, be aware of the following:

- You can import Lombardi export files with any extension such as .zip, .export, and so on.
- You must import using the Process Center Console as described in the following procedure. Command-line import is not supported in Lombardi 7. [Starting Lombardi Authoring Environment](#) explains how to start and use the Process Center Console.
- Lombardi 7 supports import of Lombardi 6.1 and earlier exports. (Imports of Lombardi 6.2 exports are supported only when the export does not include conditional activities or Critical Path Management

features. If the 6.2 export includes conditional activities or Critical Path Management features, import into Lombardi 7 fails.)

- All library items within a single export file are placed into a new process application in Lombardi 7 as described in the following procedure.
- Each imported library item is tagged with the folder name in which it was stored in the previous version of Lombardi. The tags enable you to easily find library items by clicking a category in the library and then using the drop-down menu to switch to the Tag option. To learn more, see [Tagging library items](#).
- Before importing large export files, ensure that the log level for the Authoring Environment is set to `error` and not `debug`. By default, the log level is set to `error`. You can change the log level in the following file:

```
[Lombardi_home]/Lombardi Authoring
Environment/teamworks/eclipse/plugins/teamworks.appserver.[appserver_type]_[teamworks_version].jar/resources/log4j.xml
```

- If you experience problems importing large exports from previous versions of Lombardi and the `[Lombardi_home]\process-center\logs\tw-error.log` file contains `Out Of Memory` error messages, switching from a 32-bit JVM to 64-bit JVM can resolve this problem. You can contact IBM Customer Support for further assistance.

To import files from previous versions of Lombardi:

1. In the Process Center Console, select the **Process Apps** tab.
2. Click the **Import Process App** option shown in the following image:



3. In the Import Process App dialog, click the **Browse** button to locate the export file that you want to import.
4. Click the **Next** button to continue.
5. The Process Center Console requests the information required to create the new process application to contain the imported assets.

Enter a name and an acronym for the new process application.

The acronym for a process application must be unique and is limited to seven characters. Lombardi uses the acronym as an identifier for this process application and the library items that it contains. For example, when manipulating the items within the process application using the Lombardi JavaScript API, you can use the acronym to specify the namespace of the items.

Providing a description is optional. When you enter a description, you can view it in the Process Center Console by clicking the question mark next to the process application name.

6. Click the **Import** button to import the selected file.

7. When the import completes, you can see the new process application listed in the Process Apps tab. You can grant access to other users as described in [Managing access to process applications and toolkits](#).

Administrators can see a log of all imports by clicking the **Admin** option at the top-right of the Process Center Console and then clicking **Import and Export Log**.

8. Click the **Open in Designer** option to view and edit the imported assets.

When importing from a previous version of Lombardi, all library items are placed into a single process application. For larger imports, you should examine the resulting process application and move groups of related items to other process applications and toolkits, according to an organization that makes sense for your development efforts. For example, if the imported file includes multiple library items to be shared across projects, move those items to a toolkit, following the directions provided in [Copying or moving library items](#).

When you are finished organizing library items in individual process applications and toolkits, you should verify the implementation and data mappings of activities and services to ensure that all required implementations have been properly updated to reflect their new location.

Modeling processes

A process is the major unit of logic in IBM WebSphere Lombardi Edition. It is the container for all components of a process definition, including services, activities and gateways; timer, message, and exception events; sequence lines, rules, and variables. When you model a process, you are creating a reusable Business Process Definition (BPD).

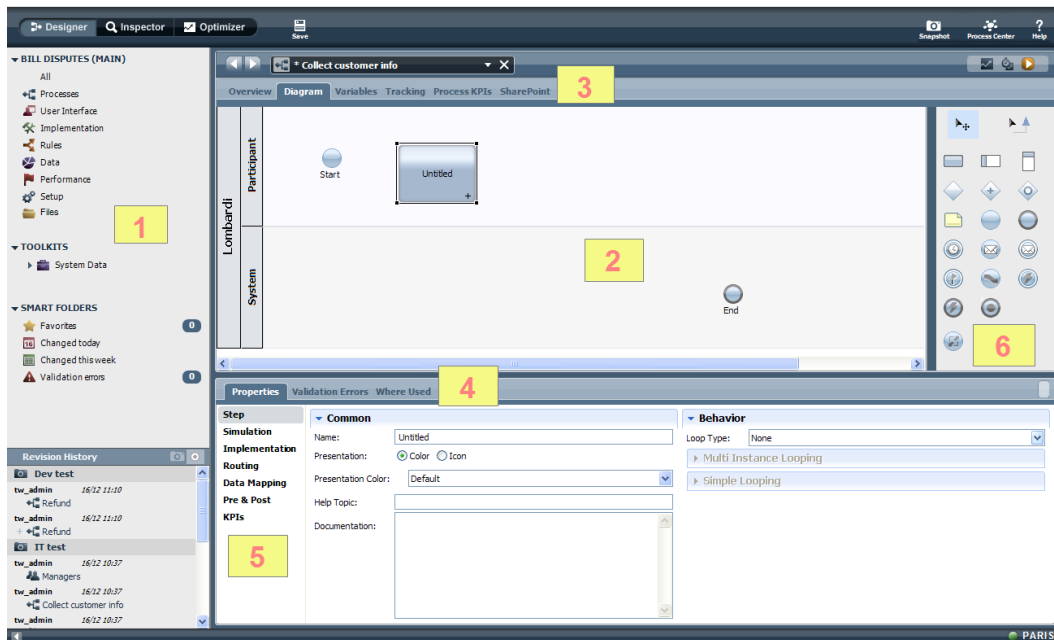
Process components enable you to define the process workflow for end users, creating logic inside a process and integrating with other applications and data sources. To understand what occurs inside a process at run time, it is important to understand the components that make up a process at design time.

Building processes in Lombardi



Many different individuals from various organizations are usually involved in developing processes using Lombardi. The overriding concern is to ensure that you are building the best possible solution for meeting the stated goals of your project. To ensure successful results, team members should work together to capture process requirements and iteratively develop the model and its implementations as discussed in [Planning Lombardi projects](#).

Using the Designer in Lombardi Authoring Environment

To access and begin using Lombardi Authoring Environment, follow the instructions provided in [Starting Lombardi Authoring Environment](#). The Designer interface provides the tools that you need to model your processes in Lombardi. The following image and corresponding table describe the parts of the Designer that you interact with when modeling processes and implementing the steps in those processes:






















1	Library	Provides access to the library items for the current process application. In this example, the process application is named Bill Disputes. You can create new library items and edit existing ones as described in Managing library items in the Designer view .
---	---------	--






		 <p>Access to process applications is controlled by users who have administrative access to the application. See Managing access to the Process Center repository for more information.</p>
2	Main canvas	Provides the area in which you can model your process using the components from the palette. See the following section to learn more about available process components. If you click a tab other than the Diagram tab, provides an interface for managing various aspects of your process such as variables, tracking, and so on.
3	Tabs to control display in main canvas area	The tab that you select determines the tool displayed in the main canvas area. For example, if you select the Variables tab, the Designer provides a dialog for creating and managing variables for the current process.
4	Properties tab	Shows the properties for the component currently selected in the diagram. In the example, the only activity in the diagram is selected and so its properties are displayed. (The other available tabs provide information about the library item that is currently open. So in this example, if you click the Validation Errors tab, you would see errors for the BPD that is currently open and if you click the Where Used tab, you would see where the currently open BPD is used within the current process application.)
5	Options to control display in properties area	The option that you select determines the properties shown for the currently selected component in the diagram. In the example, the Step option is selected, which enables you to name the activity and set its behavior.
6	Palette	<p>Provides components that you can use to model your process. Each available component is described in the following section. Drag a component from the palette to the diagram and then use the properties to control the implementation of the component.</p>  <p>You can hide the entire palette by clicking the colored border to the left of the available components. Click the same border to restore the palette and view the available components.</p>

Understanding process components

When developing the process diagram in the Designer in Lombardi Authoring Environment, the following tools and components are available from the palette:

Component icon	Description
 Selection Tool	Enables you to select and move components on the diagram.
 Sequence Flow	Enables you to connect process components to establish the order in which the steps in the process occur.
 Lane	Add lanes to your process diagram to hold the activities and events that take place during process execution. Lanes typically represent departments within a business organization. For example, you can add a Human Resources lane to hold all activities to be handled by members of the HR department during process execution.

Component icon	Description
	 <p>When you create a process, Lombardi automatically creates a pool to hold all added lanes. The default name of the pool is Lombardi, which you can change by clicking the pool in the diagram and then editing its properties.</p>
 Milestone	<p>Add milestones to your process diagram to illustrate the phases of process execution. For example, you can add a Planning milestone to capture the activities across lanes that occur in an initial phase of a process.</p>
 Start Event	<p>Use to model the start of a process if you want to manually start the process. Use the Start Message Event if you want an incoming message or event to kick off the process.</p>  <p>A start event is automatically included each time you create a BPD.</p>
 Activity	<p>Use to model the steps in your process, choosing the implementation best suited for each particular step. To learn about the options for implementing activities, see Advanced modeling tasks.</p>
 Timer Event	<p>Use to specify a time interval after or before which some activity is performed. Timer events can be attached to activities or included in the process flow with sequence lines.</p>
 Intermediate Tracking Event	<p>Use to indicate a point in a process at which you want Lombardi to capture the run-time data for reporting purposes. For examples of how tracking events are implemented, see Creating a basic custom report and Creating a more advanced custom report.</p>
 Decision Gateway	<p>Use to model a point in the process execution where <i>only one</i> of several paths can be followed, depending on a condition.</p>
 Simple Split/Join	<p>Use a simple split when you need to split, or diverge, the process along more than one path. Use simple splits when you want the process to follow <i>all</i> available paths.</p>
 Simple Split/Join	<p>Use a simple join to converge, or join, multiple paths into a single path after each path has completed its run-time execution. Use simple joins when you want to converge <i>all</i> available paths.</p>
 Note	<p>Use to add information about the overall process or each step in the process to the diagram. Adding notes helps other developers understand your design.</p>
 End Event	<p>Use to end process execution.</p>  <p>An end event is automatically included each time you create a BPD.</p>
 Start Message Event	<p>Use to model the start of a process if you want an incoming message or event to kick off the process.</p>
 Start Ad-hoc Event	<p>Use when you need to include ad-hoc actions that can be executed at any time during process execution. For example, you can include an ad-hoc event to enable end users to cancel a customer order at any time during the ordering process.</p>
 Intermediate Message Event	<p>Use to model a message event received while a process is running.</p>

Component icon	Description
 Terminate Event	Use to close all running tasks associated with the process and to cancel all outstanding timers. The process shows a status of <code>Terminated</code> in the Inspector.
 Intermediate Exception Event	Use to catch process execution exceptions and handle exceptions with an error handler activity or further process flow.
 End Exception Event	Use to throw an exception to parent processes.
 Conditional Split/Join	Use a conditional split when you need to split, or diverge, the process along more than one path and you want to follow <i>one or more</i> available paths based on a condition. A conditional split allows a process to follow more than one path, but not all at once.
 Conditional Split/Join	Use a conditional join to converge, or join, multiple paths into a single path after <i>one or more</i> paths has completed its run-time execution.

Basic modeling tasks

The following steps list the basic tasks to create a process in the Designer in Lombardi Authoring Environment:

Task	See...
1. Create a BPD in the Lombardi library.	Creating a BPD
2. Add swim lanes.	Adding lanes to a BPD
3. Assign participants to lanes.	Assigning participant groups to lanes
4. Add activities and other process components.	Adding activities and other process components to a BPD
5. Connect components with sequence flow lines.	Establishing process flow with sequence lines
6. Control process flow using gateways.	Using gateways
7. Implement activities.	Implementing activities
8. Add process variables to capture business data.	Adding process variables to a BPD
9. Add events.	Adding events to a BPD
10. Set environment variables.	Setting environment variables
11. Validate processes as you build them.	Validating processes

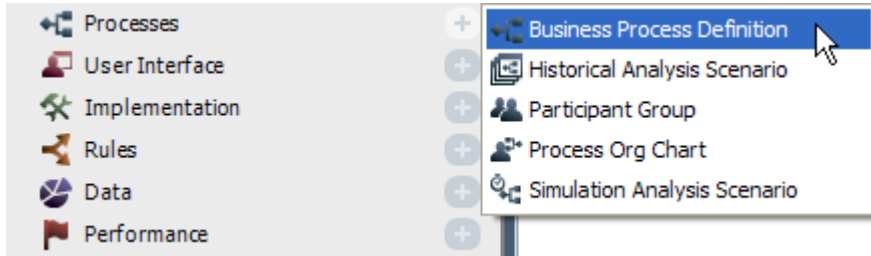
Creating a BPD

To model a process, you must first create a Business Process Definition (BPD). A BPD is a reusable model of a process, defining what is common to all run-time instances of that process model.



To create a BPD, you must have access to a process application in the Process Center repository. See [Managing access to the Process Center repository](#) for more information.

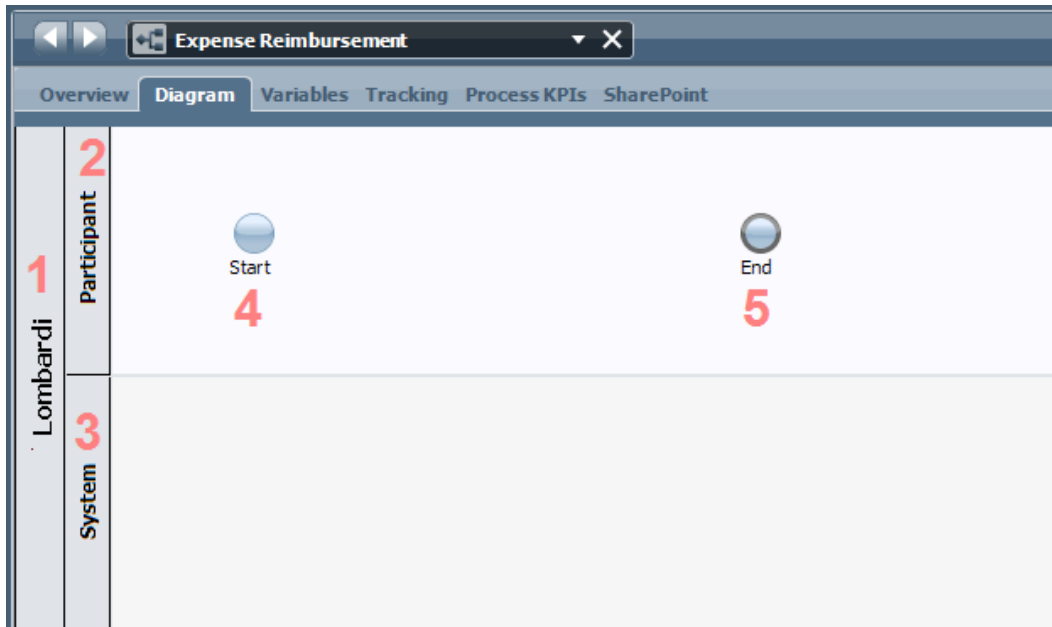
1. Start Lombardi Authoring Environment and open the appropriate process application in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. In the Designer view, click the plus sign next to **Processes** and select **Business Process Definition** from the list of components:



3. In the New Business Process Definition dialog, enter a name for the BPD and click **Finish**. (See [Lombardi naming conventions](#) for character limitations and other conventions for BPD names.)

Enter `Expense Reimbursement` if you want to create a basic BPD by following the procedures in [Basic modeling tasks](#).

4. Lombardi Designer displays the diagram of the process with the following default components:



Component	Default name	Description
1	Lombardi pool	The default pool to hold all lanes within the BPD. You can change the name by clicking the pool and editing its properties.
2	Participant lane	A default lane for end user activities. You can change the name by clicking the lane and editing its properties.
3	System lane	A default lane for system activities. You can change the name by clicking the lane and editing its properties.
4	Start event	Each BPD automatically includes a start event.
5	End event	Each BPD automatically includes an end event.

5. To continue creating a BPD and its basic components, see [Adding lanes to a BPD](#).

Adding lanes to a BPD

A Business Process Definition (BPD) needs to include a lane for each system or group of users who participates in a process. A lane is the container for all the activities to be performed by a specific group of users or system.

Using the BPD created in the preceding task, follow these steps to change the names of the default lanes and add additional lanes:

1. Click the Participant lane and in the **Name** field (in the Properties) enter `Employees`:

The screenshot displays the IBM Business Process Designer interface for a Business Process Definition (BPD) titled "Expense Reimbursement". The interface is divided into several sections:

- Diagram Area:** Shows a swimlane diagram with two lanes: "Employees" (top) and "System" (bottom). A "Start" event is positioned in the "Employees" lane. The "Lombardi" logo is visible on the left side of the diagram.
- Properties Panel:** Located at the bottom, it shows the "Common" properties for the selected "Start" event. The "Name" field is set to "Employees", and the "Presentation Color" is set to "Default".

This lane will contain activities that corporate employees will perform.

2. Leave the System lane as is for the sample Expense Reimbursement BPD that we're building.

System lanes hold activities that are to be completed by a Lombardi system. During implementation, a developer creates a service or writes the JavaScript necessary to complete the activities.

3. Drag a Lane component from the palette to the diagram so that the new lane is positioned beneath the System lane.

You can right-click the new lane and select **Move Lane Down** until it is positioned where you want it.

4. Click on the new lane in the diagram (named `Untitled_1` by default) and in the **Name** field in the properties, enter `Manager`.
5. Repeat steps 3 through 5 to add two lanes (`Accounts Payable` and `Monitor`) beneath the `Manager` lane as shown in the following image:



6. Click **Save** in the main toolbar.
7. To continue creating a BPD and its basic components, see [Assigning participant groups to lanes](#).

Assigning participant groups to lanes

Participant group lane assignments ensure that any activities that are not routed to a specific group or user have an automatic default assignment.

Before making lane assignments, you need to create the participant groups required for your process. To continue building a basic BPD by following the procedures in [Basic modeling tasks](#), follow the instructions in [Creating a participant group](#) to create the following groups:

- Employees
- Managers
- Accounts Payable
- Monitor

Be sure to add users and groups as standard members to the preceding participant groups. If you are not sure which members to add, add the `tw_allusers` security group for now so that you can run and test your BPD. You can leave all other participant group settings at the default values.



Each lane that you create is assigned to the All Users participant group by default. You can use this default participant group for running and testing your BPD in the Inspector. The All Users participant group includes all users who are members of the `tw_allusers` security group, which is a special security group that automatically includes all users in the system. See *Lombardi Administration Guide* for more information about the `tw_allusers` security group. See [Creating a participant group](#) for more information about defining the users that belong to a participant group.

To assign participant groups to lanes:

1. In the BPD diagram, click the lane in which you want to make the assignment.
2. In the Behavior section of the properties, click the **Select** button to choose the participant group that you want to use as the default group for this lane.



You need a default lane assignment to ensure that any activities that are not otherwise routed have an automatic default assignment. See [Routing activities](#) for more information.

3. Choose the participant group from the library.

Repeat steps 1 through 3 for each lane in the sample BPD: Employee, Manager, Accounts Payable, and Monitor. You should have created a matching participant group for each.

Leave the System lane assigned to the default System participant group.

4. Click **Save** in the main toolbar.

Adding activities and other process components to a BPD

You need to add activities to your process diagram to represent the steps in your process. You may also want to add other process components, such as notes, if you know they are necessary.



You can add components to the process diagram at any time and adjust sequencing and other elements as necessary.

To add activities to a process diagram:

1. Drag an activity from the palette into the appropriate lane.
2. In the **Step** tab in the properties, enter a name for the activity.

3. If you want to route an activity to someone other than the participant group assigned to the current lane, click the **Routing** tab and see [Routing activities](#) for more information.



The other tabs in the properties (Implementation, Data Mapping, and KPIs) are covered in the topics in the [Advanced modeling tasks](#) section.

To continue to build a basic BPD by following the procedures in [Basic modeling tasks](#):

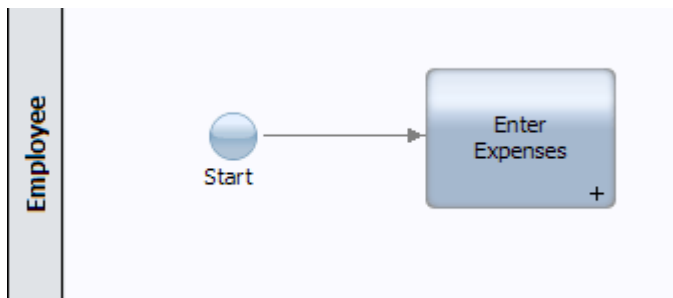
1. Drag an Activity into the `Employee` lane.
2. In the **Step** option in the properties, type the name: `Enter Expenses`.
Leave the presentation options set to the defaults.
3. Click the **Routing** tab in the properties. Notice that the `Lane Participant` option is specified for the `Assign To` field. By default, activities are assigned to the participant group established for the lane. This is the behavior that we want for our sample BPD.
4. Drag an Activity into the `System` lane.
5. In the **Step** option in the properties, type the name: `Generate Payment`.
Leave the default settings in the other properties tabs for now.
6. Drag an Activity into the `Manager` lane.
7. In the **Step** option in the properties, type the name: `Approval`.
Leave the default settings in the other properties tabs for now.
8. Drag an Activity into the `Accounts Payable` lane.
9. In the **Step** option in the properties, type the name: `Validation`.
Leave the default settings in the other properties tabs for now.
10. Drag an Activity into the `Monitor` lane.
11. In the **Step** option in the properties, type the name: `Payment Being Processed`.
Leave the default settings in the other properties tabs for now.
12. Click **Save** in the main toolbar.
Adjust your diagram so that it looks like the following image:



Establishing process flow with sequence lines

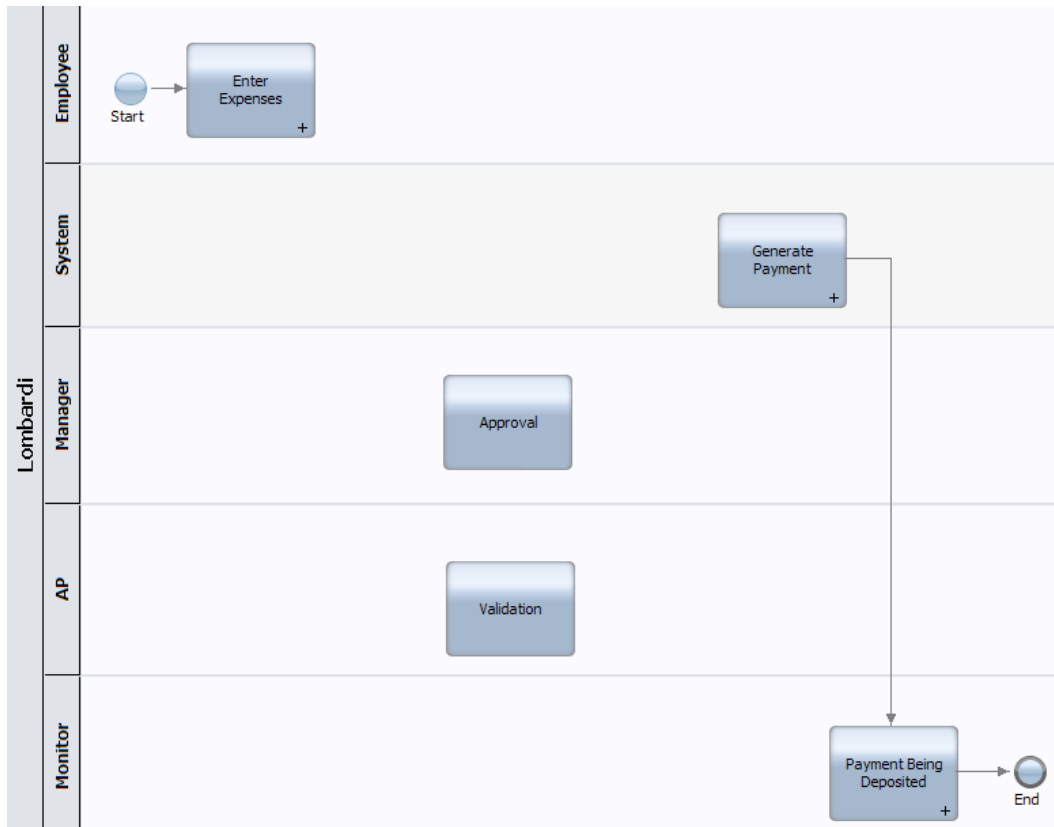
You need to connect activities and other elements in your BPD with sequence lines to establish the process flow.

1. From the palette, click to select the Sequence Flow tool.
2. In your process diagram, click the first component (normally the Start event) and then click the component that should follow the Start event in the process flow. The Sequence Flow tool connects the two items as shown in the following example:



3. Continue creating sequence lines as needed.
4. When you are finished establishing the process flow with sequence lines, click the Selection Tool in the palette to switch back to normal selection mode in the process diagram.

To continue to build a basic BPD by following the procedures in [Basic modeling tasks](#), add sequence lines so that your diagram looks like the following image:








In the following topic, [Using gateways](#), we'll add a simple split and join to the diagram and then complete the process flow.

Using gateways

Gateways control the divergence and convergence of sequence lines, determining branching and merging of the paths that a run-time process can take. You can think of conditional and decision gateways as questions that are asked at a particular point in the process flow. The question has a defined set of alternative answers, which act as gates. The process cannot proceed until a valid answer is provided. You can model questions using Javascript conditions, which are evaluated before the process is allowed to proceed.

You can model the following types of gateways in your process diagram:

Component icon	Gateway type	Description
 Simple Split/Join	Simple split (AND)	Use when you need to split, or diverge, the process along more than one path. Use simple splits when you want the process to follow <i>all</i> available paths.
 Simple Join	Simple join (AND)	Use to converge, or join, multiple paths into a single path after each path has completed its run-time execution. Use simple joins when you want to converge <i>all</i> available paths.

Component icon	Gateway type	Description
 Conditional Split/Join	Conditional split (OR)	Use when you need to split, or diverge, the process along more than one path and you want to follow <i>one or more</i> available paths based on conditions that you specify.
 Conditional Join	Conditional join (OR)	Use to converge, or join, multiple paths into a single path after <i>one or more</i> paths has completed its run-time execution.
 Decision Gateway	Decision gateway (XOR)	Use to model a point in the process execution where <i>only one</i> of several paths can be followed, depending on a condition.

Be aware of the following when using gateways:

- After you drag one of the preceding icons to your process diagram, you can choose any of the available gateway types if you decide to change the behavior.
- Split and join gateways are not always paired together. You can use one without the other.
- When you model conditional and decision gateways, if all conditions evaluate to false, the process follows the default sequence line. The default sequence line is the first line that you create from the gateway to a following step, but you can change the default sequence line at any time as described in the following procedure.

For more information about implementing conditional and decision gateways, see [Example gateways](#).

To add gateways to a process diagram:

1. Drag the type of gateway that you want from the palette to the process diagram. (The preceding table describes the types of gateways available.)
2. Using the Sequence Flow tool, create the necessary sequence lines to and from the gateway.



The default sequence line is the first line that you create from the gateway to a following step. You can change the default sequence line when specifying the implementation properties for a gateway as described in Step 5.

3. Click the gateway in the process diagram and then click the **Step** option in the properties.
4. Type a name for the gateway. The other fields are optional:

Name visible	By default, the name that you provide for the gateway displays in the process diagram. Disable this check box if you do not want the name displayed in the diagram.
Presentation Icon	If you want to use an icon other than the default provided by Lombardi, provide the pathname for the image that you want to use.
Documentation	Enter a description of the gateway.
Gateway Type	Ensure that the type of gateway you want to implement is selected from the drop-down list. The preceding table describes the types of gateways available.

5. For conditional splits and joins and decision gateways, click the Implementation tab. For each outgoing sequence line, enter the condition (in JavaScript) that controls whether the path is followed. (See [Example gateways](#) to understand the types of conditions that you can define.)

Ensure that the sequence line shown as the **Default Line** is the one that you want the process to follow if all conditions evaluate to false. If not, use the arrow icons to move the lines until the one that you want is designated the default. (The last line in the list is the default line.)



Default sequence lines do not require a condition.

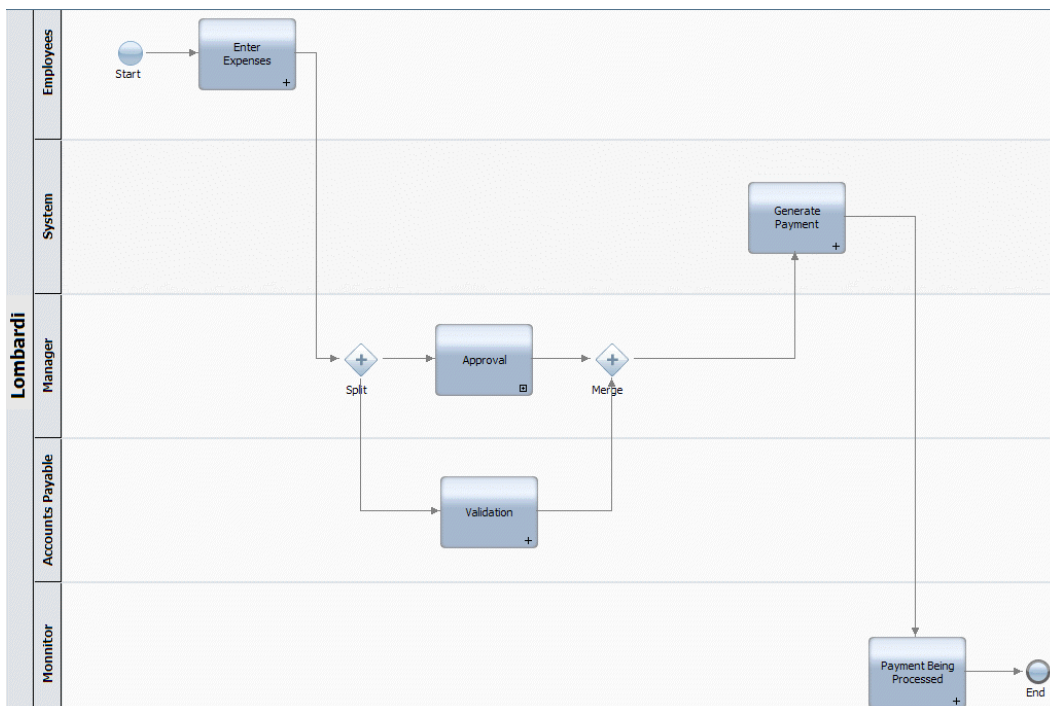
6. Click **Save** in the main toolbar.

To continue to build a basic BPD by following the procedures in [Basic modeling tasks](#):

1. Drag a simple split from the palette to the process diagram so that it directly precedes the Approval activity.
2. In the Step option in the properties for the simple split, type the name: `split`.
3. Drag a simple join from the palette to the process diagram so that it immediately follows the Approval activity.
4. In the Step option in the properties for the simple join, type the name: `merge`.

Adding the simple split and join enables us to model the Approval and Validation steps so that the process flow continues after both of these steps are complete. If you click the Implementation option in the properties for the simple split and join, you can see that there are no conditions for these types of gateways, which means all paths are followed.

5. Using the Sequence Flow tool, connect the Enter Expenses activity to the simple split and then continue to connect the process components so that your diagram looks like the following image:



Implementing activities

When implementing activities in the Designer in Lombardi Authoring Environment, you have several options. The following table lists the available options and provides a link to detailed information and procedures in the *Advanced modeling tasks* section.

Implementation option	Description	See...
Service	You can create the following types of services: Integration, Human, Ajax, Rule, and General System. The type of service you choose depends upon the requirements of the activity. For example, if an activity requires that managers enter data about their employees, you can create a Human service with a Coach. If an activity requires integration with an external system, such as a database, you can create an Integration service.	Building services
Nested process	You can implement an activity using a nested BPD. Nested BPDs provide a way to encapsulate logically related steps within a process while retaining the high-level view of the parent process.	Using nested processes
Javascript	You can write JavaScript to implement a step in your process and embed that script in an activity.	Using embedded JavaScript
External activity	You can implement an activity using an external application, such as an Eclipse RCP or Microsoft® .NET application.	Using external activities

When the implementation that you want to use has been created, such as a service, you can select it as outlined in the following steps:

1. Click to select the activity that you want in the BPD diagram.
2. Click the Implementation option in the properties.
3. Under Implementation, select the option that you want from the drop-down list.

For example, choose **Lombardi Service** if the implementation for the current activity is a Human Service with a Coach. (The preceding table describes each available option.)

4. Click the **Select** button to choose the implementation from the library.

If the implementation does not yet exist, click the **New** button to create it. (See the previous table to access the instructions for creating new implementations.)



If you are following the procedures in [Basic modeling tasks](#) to build a sample BPD, some of the activities in the sample are implemented in the linked procedures in the preceding table. For example, the Enter Expenses activity is implemented with an underlying Human service and Coach.

If you choose Lombardi Service or External Activity for your implementation option, you need to specify additional Implementation properties as outlined in the following steps.

5. (For services only) click the Clean Slate checkbox if you want to clear the runtime execution state of an activity after it is complete. By default, this option is disabled. Enable this option only when you do

not want to store execution data (such as variable values) for viewing after the process has finished execution.

- In the Task Header section, specify the following properties:

Subject	Type a descriptive subject for the task that is generated in Lombardi Process Portal when you run the BPD. You can also use the Lombardi embedded JavaScript syntax (for example, <code><#=tw.local.mySubject#></code>) to express the subject.
Narrative	Type an optional description. You can also use the Lombardi embedded JavaScript syntax to express the narrative.



For the following properties (in the Priority Settings section) you can click the JS button for an option if you prefer to use a JavaScript expression with predefined variables to establish the priority settings.

- For the Priority field, click the drop-down list to select one of the default priority codes: Very Urgent, Urgent, Normal, Low, or Very Low.
- For the Due In field, you can enter a value in the text box and then choose Minutes, Hours, or Days from the drop-down list. (When you choose Days, you can use the text box after the drop-down list to include hours and minutes in your specification.)

You also have the option of using the variable selector next to the text box to choose an existing variable from the library. At run-time, the variable should reflect the value that you want for the time period. Be sure to select the option that you want from the drop-down list: Minutes, Hours, or Days.

- For the Schedule field, click the drop-down list to select one of the options. For example, select **24x7** if you want 24 hours a day, seven days a week to be the time period in which the resulting tasks from the current activity can be due.



You can leave the Schedule, Timezone, and Holiday Schedule fields set to `(use default)`. If you do, the work schedule specified for the BPD is used. See [Setting the work schedule for a BPD](#) for more information.

- For the Timezone field, click the drop-down list to select the time zone that you want to apply to the tasks that result from the current activity. For example, you can select **US/Pacific** for end users who work in California.
- For the Holiday Schedule field, you can leave the setting at `(use default)` as described in the preceding note or you can click the JS button if you prefer to use a JavaScript expression. Each Holiday Schedule is made up of a list of Dates.

If you choose JavaScript, you can enter either a String (or String-generated JavaScript) or JavaScript that returns a `TWHolidaySchedule` variable. If you use a String, then Lombardi looks up the Holiday Schedule by name according to those rules. If you use a `TWHolidaySchedule` variable, then Lombardi assumes that the holiday schedule is filled in appropriately. (Go to the System Data toolkit and open the `TWHolidaySchedule` variable to view its parameters.)

Adding process variables to a BPD

For each BPD that you create, you need to declare variables to capture the business data that is passed from step to step in your process. In a BPD like the sample we're building in this section, you need to

capture the ID, amount, and status of each expense that is filed for approval. With variables, the end users or systems involved in each activity have the information required to complete the step so that the process can move on to the next activity. For example, the managers involved in the Approval activity in our sample BPD cannot determine whether to approve an expense without knowing the amount.



You can also use environment variables in JavaScripts and other implementations in your BPDs. To learn how to set environment variables for each process application, see [Setting environment variables](#).

At this stage, you can add variables that you know the overall process requires to function. When you are ready to fully implement the steps of your process, the members of your development team need to have a complete understanding of the entire data model that is required. See [Managing and mapping variables](#) to learn variable implementation details like the following:

- The variable types available in Lombardi
- Variable scope and how inputs and outputs flow in Lombardi

You can add the following variables to your BPDs:

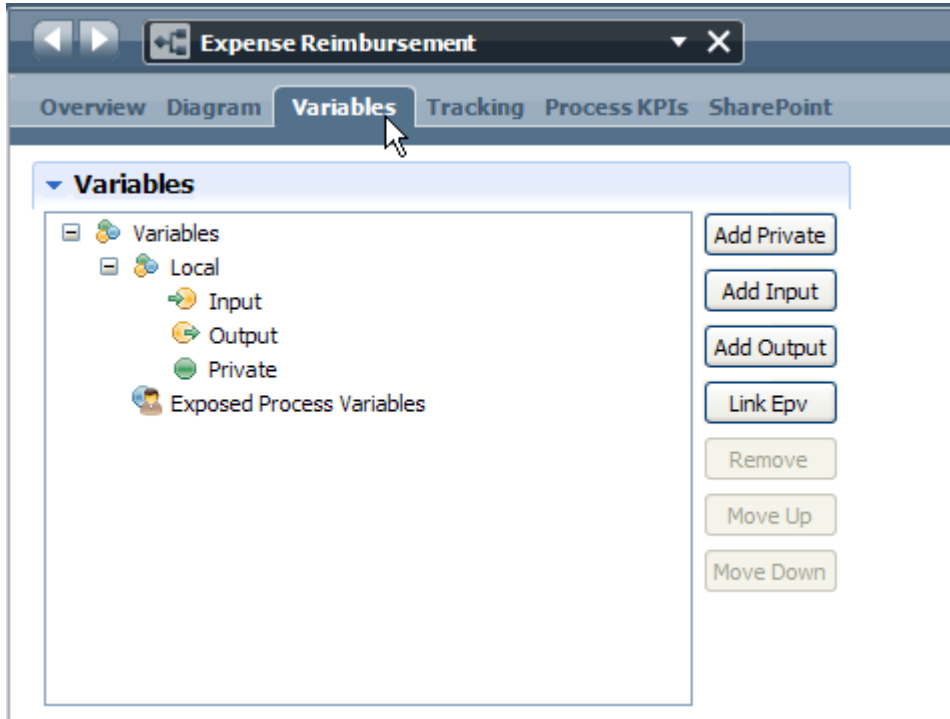
Variable	Description
Private	Private variables are local variables that are only used within the process.
Input	Input variables are mapped to values that you can pass into the current process.
Output	Output variables are mapped to values that you can pass out from the current process to a parent process.



Exposed process values (EPVs) are a special type of variable that you can create to enable end users to set or alter values while an instance of a process is running. EPVs allow end users to adjust specific variable values as constants, thereby affecting the flow of a process, task assignments, and so on. If EPVs have been created, you can link them to multiple processes and services from the Variables tab in the Designer. For more information see [Creating exposed process values \(EPVs\)](#).

To declare variables for the sample Expense Reimbursement BPD:

1. Click the **Variables** tab in the Designer as shown in the following example:



The Expense Reimbursement process does not require inputs from another process and will not provide outputs to any parent processes, which means that a private local variable is needed.

2. Click the **Add Private** button to create a new variable named `request` that includes a parameter for each aspect of the expense request (`id`, `date`, `amount`, and `status`).
3. In the details for the variable, enter `request` in the **Name** field.



All variable names should start with a lowercase letter, with subsequent words capitalized like so: `myVar`. Do not use underscores or spaces in variable names.

4. Select the **New** button next to **Variable Type**.

Because each step of the process requires all the information about each submitted expense request, we can create a variable that is a complex structure so that we can include a parameter for each aspect of the request. A complex structure is simply a way of grouping business data that is related to the same subject, in this case the expense request that is routed by our process.

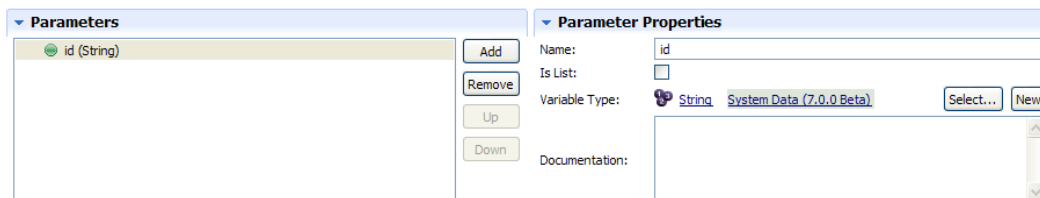


Complex structures must be initialized as described in [Initializing complex variables and lists](#).

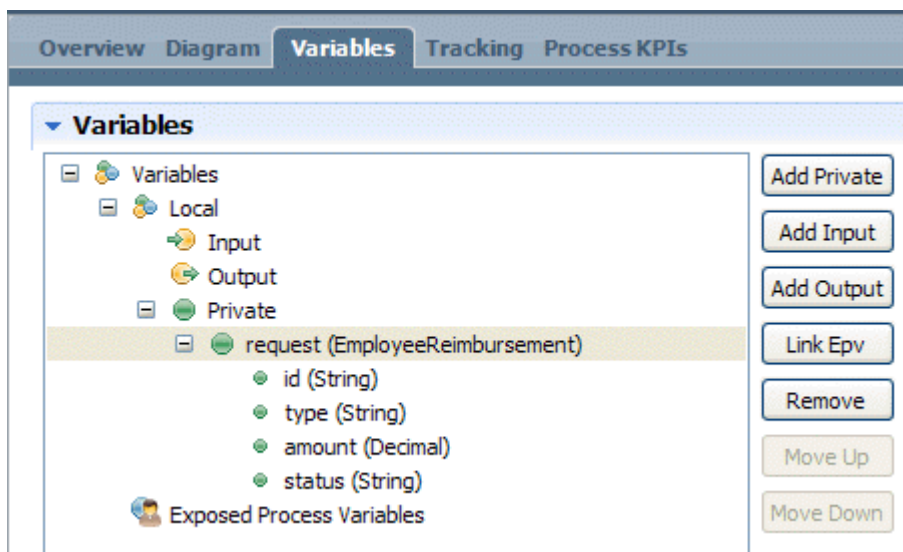
5. In the **New Variable Type** dialog, enter `EmployeeReimbursement` in the **Name** field and click the **Finish** button.

Name variable types so that each word is capitalized like so: `MyType`. This naming convention enables you to easily distinguish between the variable types that you create and the variables that you declare. Names of variable types are case sensitive. See [Creating custom variable types](#) for more information.

6. In the Variable Type editor (in the Behavior section) select **Complex Structure Type** from the Definition Type drop-down list.
7. In the Parameters section, click the **Add** button.
8. In the Parameter Properties section, replace `untitled1` in the Name field with `id`.
9. Leave the variable type set to String as shown in the following image:



10. Repeat steps 7 through 9 to add parameters for `type`, `amount`, and `status`. Set the `type` and `status` parameters to the String variable type. Set the `amount` parameter to the Decimal variable type.
11. Click **Save** in the main toolbar.
12. Go back to the Expense Request BPD and click the **Variables** tab. You can see the `request` variable and all of the parameters established for the `EmployeeReimbursement` variable type as shown in the following image:



To learn more about creating variable types and declaring variables, see [Managing and mapping variables](#).

For an example of how to pass the `request` variable as input to a Rule service, see [Building a Rule service](#). For an example of how to use the `request` variable as output from a Human service, see [Building a Human service](#).

See the following topic, [Adding events to a BPD](#), to continue to build the Expense Reimbursement sample BPD.

Adding events to a BPD

Lombardi enables you to model events that can occur at the beginning, during, or at the end of a run-time process (as opposed to activities that are performed by participants within the process). An example of an event is a message arriving from an external system. Lombardi enables you to specify the trigger type of an event and represent the event with a meaningful icon in your process diagram.

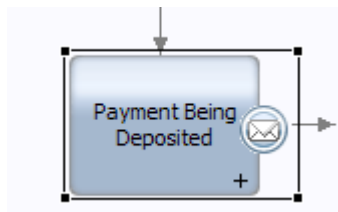
By default, when you create a Business Process Definition (BPD) in Lombardi, it includes a Start event and an End event as described in [Creating a BPD](#). Unlike other events, Start and End events are not triggered by an action. Rather, they are used to simply indicate starting and ending points within a process or service. Other events in Lombardi can be triggered by a due date passing, an exception, or a message arriving. For detailed information about Lombardi event types and their triggers, see [Modeling events](#). The trigger that you want determines the type of event you choose to implement.

Intermediate events (Timer events, Intermediate Message events, and Intermediate Exception events) can be attached to activities within your BPDs or they can be included in the process flow, connected with sequence lines. Other events are simply included in the process flow.

For the Expense Reimbursement sample BPD, we need to include an Intermediate Message Event. The event attaches to the final activity in the process (Payment Being Deposited) so that a message from the external system processing the deposit can notify the activity when the deposit is complete. When the message is received from the external system, the activity closes and the process ends.

1. In the Designer view, click the **Diagram** tab.
2. Drag an Intermediate Message Event component from the palette to the Payment Being Deposited activity.

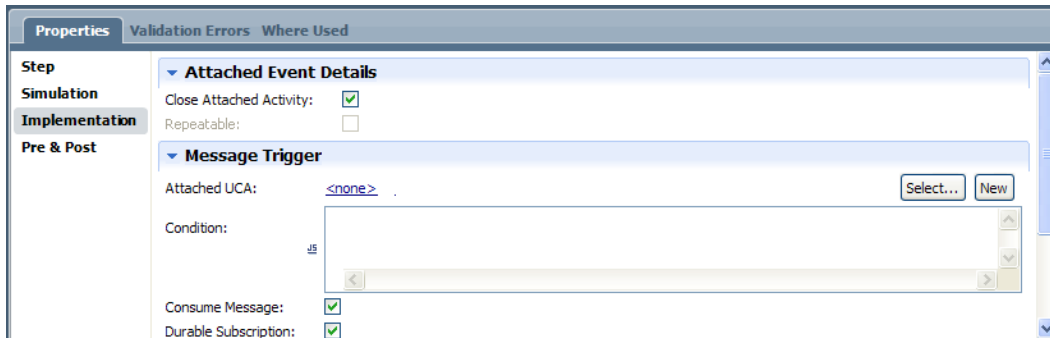
The event is anchored to the activity. To verify this, select the activity. If the activity's outline includes the event, the event is attached as shown in the following image. If not, drag the event closer to the center of the activity.



3. Click the attached message event to select it.
4. In the event properties, click the **Implementation** option.
5. In the **Attached Event Details** section, select the check box for **Close Attached Activity**.

This option directs the activity to close when the message is received. For our example process, this is how we know the process is complete.

6. In the **Message Trigger** section shown in the following image, note that you can select or create an undercover agent (UCA) to attach to the event.



Each message event must be associated with a UCA. When you run the process, the associated UCA calls its attached service to perform the required action when the event is triggered.

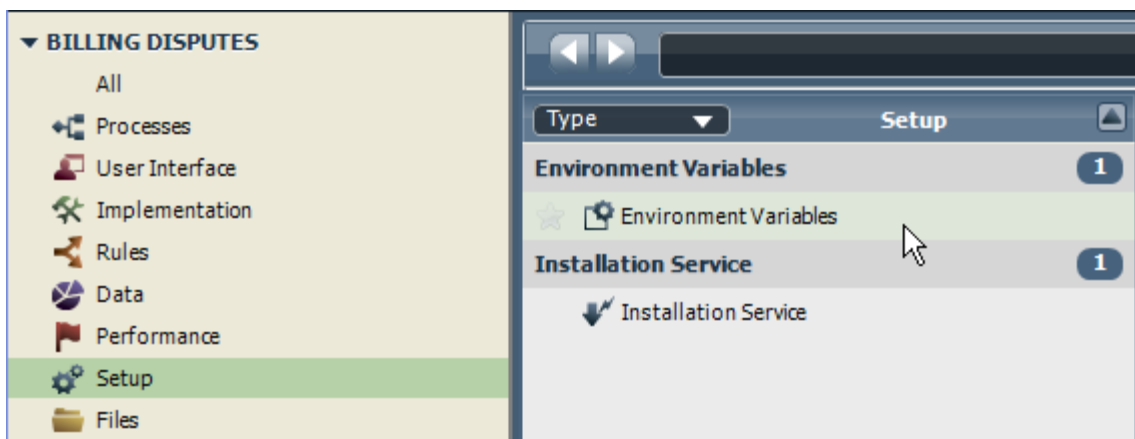
To learn more about UCAs and events, see [Understanding and using Undercover Agents](#) and [Modeling events](#).

Setting environment variables

Each process application and toolkit created in the Process Center repository includes environment variables. You should take advantage of these environment variables to ensure that your process implementations are utilizing correct values no matter which environment you deploy to or how much your environment changes at run-time. For example, suppose your process includes an implementation that requires the port number for an external application. By using an environment variable, you can set the port number for each environment in which the process will run. Plus, administrators can verify and adjust environment variable values from the Process Admin Console after a process application is installed as described in [Configuring installed snapshots](#).

To set environment variables:

1. Open the appropriate process application or toolkit in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. Click the **Setup** category and double-click **Environment Variables** as shown in the following image:



3. In the Environment Variables editor, click the **Add** button to add a new variable.
4. Click the `<NEW VARBL>` placeholder in the Key cell, type the name of the new environment variable, and press Enter.



Environment variable names should start with a lowercase letter, with words separated by periods. The following examples are all valid names: `sapconnector.port`, `sapConnector.port`, and `sap.connector.port`

- Click the Default cell for the newly entered variable, type in a value, and press Enter.

If you do not provide values for the other environments, the default value is used for all environments.

- If you know the value for the Development, Test, Staging, or Production environment, click the appropriate cell, type in the value, and press Enter.

If you do not know the appropriate value for each environment, you can leave the setting blank and an administrator can provide the correct value after installation as described in [Configuring installed snapshots](#). If you do not provide a value for an environment and an administrator does not provide a value after installation, Lombardi uses the default value.

The following image shows two environment variables set:

Key	Default	Environment:Devel...	Environment:Test	Environment:Staging	Environment:Produ...
sapconnector.port	342	333	333	333	342
sapconnector.version	4	4	4	3	3

- To use one of the preceding variables in a script in the current process application, you can simply use the variable key preceded by `tw.env`. For example, to set the value of a process variable to an environment variable in a JavaScript, you can type:

```
"tw.local.port = tw.env.sapconnector.port"
```

If the environment variable you want to use in your script is in a toolkit, you can precede the variable key with `tw.env.toolkit.[toolkit_acronym]`. So, to use the same environment variable from a toolkit with an acronym of BA, you can type:

```
"tw.local.port = tw.env.toolkit.BA.sapconnector.port"
```



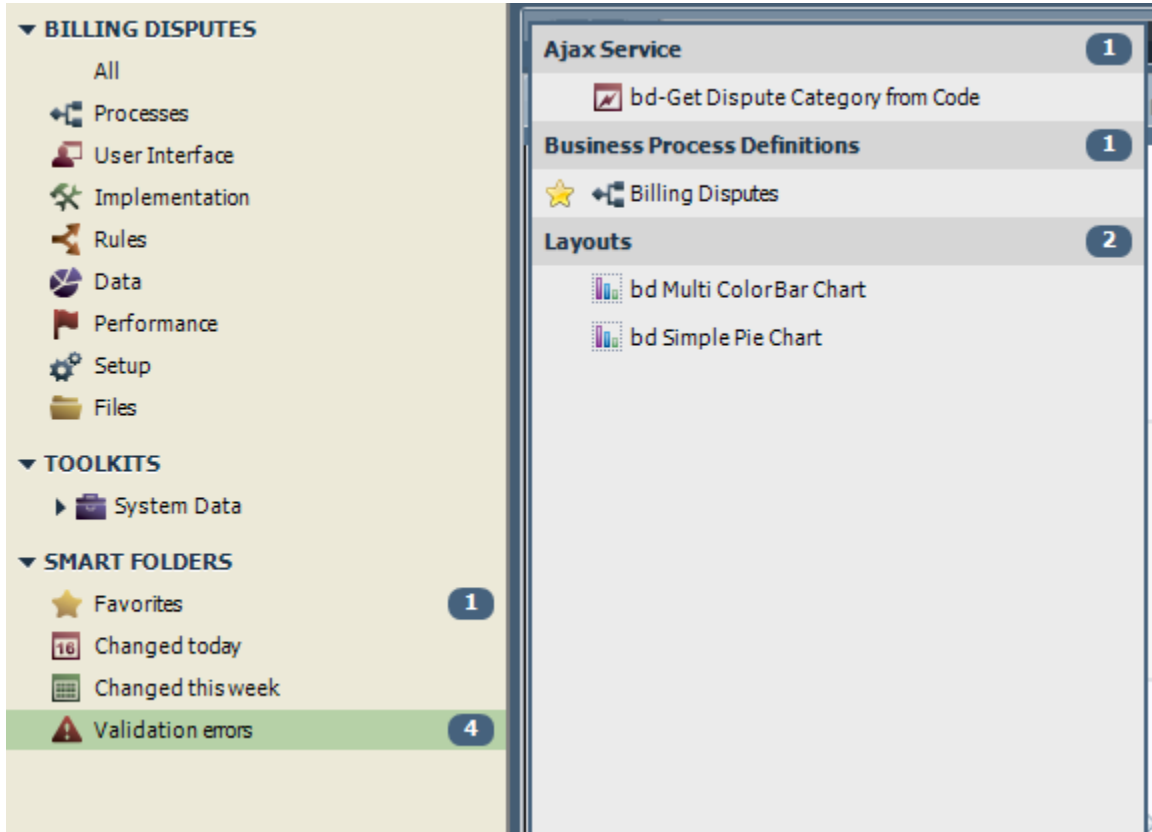
To remove an environment variable, click the Key cell and then click the **Remove** button.

Validating processes

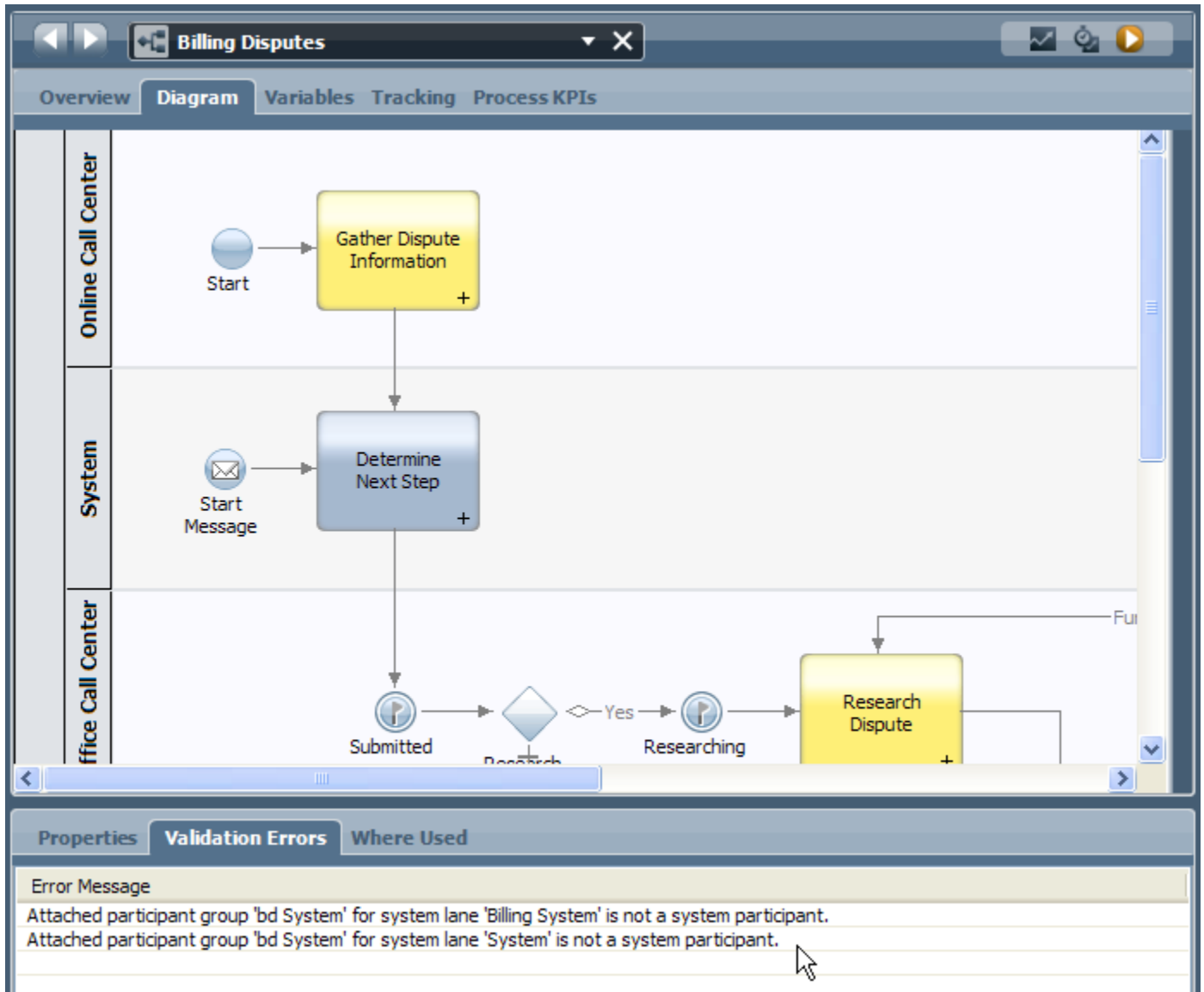
The Designer in Lombardi Authoring Environment includes validation functionality that alerts you to issues in your process applications and toolkits. Validation provides feedback about the following types of issues:

- Broken references (for example, missing implementations for activities)
- Problems with parameter mappings
- Duplicate names and other naming violations

If Lombardi Designer detects issues, the Validation errors category in the library displays the number of errors detected. You can click the category to display a list of issues as shown in the following image:



Double-click an item in the list to open the item and then click the Validation Errors tab, which lists each error for the selected item as shown in the following example:



Configuring BPDs

When building business process definitions (BPDs) in the Designer in Lombardi Authoring Environment, you need to complete several configuration tasks to ensure that run-time instances of the process meet the requirements of everyone in your organization. Tasks to complete include the following:

Task	Description	See...
Expose the BPD	During BPD creation and design, you should establish who can start the process and perform other tasks in the Process Portal.	Exposing BPDs
Set the work schedule for the BPD	Lombardi enables you to establish a schedule so that the tasks generated by a BPD have due dates that reflect the work schedule of the end users who receive them.	Setting the work schedule for a BPD

Task	Description	See...
Set the name and due date for BPD instances	When you run a BPD, Lombardi establishes a default name and due date for each instance. You can change these default settings for each BPD.	Setting the name and due date for BPD instances
Create exposed process values (EPVs)	Lombardi enables you to expose specific process variables so that end users can adjust the values of those variables to affect the flow of a process at run time.	Creating exposed process values (EPVs)
Track data for reports	Enable tracking for variables, add tracking points, and complete other configuration tasks required to collect necessary data for Lombardi reports.	Creating and configuring reports
Set simulation properties and track data for analyses using the Optimizer	Configure your BPDs and the activities within them for simulations and historical analyses using the Optimizer.	Simulating and optimizing processes

Exposing BPDs

You need to expose a BPD to particular participant groups to establish who can:

- Start instances of the process in Lombardi Process Portal
 - View data for instances of the process in reports in Lombardi Process Portal
1. In the Designer in Lombardi Authoring Environment, open the BPD that you want to expose.
 2. Click the Overview tab.
 3. In the Exposing section, enable the exposure setting(s) that you want:

Expose to start	Click the Select button to choose the participant group whose members can start instances of this process in Lombardi Process Portal. Members of the selected participant group can start instances of the process using the New icon in the Process Portal Inbox. See <i>Starting a new process in Lombardi Process Portal User Guide</i> or online help for more information. To create a new participant group to use for this exposure setting, click the New button and see Creating a participant group for instructions.
Expose business data	Click the Select button to choose the participant group whose members can create ad-hoc reports that include data for this process in Lombardi Process Portal. Members of the selected participant group can select this process for their ad-hoc report as described in <i>Creating ad-hoc reports in Lombardi Process Portal User Guide</i> or online help. To create a new participant group to use for this exposure setting, click the New button and see Creating a participant group for instructions.
Expose performance metrics	Click the Select button to choose the participant group whose members can view data for this process in the Process Performance scoreboard in Lombardi Process Portal. For more information about the Process Performance scoreboard, see <i>Working with reports in Lombardi Process Portal User Guide</i> or online help. To create a new participant group to use for this exposure setting, click the New button and see Creating a participant group for instructions.



To remove an assigned participant group, click the X icon next to the exposure setting.

4. Save your changes.



Exposed BPDs and data from the current working version (tip) are always available in Lombardi Process Portal. However, If you want exposed BPDs and data from a particular snapshot to be available in Lombardi Process Portal while under development on the Process Center Server, you need to activate the snapshot (version) that you want. Anyone with administrative access to the process application can activate snapshots. For instructions, see [Activating snapshots in the Process Center Console](#). When you install snapshots of process applications on Process Servers in other environments, such as test and production environments, those snapshots are active by default. You can deactivate installed snapshots, change participant group members, and perform other configuration tasks as described in [Configuring installed snapshots](#).

Setting the work schedule for a BPD

The settings that you specify for the BPD work schedule enables Lombardi to calculate appropriate timeframes for instance and task due dates, and any other dates required to run instances of the process.

Be aware of the following when establishing the work schedule for a BPD:

- The default work schedule for BPDs is established in the `[Lombardi_home]/[process-server|process-center]/config/system/99Local.xml` configuration file. If you do not set the work schedule for a BPD as described in the following steps, or if you leave the settings at (use default), date calculations for instances of the BPD are based on the `<default-work-schedule>` in the configuration file.
- For each of the following settings, you can click the **JS** button if you prefer to use a JavaScript expression with predefined variables to establish the BPD work schedule.
- If you choose to use JavaScript for the following settings, you can enter either a String (or String-generated JavaScript) or JavaScript that returns a `TWTimeSchedule` or `TWHolidaySchedule` variable. If you use a String, then Lombardi looks up the schedule by name according to those rules. If you use one of the `TWSchedule` variables, then Lombardi assumes that the schedule is filled in appropriately. (Go to the System Data toolkit and open the `TWTimeSchedule` or `TWHolidaySchedule` variable to view its parameters.)
- You can write a Lombardi service to dynamically set the overall work schedule for a BPD and store the entire work schedule in the `TWWorkSchedule` variable. The `TWWorkSchedule` variable includes parameters for `timeSchedule`, `timeZone`, and `holidaySchedule`. Go to the System Data toolkit and open the `TWWorkSchedule` variable to view its parameters.

1. In the Designer in Lombardi Authoring Environment, open the BPD for which you want to establish a work schedule.
2. Click the Overview tab.
3. In the Work Schedule section, click the drop-down list for the `Time Schedule` field to select a timeframe in which work for running instances of this BPD can be completed.

For example, select **9AM-5PM M-F** if you want due dates for instances of this process to be calculated using standard business hours.

4. Click the drop-down list for the `Timezone` field to select the time zone that you want to apply to running instances of the current BPD. All standard time zones are available.

5. For the Holiday Schedule field, you can leave the setting at (use default) as described in the preceding note or you can click the JS button if you prefer to use a JavaScript expression. Each Holiday Schedule is made up of a list of Dates.
6. Save your changes.

Setting the name and due date for BPD instances

By default, the name for each instance of a process is the BPD name plus the instance ID. You can change the name as described in the following procedure. By default, each instance of a BPD is due 14 days after it is started. You can also change the due date for a BPD if you want to establish a different deadline for running instances of the process. The name and due date for each process instance are displayed in the Inbox in Lombardi Process Portal, letting end users know when all tasks generated by the process should be finished.

To set the name and due date for a BPD:

1. In the Designer in Lombardi Authoring Environment, open the BPD for which you want to set the name or due date.
2. Click the Overview tab.
3. In the Advanced section, change the text in the **Instance name** field if you want the name of each instance to be something other than the BPD name. Be sure to retain the quotes around the text.

By default, the instance ID is appended to the instance name, using the `tw.system.process.instanceId` variable. You can remove this variable or use the variable picker to choose additional variables.

4. For the **Due in** field, you can enter a value in the text box and then choose Minutes, Hours, or Days from the drop-down list. (When you choose Days, you can use the text box after the drop-down list to include hours and minutes in your specification.)

You also have the option of using the variable selector next to the text box to choose an existing variable from the library. At run-time, the variable should reflect the value that you want for the time period. Be sure to select the option that you want from the drop-down list: Minutes, Hours, or Days.

5. Save your changes. When you run the BPD, the due date and name displayed in the Inspector and in Lombardi Process Portal should reflect your changes.

Advanced modeling tasks

After creating a BPD in the Designer in Lombardi Authoring Environment that includes necessary activities in the order to be executed, you need to develop the underlying implementations for those activities and also possibly create additional advanced components.

The options for implementing activities include the following:

Implementation option	Description	See...
Service	You can create the following types of services: Integration, Human, Ajax, Rule, and General System. The type of service you choose depends upon the requirements of the activity. For example, if an activity requires that managers enter data about their employees, you can create a Human	Building services

Implementation option	Description	See...
	service with a Coach. If an activity requires integration with an external system, such as a database, you can create an Integration service.	
Nested processes	You can implement an activity using a nested BPD. Nested BPDs provide a way to encapsulate logically related steps within a process while retaining the high-level view of the parent process.	Using nested processes
Javascript	You can write JavaScript to implement a step in your process and embed that script in an activity.	Using embedded JavaScript
External activity	You can implement an activity using an external application, such as an Eclipse RCP or Microsoft .NET application.	Using external activities


You also need to understand and possibly perform the following tasks when developing the implementation for your process steps:

Task	Description	See...
Modeling events	Start and end events indicate starting and ending points within a process or service. Other events in Lombardi can be triggered by a due date passing, an exception, or a incoming message from an external system.	Modeling events
Integrating with external systems	You can configure Lombardi processes to communicate with an external system to retrieve, update, or insert data. And, external applications can call into Lombardi to initiate services.	Integrating with other systems
Declaring, mapping, and managing variables	Variables capture the business data that is passed from step to step in a process. You should determine how to best represent this business data in Lombardi, and develop an understanding of how to effectively pass that data from one step to the next in your process.	Managing and mapping variables
Handling errors	Design processes and services to catch and handle exceptions generated by integrations and other more complex implementations.	Handling exceptions
Creating loops (simple and multi-instance)		Creating loops

Building services

In Lombardi, you can use services to implement the activities in a business process definition (BPD). When a BPD starts and the steps (activities) are invoked, services can perform the required functions. The type of service that you choose to create depends upon the requirements of the activity. For example, If an activity requires integration with an external system, such as a database, you can create an Integration service. If an activity requires that call center personnel enter data about customer requests, you can create a Human service with a Coach.

The following table describes the types of services available in Lombardi:

Service type	Description	See...
Rule service	Use a Rule service when you want a condition to determine the implementation invoked. For example, when a certain condition evaluates to true, Lombardi implements the JavaScript expression that you provide. Rule services cannot include Java or Web Service integrations directly. You can call a Rule service from any other type of service and a Rule service can call other nested services.	Building a Rule service
Human service	Use a Human service when you want to create an interactive service. A Human service is the only type of service that can contain Coaches and postpones. Human services generate tasks in Lombardi Process Portal.  A Human service is the only type of service that should reside in a non-system lane. And, a Human service is the only type of service that can call other nested Human services.	Building a Human service
Ajax service	Use an Ajax service when you want to include a control in a Coach to implement dynamic data selection such as automatically populating drop-down lists and automatically completing edit boxes. An Ajax service can pull data dynamically from a connected data source, such as a database. You cannot call an Ajax service from other types of services, but an Ajax service can call other nested services.	Building an Ajax service
Integration service	Use an Integration service when you want to integrate with an external system. An Integration service is the only type of service that can contain a Java or Web Service integration. You can call an Integration service from any other type of service and an Integration service can call other nested services.	Building an Integration service
General System service	Use a General System service when you need to coordinate other nested services or you need to manipulate variable data. For example, if you need to implement data transformations or generate HTML for a coach, you can use a General System service. General System services cannot include Java or Web Service integrations directly. You can call a General System service from any other type of service and a General System service can call other nested services.	Building a General System service
























To learn about other implementation options for activities, see [Advanced modeling tasks](#).

Understanding service components

Developers and analysts can build services in the Designer in Lombardi Authoring Environment. To learn more about the Designer interface, see [Using the Designer in Lombardi Authoring Environment](#).

When developing a service diagram in the Designer in Lombardi Authoring Environment, the following tools and components are available from the palette. Not all components are available for each type of service.

Component icon	Available with...	Description
 Selection Tool	All service types	Enables you to select and move components on the diagram.
 Sequence Flow	All service types	Enables you to connect service components to establish the order in which the steps in the service occur.
 Web Service Integration	Integration service only	Use to execute an external Web service. This component enables you to supply a WSDL URI and then use any of the available services.
 Java Integration	Integration service only	Use to call methods from a Java class. You can use the methods to complete tasks like reading or writing files or sending SMTP mail.
 Coach	Human service only	Use to create the interfaces for your Human services. Coaches enable you to easily add the fields, buttons, and other controls to enable end users to participate in a business process. See Building Coaches for more information.
 Server Script	All service types	Use when you want to write JavaScript to execute on the Process Server in the service context. The Server Script component is useful for parsing through variables and executing programmatic commands.
 Rule Script	Rule service only	Use to build the conditions for your Rule services.
 Server Scriptlet	All service types	Use to bind blocks of formatted text (for example, HTML, XML, or XSLT) directly to a service variable. This eliminates the need to store large blocks of text in default values for variables.
 Modify Task	Human service only	Use to change the priority, due date, status, or other aspects of a task. For example, if you want the status of a task to change to Closed each time a user completes a task, you can use this component to properly set the status and move the task into each user's Closed folder in Lombardi Process Portal.
 Postpone Task	Human service only	Use to halt processing without changing the status of a task.
 Decision Gateway	All service types	Use to model a point in the process execution where <i>only one</i> of several paths can be followed, depending on a condition.
 End Event	All service types	Use to end service execution. For services that contain multiple paths, each path requires its own end event.  An end event is automatically included each time you create a service.
 Note	All service types	Use to add information about the overall service or each step in the service to the diagram. Adding notes helps other developers understand your design.

Component icon	Available with...	Description
 Throw Exception	All service types	Use to purposely throw an error and end processing. You might, for example, use a Throw Exception component if you return too many rows from a database (over a limit that is normal and would bog down the server).
 Invoke UCA	All service types	Use to invoke an Undercover Agent (UCA) from your service.
 Catch Exception	All service types	Use to listen for exceptions from the service component to which it is attached.
 Intermediate Tracking Event	All service types	Use to indicate a point in a service at which you want Lombardi to capture the run-time data for reporting purposes. For examples of how tracking events are implemented, see Creating a basic custom report and Creating a more advanced custom report .
 Nested Service	All service types	Use to incorporate other services in your current service. Nested services are generally defined to perform specific, repeatable functions such as exception handling routines, integration with outside systems, or data manipulation. Nested services are often used in multiple process applications and likely reside in a toolkit.  Human and Ajax services cannot be nested.
 Send Alert	All service types	Use to send task-related alerts to Lombardi Process Portal.

Creating a service

To create services, you must have access to a process application or toolkit in the Process Center repository. Access to process applications and toolkits is controlled by users who have administrative rights to the repository. For more information, see [Managing access to the Process Center repository](#).

1. Start Lombardi Authoring Environment and open the appropriate process application in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. In the Designer view, select the options as instructed in the following table according to the type of service you want:

Service type	Select the plus sign next to...	And then select this component from the list...
Human service	User Interface	Human Service
Ajax service	User Interface	Ajax Service
Rule service	Rules	Rule Service
Integration service	Implementation	Integration Service
General System service	Implementation	General System Service

3. In the New Service dialog, enter a name for the service and click **Finish**.

- Lombardi Designer displays the diagram of the service with the default Start Event and End Event components.
- Continue to build your service as explained in the following topics:

Service type	See...
Human service	Building a Human service
Ajax service	Building an Ajax service
Rule service	Building a Rule service
Integration service	Building an Integration service
General system service	Building a General System service



You can also use Human services to customize Lombardi consoles and fulfill other requirements as described in [Exposing a Human service](#).

Exposing a Human service

In addition to implementing the activities in a BPD, the Human services that you create in the Designer in Lombardi Authoring Environment can also be used to customize the Process Admin Console or to create a custom project page for Lombardi Process Portal. The exposure settings for a service determine its purpose as described in the following procedure.

- In the Designer in Lombardi Authoring Environment, open the Human service that you want to expose.
- Click the Overview tab.
- In the Exposing section, click the **Select** button next to the `Exposed to` field to choose the participant group whose members can view and use the exposed service.

To create a new participant group to use for this exposure setting, click the **New** button and see [Creating a participant group](#) for instructions.

To remove an assigned participant group, click the X icon next to the `Expose to` field.

- In the Exposing section, click the drop-down list next to the `Exposed as` field and select one of the available options:

Not Exposed	This is the default option and is the setting that you should use for services that implement the activities within a BPD. When this option is selected, the <code>Exposed to</code> setting is not used.
Administration Service	Makes the service available (to members of the selected participant group) as a separate page in the Process Admin Console in the Server Admin capabilities. A new category is added to the menu and that category has the same name as the process application that contains the service. The name of the individual page in the new category matches the service name.
Startable Service	Enables members of the selected participant group to start the service using the New option in the Process Portal Inbox. See <i>Starting a new process in Lombardi Process Portal User Guide</i> or online help for more information.
Project Page	Makes the service available (to members of the selected participant group) in Lombardi Process Portal as a custom project page under the My Projects category. The custom project page has the same name as the service that you expose.

URL	<p>Makes the service available from a URL. For example, if you expose a service named MyService, you can access it from the following URL (using the name of the host on which Lombardi Process Center Server or Process Server is installed and the port designated for the server during Lombardi installation; as well as the acronym for the process application in which the service resides):</p> <pre>http://[host_name]:[port]/teamworks/executeServiceByName ?processApp=[acronym]&serviceName=MyService</pre>
-----	---



If you expose a Human service as an Administration Service, Startable Service, or Project Page, it is also exposed as a URL to members of the selected participant group.

5. Save your changes.
6. Exposed services from the current working version (tip) are always available in Lombardi portals and consoles. However, If you want exposed services from a particular snapshot to be available in Lombardi portals and consoles while under development on the Process Center Server, you need to activate the snapshot (version) that you want. Anyone with administrative access to the process application can activate snapshots. For instructions, see [Activating snapshots in the Process Center Console](#). When you install snapshots of process applications on Process Servers in other environments, such as test and production environments, those snapshots are active by default. You can deactivate installed snapshots, change participant group members, and perform other configuration tasks as described in [Configuring installed snapshots](#).
7. Start the appropriate interface to ensure the service is exposed as expected.

Building a Rule service

Build a Rule service when the actions that should take place in your process depend upon one or more conditions. A Rule service enables you to make variable assignments based on potential values of other variables. For example, if an employee holds the position of Director and submits a meal expense for more than \$250, then you could set a variable such as `approvalRequired` to `true`. With a Rule service, you can set several variables to different values when conditions that you express as rules are met. The actions that take place when a condition evaluates to true are defined using a JavaScript expression, which provides a great deal of flexibility in your implementation.

Rules services are often lower-level, nested services that handle the background tasks necessary to the successful completion of the overall process.

When building Rule services, be aware of the following:

- Each row in the Rule Conditions table represents a Boolean condition that will evaluate to true or false at run time. The condition is only evaluated true if the values of all the associated variables produce matches with the provided values.
- The – value in a variable field indicates that any variable value is considered a match.
- When a rule evaluates to true, the JavaScript expression that you provide as the action is executed. This expression may contain any valid JavaScript.
- A rule only executes the JavaScript expression once per a rule, using the JavaScript expression associated with the first condition that evaluates to true.

When building Rule services, you should:

- Build your rule hierarchy so that rule conditions are ordered from most complex to least complex.
- Create a final condition that is a catch-all rule. Doing so is necessary if you cannot guarantee that the variable that you want to modify in the rule will be set prior to the running process triggering the Rule service.
- Consider encapsulating rules in a single-function Rule service, allowing the service to be available to any other part of the process application that needs the same rule logic.

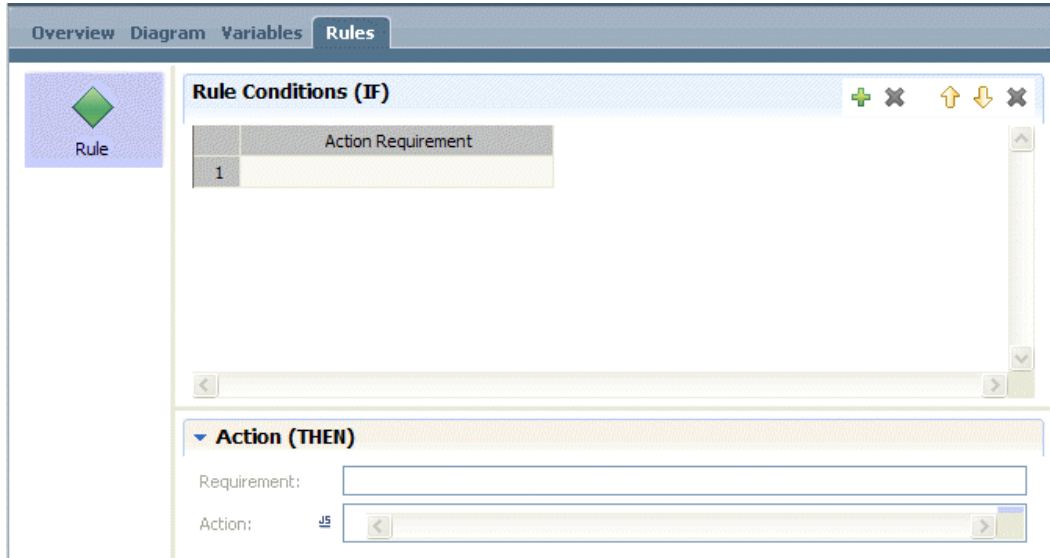
The following steps describe how to build a sample Rule service. The service in the sample is used to determine whether approval is required for certain employee expenses and it is a single-function Rule service that can be called from any other service.

1. Open the process application that contains the BPD that you created by following the steps in [Basic modeling tasks](#).
2. Create the appropriate service type as described in [Creating a service](#).
3. Drag a Rule Script component from the palette to the service diagram.
4. Click the **Variables** tab.
5. Click the **Add Input** button so that the service can receive the variables to act upon. (In this sample, we input the private variable, `request`, originally created in [Adding process variables to a BPD](#).)
6. Replace `Untitled1` in the **Name** field with `request`.
7. Click the **Select** button next to Variable Type and select the `EmployeeReimbursement` type from the list of types displayed. (The `EmployeeReimbursement` variable type is available only if you first implement the steps in [Adding process variables to a BPD](#).)



If you use the Activity Wizard to create a Rule service, you can choose existing variables to add as input and output. You should use the Activity Wizard when you start with an existing activity and want to quickly create a service to implement the activity. To access the wizard, right-click an activity and select **Activity Wizard** from the list of options.

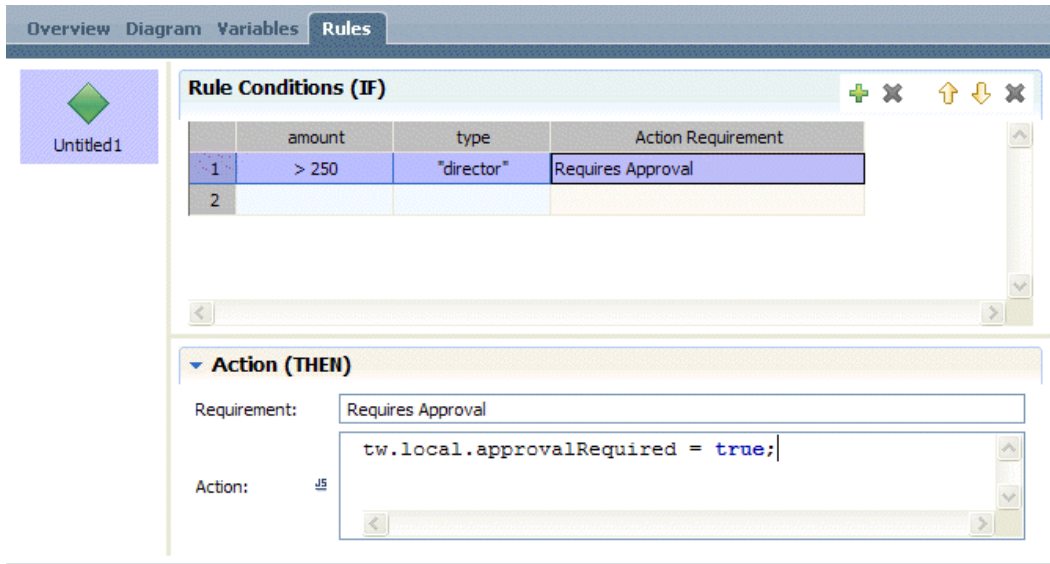
8. Click the **Add Private** button.
9. Replace `Untitled1` in the **Name** field with `approvalRequired`.
10. Click the **Select** button next to Variable Type and select the `Boolean` type from the list of types displayed. (The `Boolean` variable type is included in the Lombardi System Data toolkit. The System Data toolkit is included in each process application by default.)
11. Click the **Rules** tab to open the rules editor as shown in the following image:



12. In the rules editor, click the plus sign to add a variable (column) to the first rule (row).
13. From the variables displayed, pick the amount variable from the request structure.
14. Type >250 as the value.
15. In the rules editor, click the plus sign again. Make sure the first rule (row) is selected because you want to add another variable (column) to this rule.
16. From the variables displayed, pick the type variable from the request structure.
17. Type "director" as the value.
18. In the **Action Requirement** field for the first rule (row), type Requires Approval.
19. In the rules editor, click the **Action** section to expand it.
20. For the Requires Approval requirement, enter the following JavaScript for the Action:

```
tw.local.approvalRequired = true;
```

The rules editor should include the rule shown in the following image:



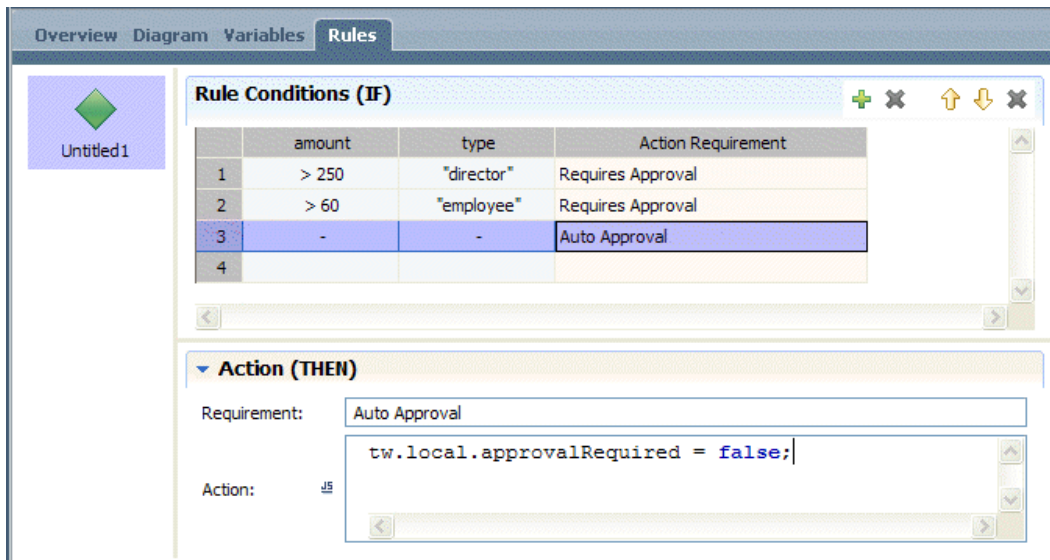
21. In the rules editor, click the second row to select it. Create a new rule so that expenses above \$60 for employees requires approval.
22. In the rules editor, click the third row to select it. Create your catch-all condition by typing - for both the amount and type.

The - value in a variable field indicates that any variable value is considered a match.

23. In the **Action Requirement** field for the third rule (row), type Auto Approval.
24. In the **Action** section, enter the following JavaScript for the Auto Approval action:

```
tw.local.approvalRequired = false;
```

The rules editor should include the rules shown in the following image:





25. Click the **Diagram** tab.

26. Use the Sequence Flow tool to connect the Rule Script component and the Start and End events.
27. Name the Rule Script component and save your work.

You can now nest this Rule service in any other service within your process application that requires this logic. Be sure to adjust the input and output variables as required for each implementation.

Using the rules editor

The toolbar options in the rules editor perform the following functions:

	Add a new variable (column) or remove the selected variable (column) from the rule.
	Move the selected rule (row) up or down in the rules table or remove the selected rule (row) from the table.

Specifying variable values in the rules editor

The following samples demonstrate how to specify the value of a variable when using the rules editor:

Sample	Description
"ok"	Matches the exact string ok (no quotes)
1.4	Matches the exact number 1.4
{"A", "B" }	Matches either of the strings A or B
!{"A", "B" }	Matches anything except the strings A or B
1..5	Matches any number between 1 and 5 (inclusive)
>3	Matches any number greater than 3
<3	Matches any number less than 3
>=3	Matches any number greater than or equal to 3
<=3	Matches any number less than or equal to 3
{1, 3, 5}	Matches 1, 3, or 5
{1, 3, 5..10}	Matches 1, 3, or any number between 5 and 10 (inclusive)
!{1, 3, 5..10}	Matches any number except 1, 3, or a number between 5 and 10 (inclusive)
true	Matches the Boolean value true
false	Matches the Boolean value false

Building a Human service

Build a Human service when you want a step in your BPD to create an interactive task that process participants can perform in a Web-based user interface. When you build Human services, you include Coaches, which are the web-based forms that provide process-related data to end users as well as collect input from those users. Coaches enable you to easily add standard fields and controls such as radio buttons, drop-down menus, and so on.

If you add an activity to a non-system lane in a BPD, the activity is initially implemented using the default human service. You can double-click an activity in a non-system lane to open the default human service. By examining the service components and running the default service in the Inspector, you can get an idea of how Human services work and how Coaches are used to display and collect data from process participants.

The following steps describe how to build a sample Human service. The service in the sample enables employees to enter expenses for the Expense Reimbursement BPD that you can create by following the tasks outlined in [Basic modeling tasks](#).



In the following procedure, we use the Activity Wizard to create a Human service. You can also create a Human service from scratch as described in [Creating a service](#).

1. Starting with the Expense Reimbursement BPD, right-click the Enter Expenses activity and select **Activity Wizard** from the list of options.
2. In the Activity Wizard - Setup Activity dialog, make the following selections:

Service Type	Human Service (Service with human interaction)
Service Selection	Create a New Service

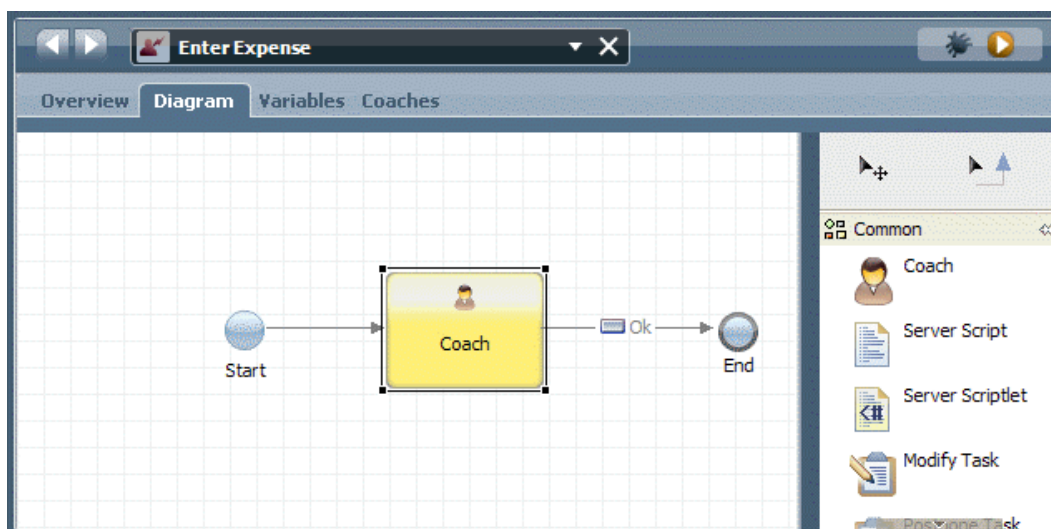
And then type a name for the new service in the **New Service Name** field. (The name for this sample service is `Enter Expense`.)

3. In the Activity Wizard - Setup Activity dialog, click the **Next** button.
4. In the Activity Wizard - Parameters dialog, choose the existing process variables to use as input and output for this new service.

If you implemented the steps in [Adding process variables to a BPD](#), you can see the private variable named `request`. For this sample, click true in the Input field to change the setting to **false** and leave the Output field set to **true**. This enables us to collect the data for the expense using this new service and then output those values for the subsequent process steps to act upon.

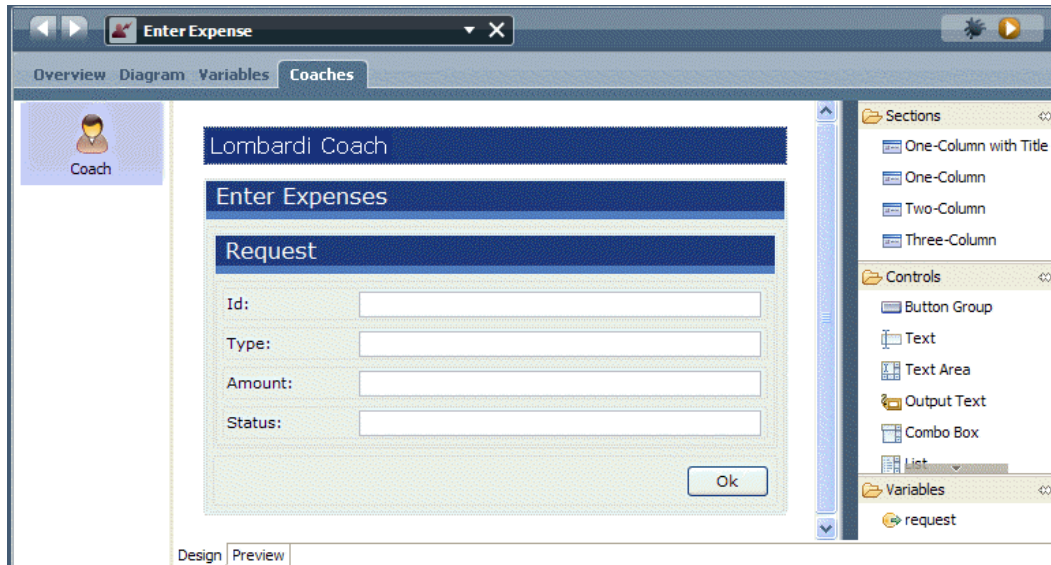
Click the **Finish** button. The new service is created and attached to the activity. The new service includes a single Coach.

5. Double-click the activity for which you created the new service using the Activity Wizard. (In this sample, the Enter Expenses activity.)
6. The new service opens in the Designer and you can see the diagram as shown in the following image:



7. Click the **Coaches** tab and then click the listed Coach component.

Because we used the Activity Wizard, the Coach includes a form element for each of the parameters in the `request` structure as shown in the following image:



The Coach designer is where you customize the Coach layout and create or edit the bindings between inputs and outputs. Notice that when the Coach designer is open, the Palette view shows all the elements—Sections and Controls—that you can use in a Coach. (Hover over a control to view a brief description. See [Building Coaches](#) for more information about the Coach designer.)

8. In the Coach designer, right-click the Status control (input text field) and select **Delete** from the list of options. The status of a request is not data that we need to collect from employees, but is a value set later after a request is further processed and so it can be removed.
9. In the Coach designer, click the Id control (input text field).

In the properties, you can see the label for the field is `Id:` to match the parameter in the `request` variable. Change the label to `Employee ID:` so that employees know exactly which ID to provide.

10. In the Coach designer, click the Type control (input text field).

In the properties, you can see the label for the field is `Type:` to match the parameter in the `request` variable. Change the label to `Employee type:`.

11. To enable employees to select from an existing list of employee types, in the properties for the Employee type control (input text field) click the drop-down list for Control Type and choose **Single Select**.

Select the **Presentation** option in the properties. In the Widget Style section, choose **Drop Down List** for the Widget Type option.

In the Manual Data section, click the **Add** button to add a value and associated display text for each option that you want in the drop-down list.

12. To add a Cancel button to the Coach, select the control that contains the Ok button in the Coach designer.

In the Presentation properties for the control, go to the Buttons section and click **Add**.

In the Button Details, enter `Cancel` for the label and click the **Is Cancel** checkbox.

13. Click **Save** in the main toolbar.
14. Click the **Preview** tab at the bottom of the Coach designer to view the Coach. The Preview tab shows how the Coach will appear to end users when the BPD runs.

You can also click the Run icon in the upper right to view the Coach in a Web browser.

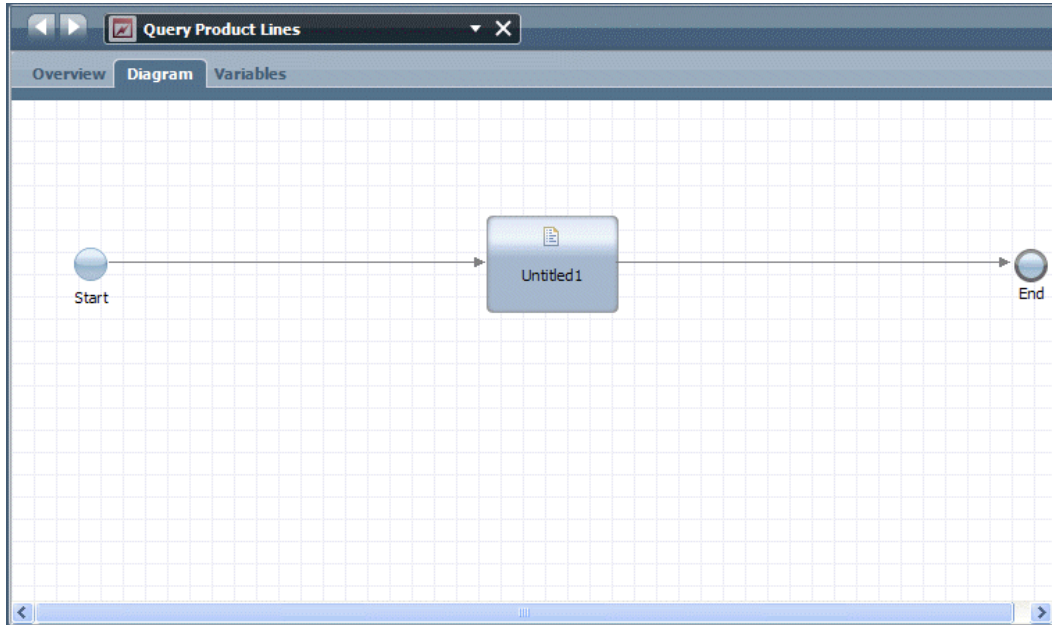
The screenshot shows a window titled "Lombardi Coach" with a "Request" section. Inside the "Request" section is a sub-dialog titled "Enter Expenses". This sub-dialog has three input fields: "Employee ID" (a text box), "Employee Type" (a dropdown menu currently showing "Employee"), and "Amount" (a text box). At the bottom right of the "Enter Expenses" dialog are two buttons: "OK" and "Cancel".

Building an Ajax service

The Ajax services that you build in Lombardi are subsequently bound to coach controls to perform functions such as automatically populating drop-down lists and enabling type-ahead capability in input fields. You can use an Ajax service to pull data dynamically from a connected data source, such as a database.

The following steps describe how to build a sample Ajax service. The service in the sample is used to populate the fields in a table control in a Lombardi Coach.

1. Create the appropriate service type as described in [Creating a service](#). Name the new service `Query Product Lines`.
2. Drag a Server Script component from the palette to the service diagram and then use sequence lines to connect the script to Start and End Events as shown in the following image.



3. Click the **Variables** tab to specify the input and output variables for the sample service.

As shown in the following image, the sample service contains an input variable named `inputVar` and an output variable named `result` of type `ProductLine`, which contains three parameters: `sku`, `description`, and `price`. (You need to create the `ProductLine` variable type and add the parameters. See [Declaring and passing variables](#) and [Creating custom variable types](#) if you need instructions.)

Be sure to enable the **Is List** checkbox for the `result` output variable as shown in the following example.



For the `inputVar` variable, ensure that the `Has Default` check box is enabled.

4. Click the Diagram tab and then click on the Server Script component in the diagram.
5. Click the Implementation option in the properties to write the script for the Ajax service using the specified variables.

The script for the sample service (shown in the following image) specifies the `sku`, `description`, and `price` for two different suppliers: `QuickServ` and `ProServ`.

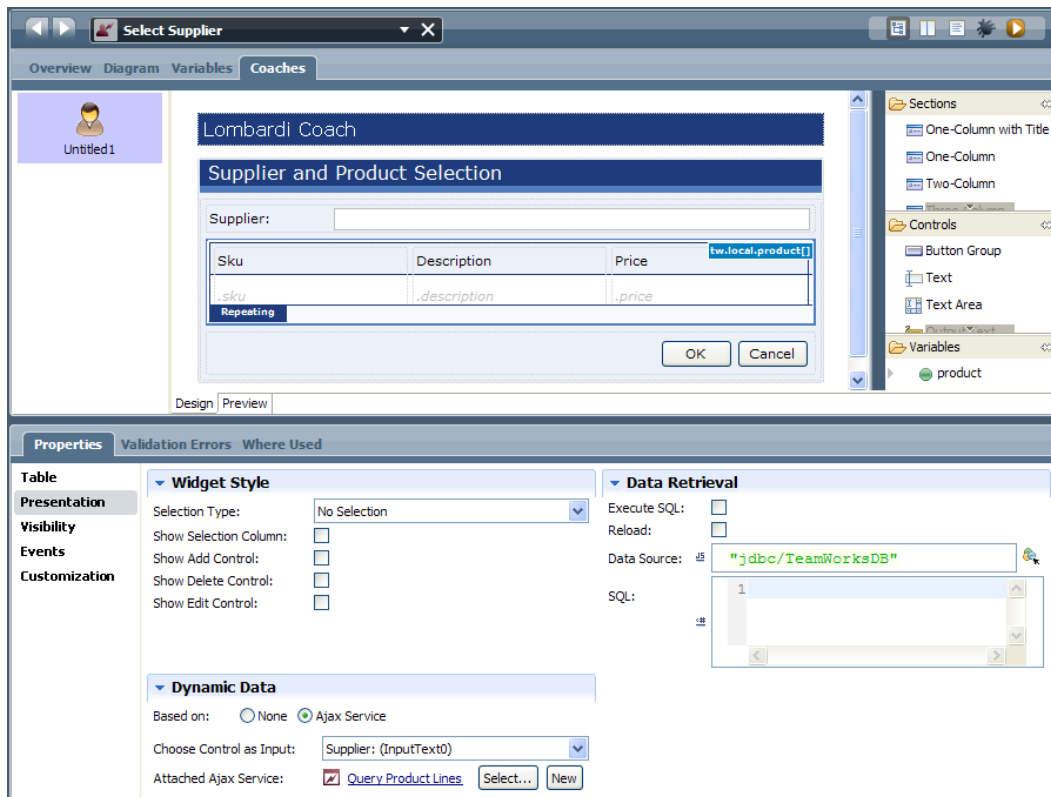
```

1 tw.local.result=new tw.object.listOf.ProductLine();
2
3 switch (tw.local.inputVar)
4 {
5 case "QuickServ":
6 tw.local.result[0] = new tw.object.ProductLine();
7 tw.local.result[0].sku = "Z34RT1GF";
8 tw.local.result[0].description = "PowerServ 1500";
9 tw.local.result[0].price = 3540;
10 tw.local.result[1] = new tw.object.ProductLine();
11 tw.local.result[1].sku = "YT76UJ8F";
12 tw.local.result[1].description = "PowerServ 1735";
13 tw.local.result[1].price = 3750;
14 break;
15 case "ProServ":
16 tw.local.result[0] = new tw.object.ProductLine();
17 tw.local.result[0].sku = "Z34RT1GF";
18 tw.local.result[0].description = "Reliant DW";
19 tw.local.result[0].price = 2039;
20 tw.local.result[1] = new tw.object.ProductLine();
21 tw.local.result[1].sku = "YT76UJ8F";
22 tw.local.result[1].description = "Reliant X1";
23 tw.local.result[1].price = 6750;
24 }

```

6. Save your work.
7. Create a Human service named `Select Supplier`.
8. Drag a Coach from the palette to the service diagram and then use sequence lines to connect the Coach to Start and End Events.
9. Click the **Variables** tab and add a private variable named `product` of type `ProductLine`. Enable the **Is List** and **Has Default** check boxes for the `product` variable.
10. Click the Coaches tab and then drag the `product` variable from the palette to the Coach editor.
This automatically creates a table control. Make sure the table control is directly beneath the default Input Text field.
11. Click the Input Text control in the Coach editor to select it and then click the Input Text option in the properties.
12. In the Common section, change the Label field to `Supplier:`.
13. Right-click the default Check box control in the Coach editor and select **Delete** to remove it. It is not necessary for the current Coach.
14. Click the table control in the Coach editor to select it.
15. Click the **Presentation** option in the properties for the table control and in the **Dynamic Data** section, click the radio button next to **Ajax Service**.

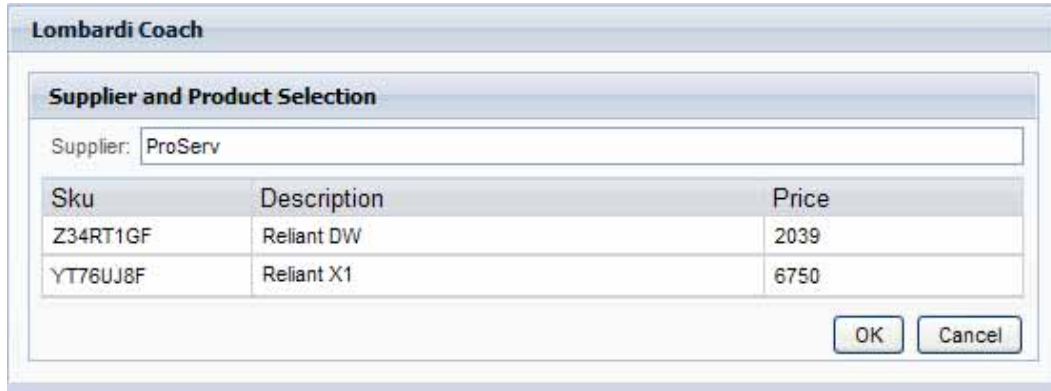
16. From the Choose Control as Input drop-down list, select `Supplier` (`Input Text`).
17. Click the **Select** button and choose the `Query Product Lines Ajax` service. The presentation properties for the control should be set as shown in the following image:



18. Click the Section control in the Coach editor and change the title from `Section Title` to `Supplier and Product Selection`.
19. Save your work.
20. Click the Run icon in the upper right corner. The Coach runs in a browser and when you provide one of the Supplier names included in the Ajax script (`QuickServ` or `ProServ`), the Coach displays the information for that supplier as shown in the following example.



If the supplier information does not immediately display, reload the browser page.



Building an Integration service

Build an Integration service when you want to integrate with an external system to complete a task. For example, you may want end users to choose from a list of products available from a common site on the internet. In that case, you can build an Integration service that calls a Web service to display the list of options. Integration services are the only services that can include Web Service Integration and Java Integration components.

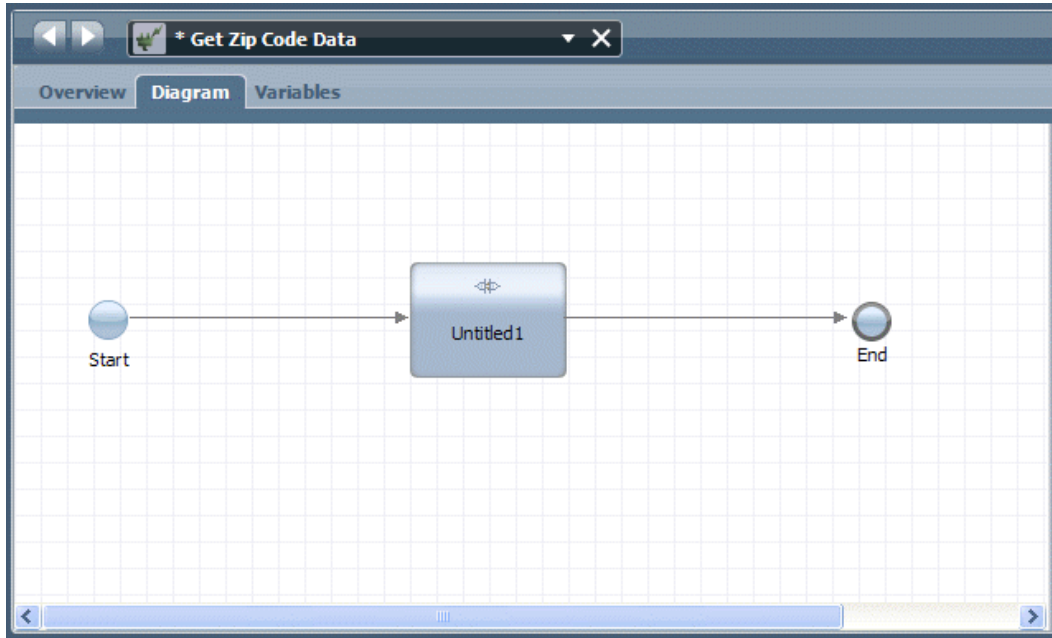
For more information about both inbound and outbound integration, see [Integrating with other systems](#).

Read the following procedures to learn how to:

- Build a sample Integration service. The sample service illustrates how you can easily use existing Web services in your Lombardi BPDs.
- Map the input and output variables for the nested Integration service.
- Create a Human service that implements the Integration service as a nested service.
- Create the Coach interfaces to collect the input and display the output for the nested Integration service.

Building the sample Integration service

1. Create the appropriate service type (Integration service) as described in [Creating a service](#) and name it `Get Zip Code Data`.
2. Drag a Web Service Integration component from the palette to the service diagram and then use sequence lines to connect the component to Start and End Events as shown in the following image:



3. Click on the Web Service Integration component in the diagram and then click the Step option in the properties.

For this sample service, type `zip Code Lookup` in the **Name** field.

4. Click the Implementation option in the properties and in the WSDL URI text box, type the URI for the WSDL that you want to use.

For this sample service, type:

`http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl`

5. Click the **Discover** button.

To use a protected WSDL, enable the check box and then provide the user name and password required to access the WSDL. For this sample, simply click the **Discover** button.

The Operations drop-down list is populated with the services that are available.

6. Choose the operation that you want to use from the drop-down list.

For this sample service, choose the `LatLonListZipCode` operation.



You can use any of the operations listed. You can enter the URI for the WSDL in your browser and review the WSDL documentation to understand the input to provide to implement and test the chosen operation.

7. Click the **Generate Types...** button.

In the Generate Types dialog, select the same operation from the previous step (`LatLonListZipCode`) and then click the **Next** button.

The Generate Types Wizard lists the types to be generated.

8. In the wizard, click the **Next** button to continue and then click the **Finish** button when type generation is complete.
9. Click the Variables tab for the `Get Zip Code Data` service.

Click the **Add Input** button and in the **Name** text box, type: `zipList`.

Click the **Select** button next to Variable Type and choose `zipCodeListType` from the available types. (This type was generated in the preceding step.)

Click the **Add Output** button and in the **Name** text box, type: `latLong`.

Click the **Select** button next to Variable Type and choose `listLatLonType` from the available types. (This type was generated in the preceding step.)

10. Click the Data Mapping option in the properties.

For the input mapping, click the variable selector icon to the right of the text box and choose the previously created `zipList` variable. When the mapping is done, `tw.local.zipList` displays in the text box.

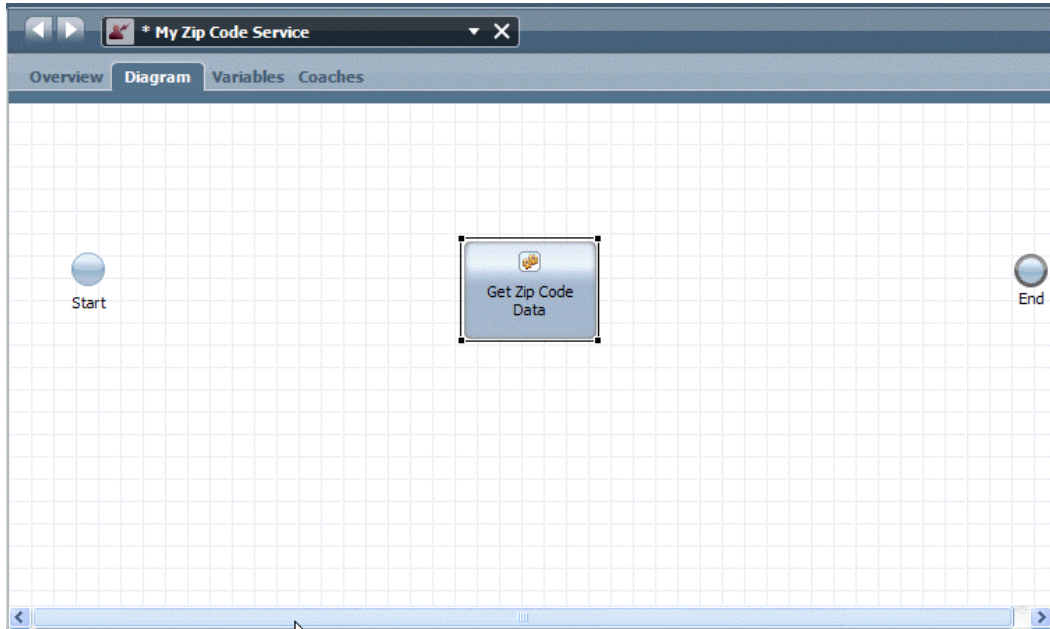
For the output mapping, click the variable selector icon to the right of the text box and choose the previously created `latLong` variable. When the mapping is done, `tw.local.latLong` displays in the text box.

11. Click **Save** in the main toolbar.

Nesting the Integration service and mapping its variables

1. Create a Human service as described in [Creating a service](#) and name it `My Zip Code Service`.
2. Open the diagram for the new Human service and drag the Integration service that you created in the preceding steps (`Get Zip Code Data`) from the library to the diagram.

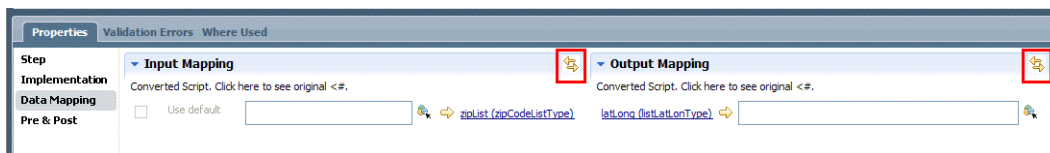
When you have an existing service that you want to nest in another service, you can drag the existing service directly from the library to the diagram of the parent service. This creates the Nested Service component with the service attached in a single step as shown in the following image:



3. If not already selected, click the nested service in the diagram (Get Zip Code Data) to view its properties.
4. Click the Data Mapping option in the properties.

For the parent service to be able to pass the data to and from the nested service into its Coaches, you need to map the input and output variables of the nested service to the parent service. The Designer can auto-map these variables for you.

Because you already created the input and output variables for the nested service, the Data Mapping tab for the parent service includes those variables as shown in the following image:



5. From the Input Mapping section, click the auto-map icon highlighted in red in the preceding image.

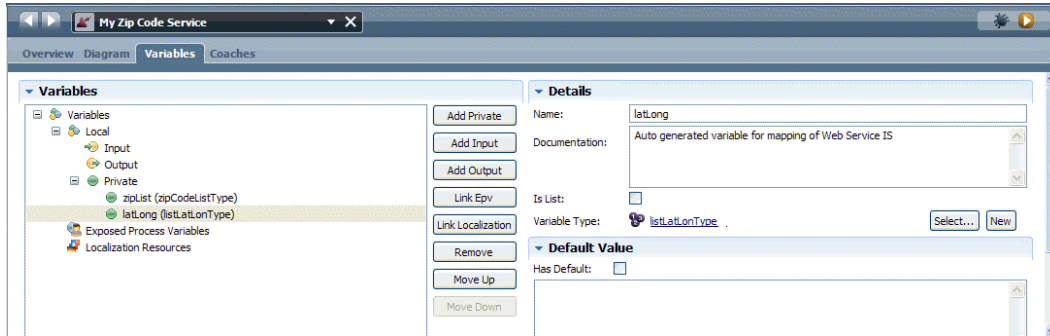
The Create variables for auto-mapping dialog box opens, indicating that a matching private variable was not found in the parent service, and should be created.

6. Select the suggested variable item and then click **OK**.

A private variable of that name is created for the parent Service (My Zip Code Service) and automatically mapped to the input variable of the nested service, making it available to all components in the parent service.

7. From the Output Mapping section, perform the automapping step to create the matching private variable to capture the output from the nested service.

You can see the private variables added for the parent service (My Zip Code Service) as shown in the following image:



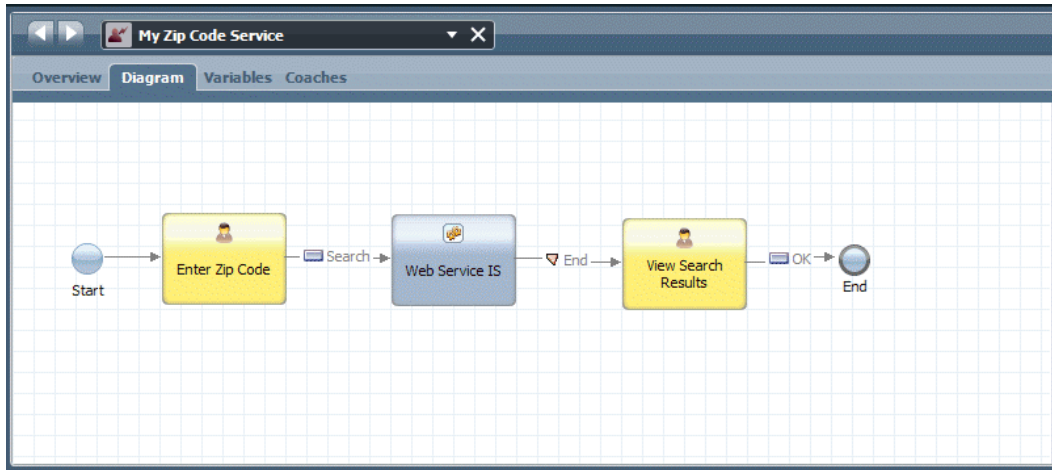
- Click **Save** in the main toolbar.

Building the Coaches to collect input and display output

In the following steps, you'll create the Coach interfaces in the parent service, and map the associated Coach controls to the variables from the previous procedure.

- Click the Diagram tab for My Zip Code Service and then drag a Coach component from the palette to the diagram. (Place the Coach component before the Nested Service component).
- While the Coach component is selected in the diagram, click the Step option in the properties and type `Enter Zip Code` in the **Name** field.
- Click the **Coaches** tab.
- Right-click the Checkbox control, and from the shortcut menu that opens, select **Delete**. Do the same for the Input Text control. Neither of these controls is needed for the Coach.
- Drag the `zipList` variable from the palette to the Coach.
An input text field is created with a mapping to the variable, and a label that matches the variable.
- In the Coach, select the group containing the default OK and Cancel buttons, and then in the properties, click the **Presentation** option.
- In the Buttons section, click the OK Button definition and type `Search` in the **Label** text box.
- Click the Preview tab for the Coach to see your label change.
- Click **Save** in the main toolbar.
- Click the **Diagram** tab for My Zip Code Service.
- Drag another Coach from the palette to the diagram and name it `View search results`. (Place the Coach component after the Nested Service component.)
- Click the **Coaches** tab.
- You can delete the Input Text and Checkbox controls from the Coach since they are not needed.
- From the palette, drag an Output Text control to the Coach.
- In the properties, select the Output Text option in the properties and in the Behavior section, click the **Select** button to create a binding to the results variable.

16. From the list that opens, find and select the `latLong` variable.
17. Click the Diagram tab to return to the diagram view of My Zip Code Service. Select the Sequence Flow tool from the palette and then connect the components as shown in the following image:



18. Click **Save** in the main toolbar.
19. To test My Zip Code Service, click the Run icon in the upper right corner.

The Coach opens in your browser. When prompted, enter a valid zip code and then press the **Search** button.

The service returns the latitude and longitude for the zip code location.

Building a General System service

Use General System services when you want to orchestrate other background services, manipulate variable data, generate HTML for a Coach, or perform some other actions that do not require any integrations or business rules. General system services are likely to be called directly from a BPD or from a Human Service.

General System services can include only basic service components such as scripts and cannot contain Coaches or integration components (Web Service integration or Java integration). General System services can be nested within any other type of service.

You can create a General System service as described in [Creating a service](#).

Building Coaches

When you build Human services you usually include Coaches, which provide the interfaces for end-user interaction. In the first stage of designing a Coach, your goal may be to build a mockup with static elements so that you can visualize what data is needed in the runtime Coach, and where the data should be displayed in the layout. After you have designed the look and feel of the Coach, you need to feed real business data to the Coach controls for your process participants to interact with and to help them make the appropriate decisions. This requires creating bindings between the Coach controls and the data structures (variables) that represent the business data within your Lombardi processes.

To understand how to build Coaches and how to include them in services, you can review the following procedures which include samples of Coaches:

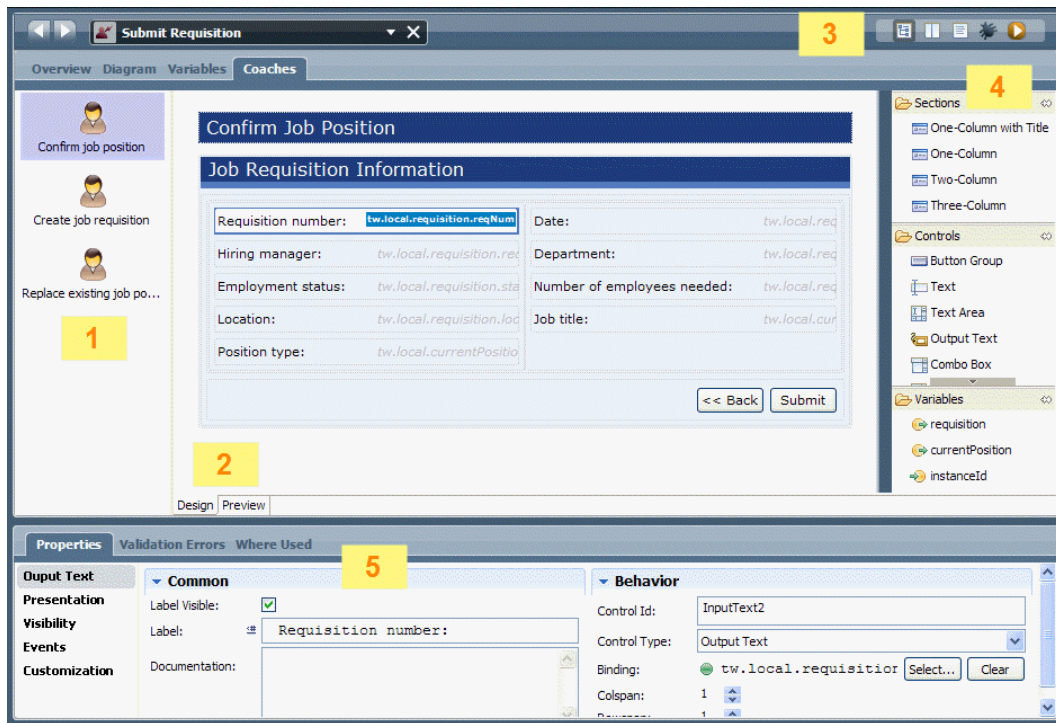
- [Building a Human service](#)
- [Building an Ajax service](#)
- [Building an Integration service](#)

See the following topics to learn more about building Coaches:

To learn how to...	See...
Use the Coach designer to begin to build interfaces for the end users of your processes	Understanding the Coach designer
Use the Coach designer to build an initial mockup of a Coach	Adding sections to a Coach and controlling the layout
Make Coach controls required fields, bind variables to Coach controls, and perform other tasks to configure the controls in your Coaches	Configuring Coach controls
Enable end users to upload documents and then display those documents in Coaches and also embed Lombardi reports in Coaches	Adding documents and reports to Coaches
Include custom images, override CSS styles, or perform other customization tasks	Customizing Coaches
Identify and fix problems with Coaches	Troubleshooting Coaches

Understanding the Coach designer

When creating a Human service that includes one or more Coaches, you can click on the Coaches tab to access the Coach designer. The following image shows the Coach designer and each functional area:

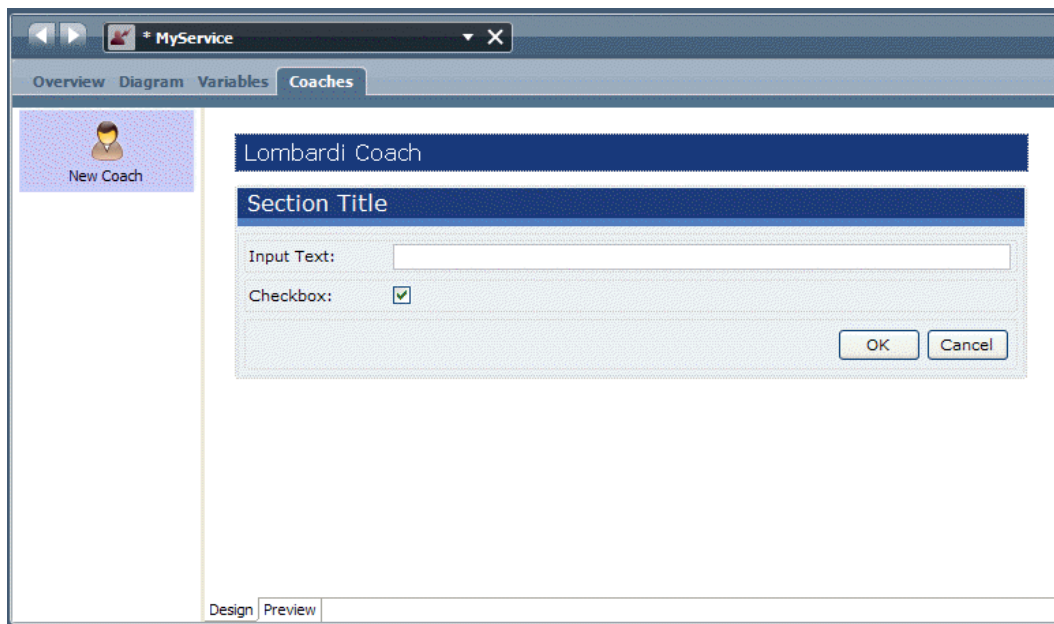


1	Shows all the Coaches in the currently open service. Click one of the Coaches to change or implement the controls it contains. In the preceding example, the first
---	--

	Coach, named Confirm job position, is open and you can see its controls in the Design tab.
2	Shows the design and implementation of the selected Coach in the Design tab. You can click the Preview tab to see how the Coach will appear to end users when the service runs.
3	Use the first three icons to: (1) show the design of the Coach; (2) show both the design and the code of the Coach; (3) show only the code of the Coach, respectively. When you show both the design and the code, the design displays in the top half of the dialog and the code shows in the bottom half. (The final two icons are used for the service itself; to debug or run the entire service.)
4	Lists the sections, controls, and variables that you can include in your Coach. Drag a section or control from the palette to the Design tab to add it to your Coach. To create a control for a variable included in the service, drag a variable from the palette to the Design tab. The type of control created depends upon the variable. For example, an input variable that is a String creates an Input Text field.
5	Shows the properties for the control selected in the Design tab. In the preceding example, the output text field called Requisition number is selected and you can see its properties. Click another option in the properties to configure other aspects of the control such as Presentation and Visibility.

Adding sections to a Coach and controlling the layout

When you create a Human service and drag a new Coach from the service palette to the diagram, the Coach includes several controls by default as shown the following image:



By default, a new Coach contains a single-column section with a title that includes an Input Text control, a Checkbox control, and a Button Group. You can add sections and controls and adjust the layout for your Coach as described in the following steps. To refine the layout, see the following procedures:

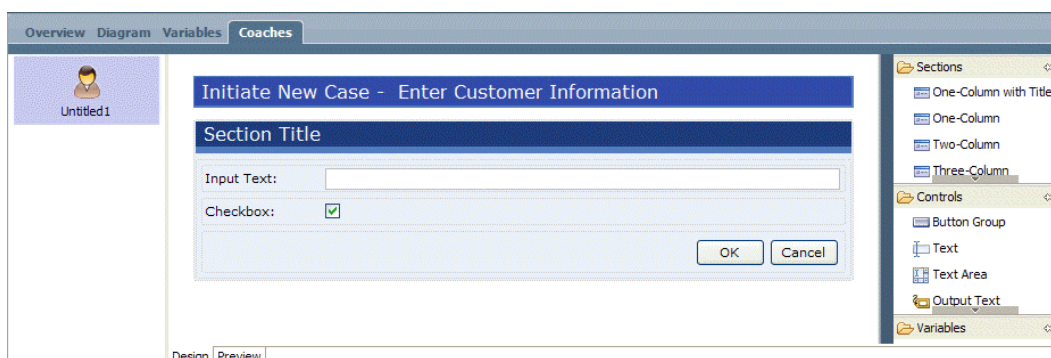
- [Setting column widths in a Coach](#)
- [Setting the number of columns in a Coach](#)



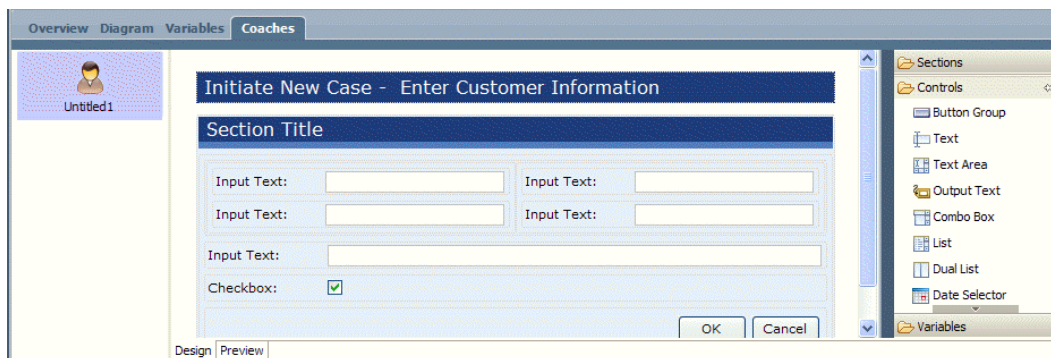
When you use the Activity Wizard to create a Human service, resulting Coaches include a section or control for each of the parameters you choose to use as input or output for the service. For an example of a Coach that is created when using the Activity Wizard, see [Building a Human service](#).

The following steps describe how to build a mockup of a Coach that enables personnel in a call center to collect data about customer issues. The mockup allows you to demonstrate the design to end users as you develop a plan for the steps within a process. You can use feedback from end users to refine the design and thus help guarantee that the eventual implementation meets all requirements.

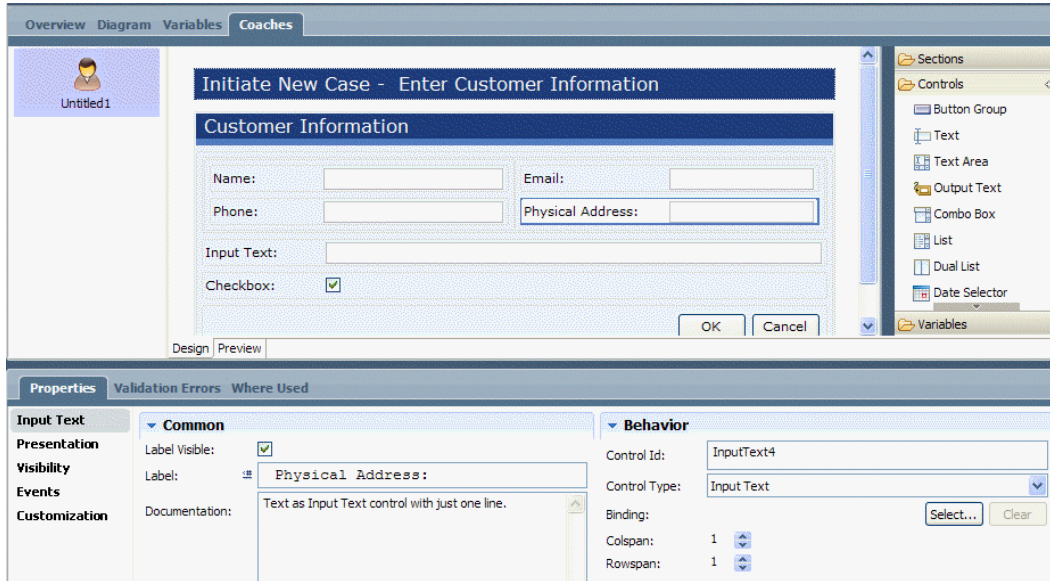
1. To start developing the mockup, change the title of the Coach by clicking the Lombardi Coach title bar in the design area.
2. In the Coach option of the properties, type `Initiate New Case - Enter Customer Information` as shown in the following image:



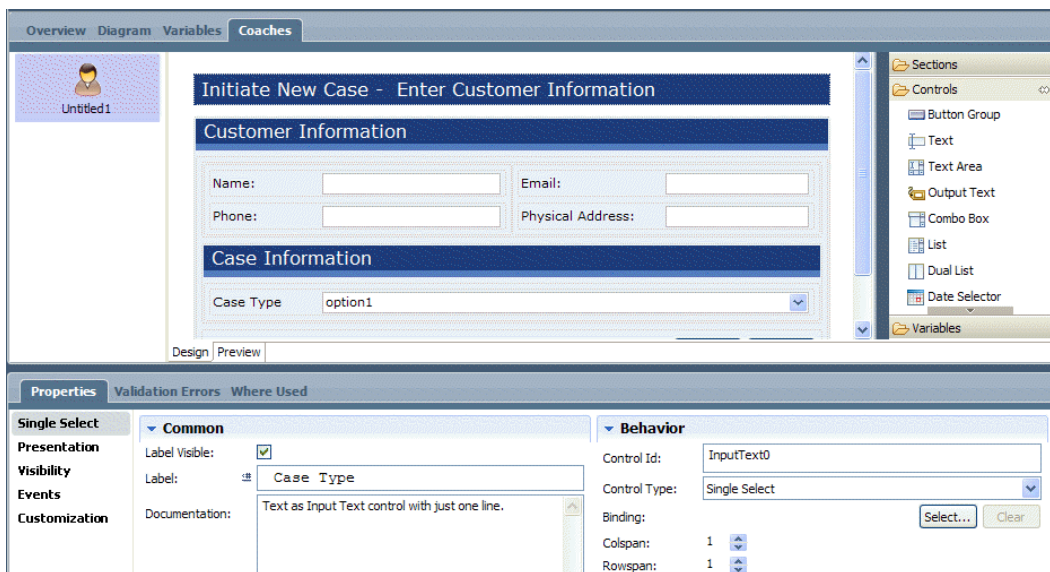
3. Drag a `Two-Column` section from the palette onto the design area so that it is positioned directly below the existing `Section Title`.
4. Drag four `Text` controls from the palette onto the design area so that two `Text` controls are in each column as shown in the following image:



5. Click the `Section Title` in the design area and in the properties type `Customer Information` in the Title text box.
6. Click an `Input Text` control in the design area and in the properties, change the Label for each control to match the following example:

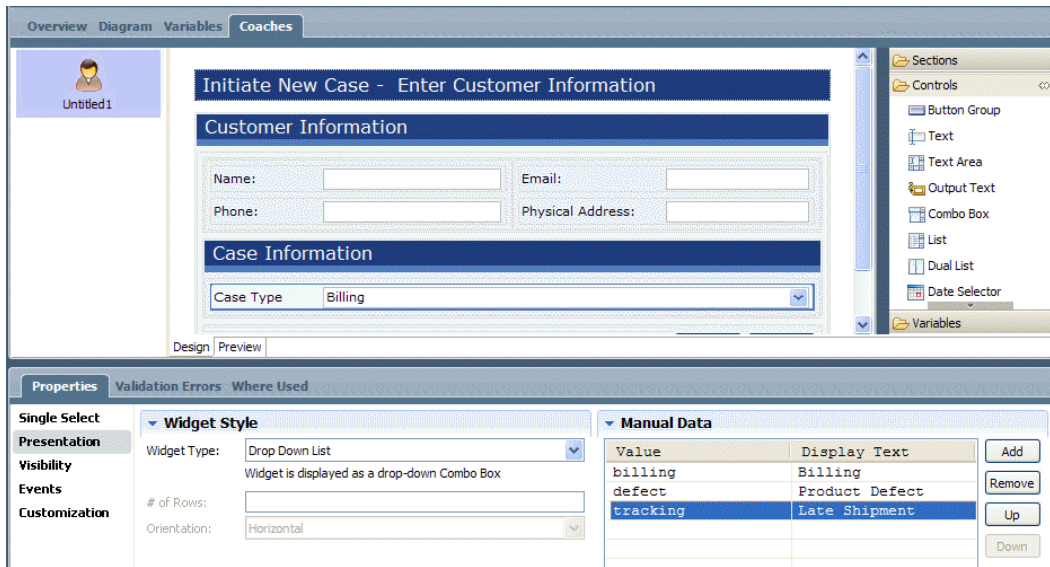


7. Right-click the default Checkbox control, and from the shortcut menu that opens, select **Delete**. Do the same for the default Input Text control. Neither of these controls is needed for this sample Coach.
8. Drag a One-Column with Title section from the palette onto the design area so that it is positioned directly below the existing Customer Information section.
9. While the new section is still selected, type `Case Information` in the Title text box in the properties.
10. Drag a Text control from the palette onto the design area and place it in the new Case Information section.
11. While the new Text control is still selected, type `Case Type` in the Label text box and choose **Single Select** from the Control Type drop-down list as show in the following image:

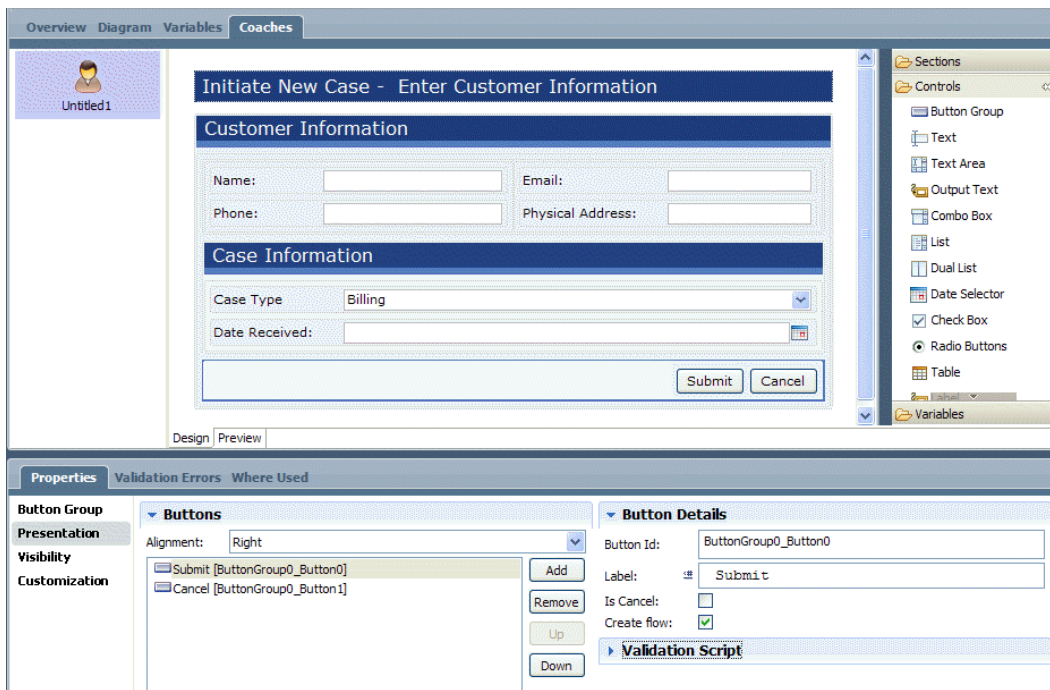


12. Select the Presentation option in the properties and in the Manual Data section, click the **Add** button.

13. Add the values and display text shown in the following example:



14. Drag a Date Selector component from the palette onto the design area and place it directly below the drop-down control in the new Case Information section.
15. In the properties for the Date Selector component, change the label to Date Received.
16. Click the default Button Group at the bottom of the design area, click the Presentation option in the properties, and change the label for the OK button to submit as shown in the following image:

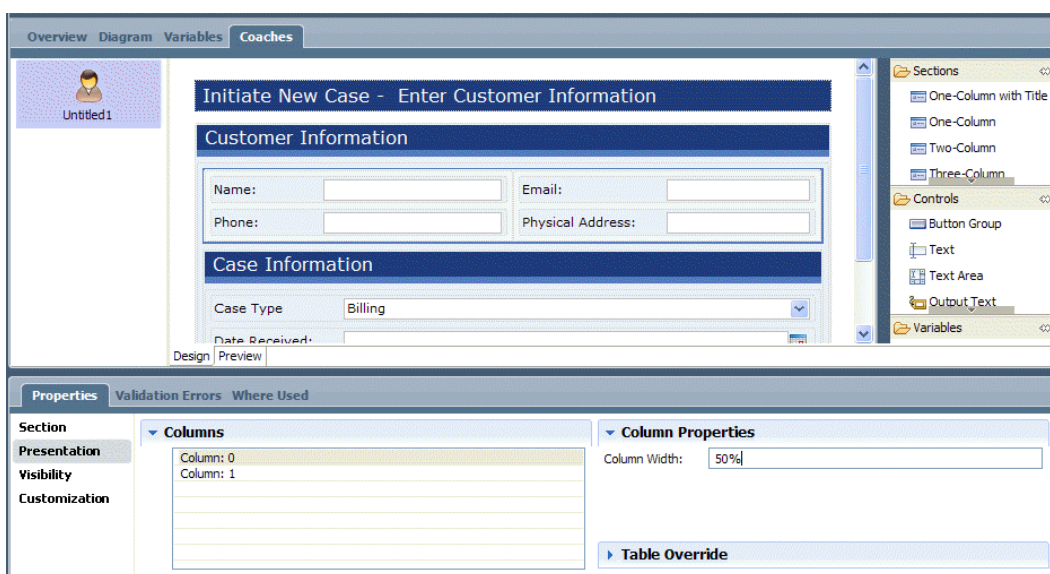


17. Save your work.

- Click the Preview tab to see how the Coach will look when the service runs. You can make adjustments as you see fit in the Design tab.

Setting column widths in a Coach

- Open the service that contains the Coach that you want to change and then click the Coaches tab.
- In the design area, click the section that contain the columns you want to set.
- Click the Presentation option in the properties. If you selected a section containing more than one column, each column is listed in the Columns area in numeric order based on the column ID. Click one of the columns listed to enable the Column Width text box.
- Specify the width of the column in the Column Width text box using any valid HTML size attribute such as 50% or 110px as shown in the following image:

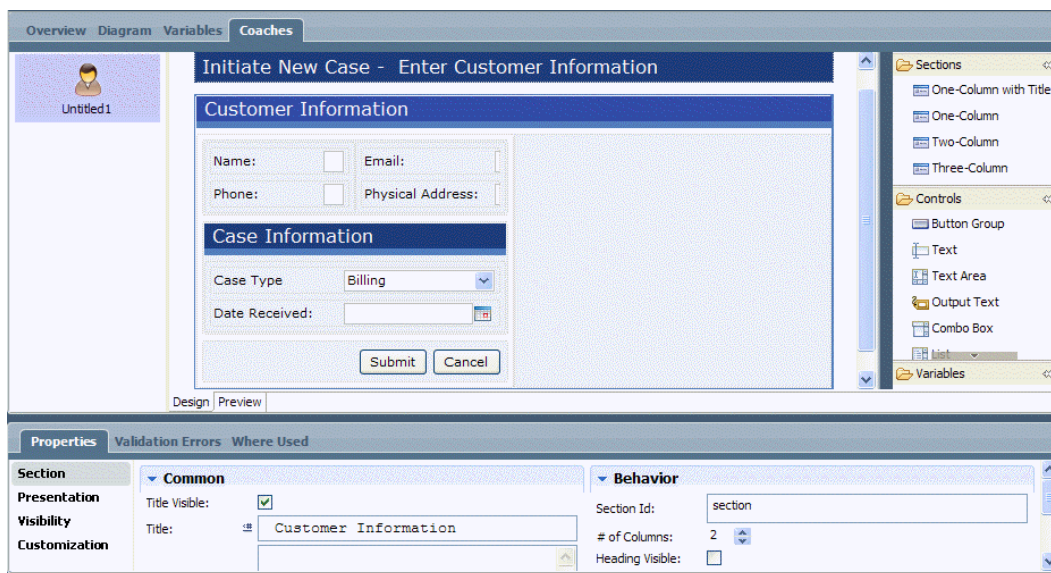


- Click the Preview tab to see how the Coach layout will look when the service runs.

Setting the number of columns in a Coach

- Open the service that contains the Coach that you want to change and then click the Coaches tab.
- In the design area, click the section that you want to change.
- Click the Section option in the properties and then click the up and down arrows for the **# of Columns** option to increase or decrease the number of columns in the section.

In the following example, one additional column is added to a section that contains a nested two-column section. Notice the behavior of the nested section when its parent section is configured.



4. Click the Preview tab to see how the Coach layout will look when the service runs.

Configuring Coach controls

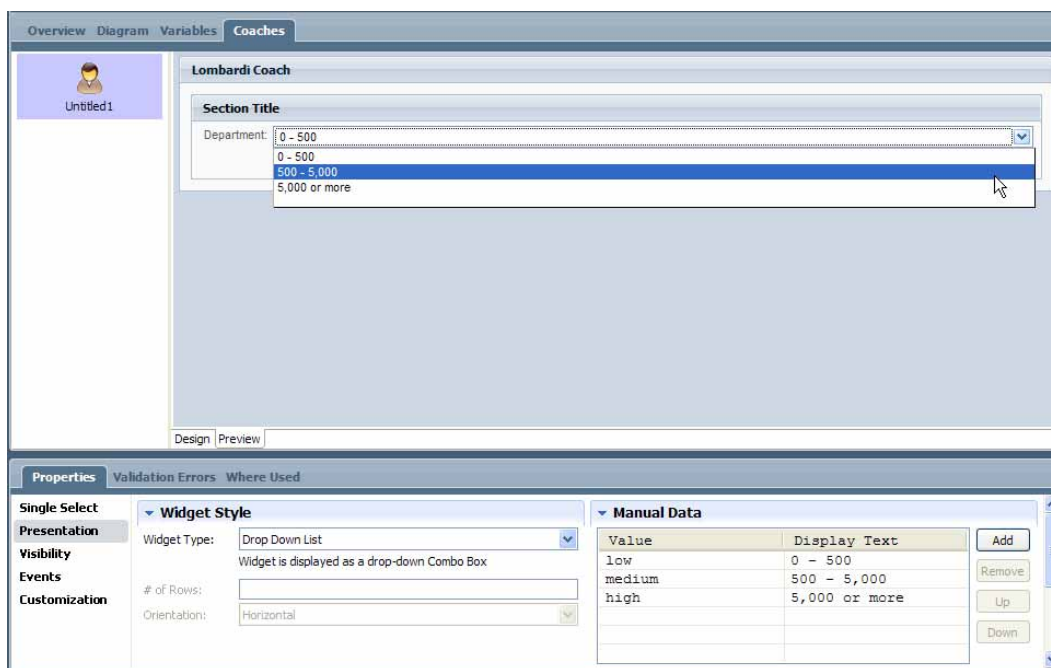
When building Coaches, you have several options for configuring the Coach controls that you add. Read the following topics to learn more:

To learn how to...	See...
Demonstrate the type of data that the Coach will display at run time	Populating a list with static data
Bind dynamic data to a Coach control	Populating a list with dynamic data
Bind a complex data structure to a Table control in a Coach	Binding a complex data structure to a Table control
Use the Execute SQL option to retrieve data directly from a data source	Populating a Table control using a SQL query
Render the content in an HTML block using the runtime value of a variable	Binding a variable to a Custom HTML component
Make input mandatory by configuring a Coach control	Making an input control a required field
Create a control that is displayed only when a related control is set to a specific value	Displaying a control based on the input value of another control
Display a control to only those users who are members of a particular participant group	Displaying a control to a specific group
Use a custom script to override the default visibility rules	Using a custom script to control visibility
Add validation scripts for button controls to ensure that end users provide all required input	Using validation scripts for button controls
Add field formatting capability to Input Text and Output Text controls and align button controls	Controlling field and other formatting in Coaches

Populating a list with static data

Typically, a Coach displays business data that resides in a variable, enabling Lombardi end users to view and interact with the data. In the initial design stages, you may need to populate a Coach with static (manual) data so that you can illustrate the type of data that the process will display at run time. The following example illustrates a Combo Box control that uses static data to populate a list of options.

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Drag a Combo Box control from the palette onto the design area.
3. Click the Presentation option in the properties.
4. Under Manual Data, click the **Add** button to create a row for each option that you want to add to the list. The value that you type in the Display Text column is the name of the option that is displayed to the end user at run time.
5. As shown in the following example, click the Preview tab to see how the list will work when the service runs:



Populating a list with dynamic data

Before you can bind dynamic data to a Coach control, you need to create the appropriate variables for your process or service. See the following topics to learn more:

To learn how to...	See...
Create variables for a process	Adding process variables to a BPD and Declaring and passing variables
How to map those process variables to Coach controls	The following procedure and Building a Human service for an example
Map variables to a nested service and then bind those variables to Coach controls	Building an Integration service for an example



You can also bind Ajax services to Coach controls to perform actions such as automatically populating drop-down lists and enabling type-ahead capability in input fields. For more information, see [Building an Ajax service](#).

The following procedure illustrates a Combo Box control (single-selection list) that uses a preexisting process variable to populate a list of options:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.



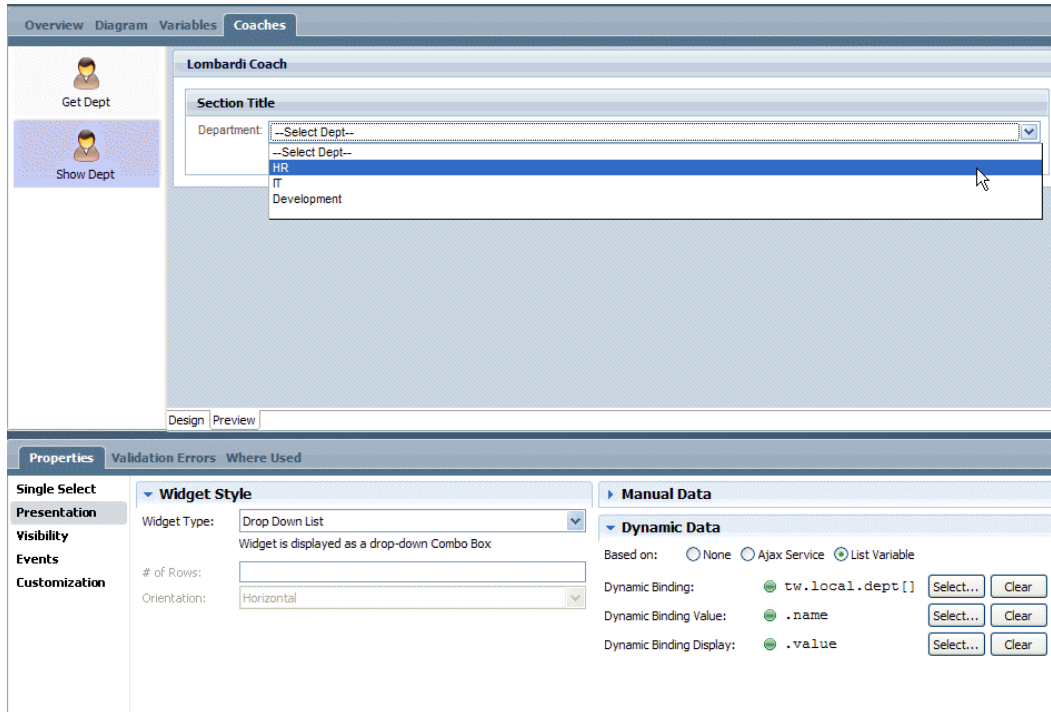
Open or create a service for which you have declared a variable that is a complex structure.

2. Drag a Combo Box control from the palette onto the design area.
3. While the Combo Box control is still selected, click the Presentation option in the properties.
4. Under Manual Data, click the **Add** button to include static instructions at the top of the drop-down list. For this example, the static text is: -- Select Dept --
5. Under Dynamic Data, for the Based On option, click the List Variable radio button.
6. For the Dynamic Binding option, click the **Select** button to choose the preexisting variable from the library as shown in the following image.



For this example, the control must be bound to a complex structure that is a list.

7. Click the Preview tab to see how the list will work when the service runs as shown in the following image:



Binding a complex data structure to a Table control

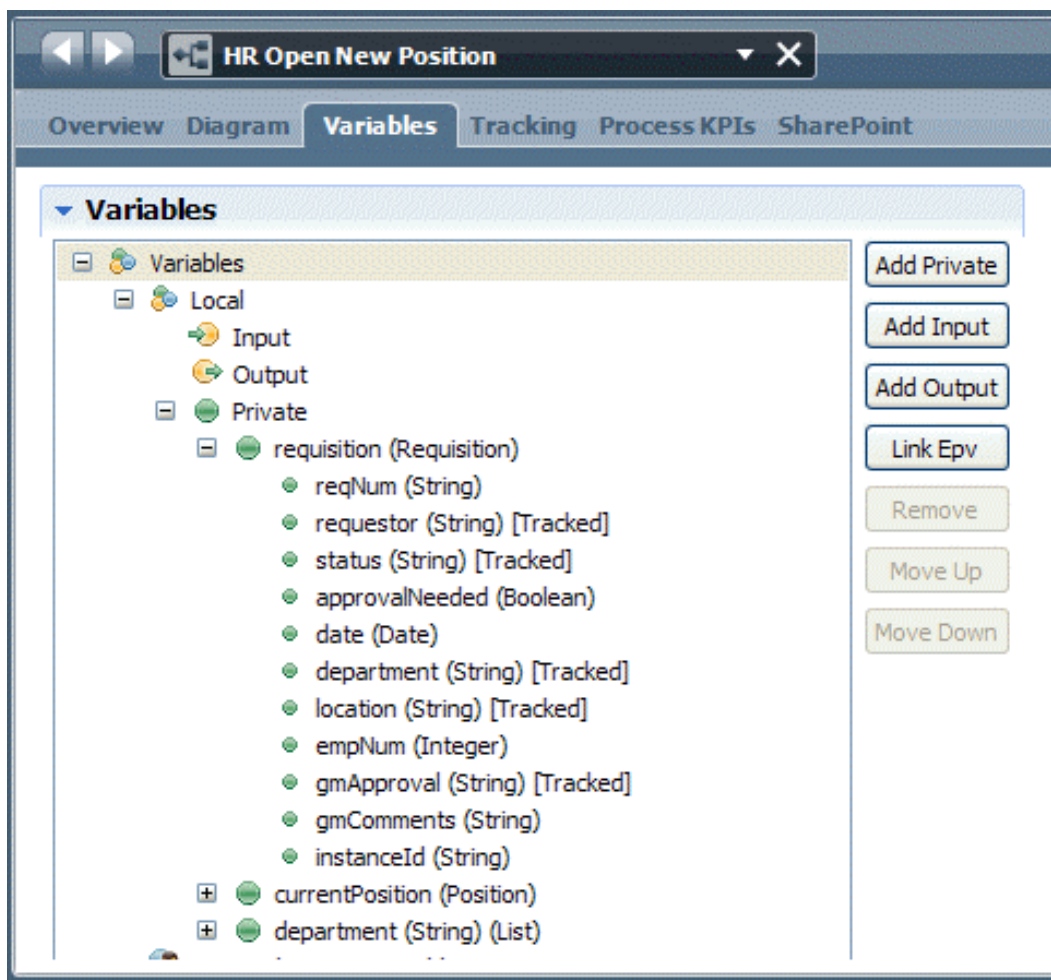
If you have created a complex data structure and you want to bind it to a Table control in a Coach, you do not have to create the table and then bind each element of the table to the appropriate variable parameter. You can create the table and the bindings automatically by dragging the data structure to the Coach. Before you can perform this task, you must first create the data structure and declare it in the service where you want to build a Coach.



You can also use the Activity Wizard to automatically create Coach control bindings. See [Building a Human service](#) for an example.

The following procedure illustrates how to create a complex structure and then use it in a Coach:

1. The complex data structure shown in the following image is a private variable in an HR process used to submit requests to open new positions:

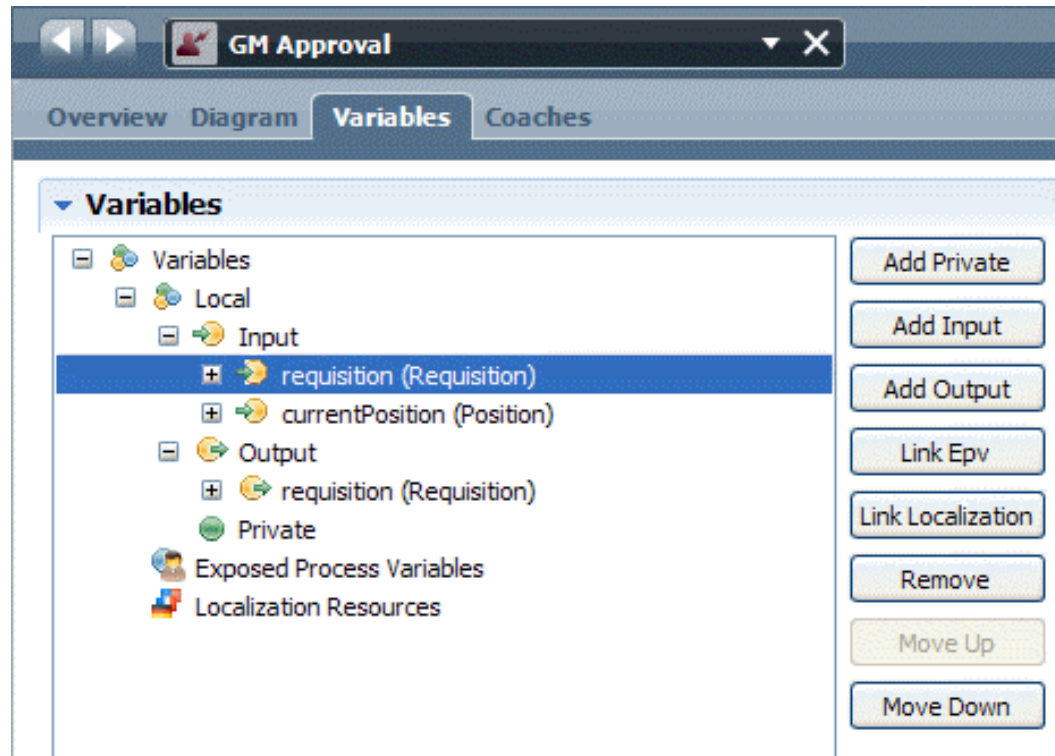


You can see this process and its variables in the Quick Start Tutorial process application. (For more information see the *Quick Start Tutorial Guide* or online help.)

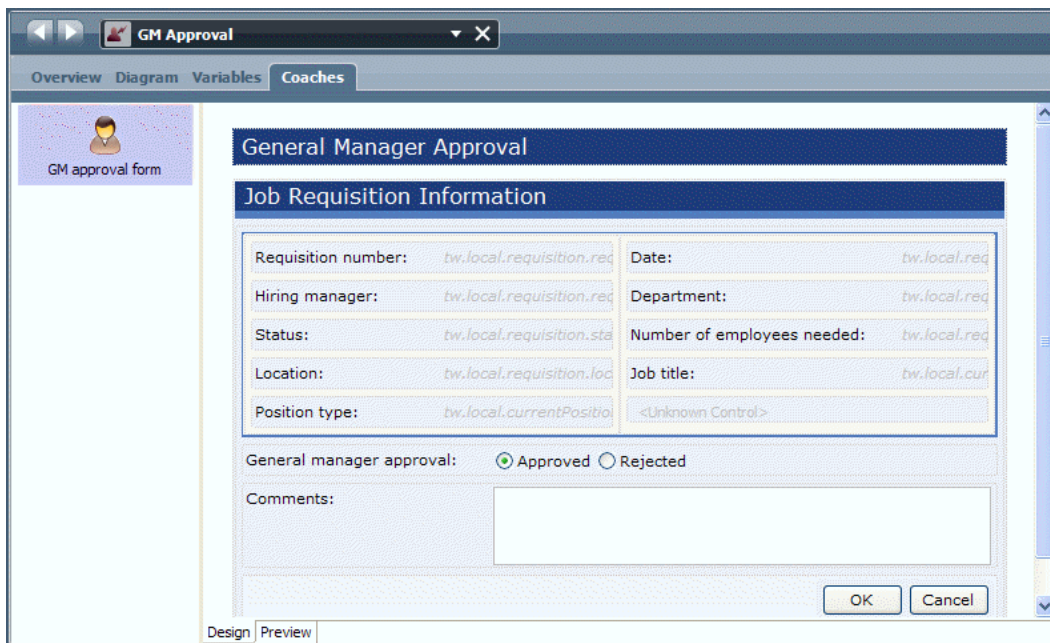
2. At one point in the process, a General Manager must approve a submitted request. The Coach displays the information submitted in the requisition to enable the General Manager to make a decision. The

GM Approval service (in the Quick Start Tutorial process application) includes the requisition variable as both input and output as shown in the following image.

This enables the Coach to display the requisition parameters to the General Manager and then the service passes the values of the parameters in the requisition variable back to the BPD for processing by subsequent steps in the process.



3. To display the values of the parameters in the `requisition` variable in a Coach, click the Coaches tab.
4. Drag the input `requisition` variable from the palette to the design area.
5. Right-click and select **Delete** for the parameters that should not be displayed as output text; select all remaining fields and change the Control Type to Output Text.
6. Click the Section option in the properties and increase the number of columns to 2 so that the Coach Design tab resembles the following image:



7. Click the Preview tab to see how the table will look when the service runs. When you run the tutorial BPD, the Coach displays the table with the appropriate data for the manager to act upon as shown in the following image:



Populating a Table control using a SQL query

Table controls include an Execute SQL option that enables you to retrieve data directly from a data source. The Execute SQL option enables you to populate a Table control dynamically, without having to initialize the variable first. The following procedure illustrates how to dynamically populate a Table control using a query.

Before using a SQL query to populate a Table control, be aware of the following:

- The Execute SQL option is only used for retrieval of data (insert, update, and delete operations on data are not allowed).

- The SQL query is executed before the Coach is displayed, and only if the variable to which the table is bound has no current value. If you want to refresh the value every time the Coach is rendered, select the **Reload** option. For example, if the Service updates the value during the life of the run-time task, you would want to reload this value each time the Coach is rendered.
- The property names of your custom data structure must match the column names in the database table that you want to query. When executing the SQL query, Lombardi uses these labels to match the values from the columns to the correct rows in your Coach table. Note that the names are not case-sensitive. If the property names do not match the column names, you can use column aliases in your SQL statement to perform the correct matching. The following example:

```
SELECT PRICE AS Cost, ITEM_NAME AS ItemName
```

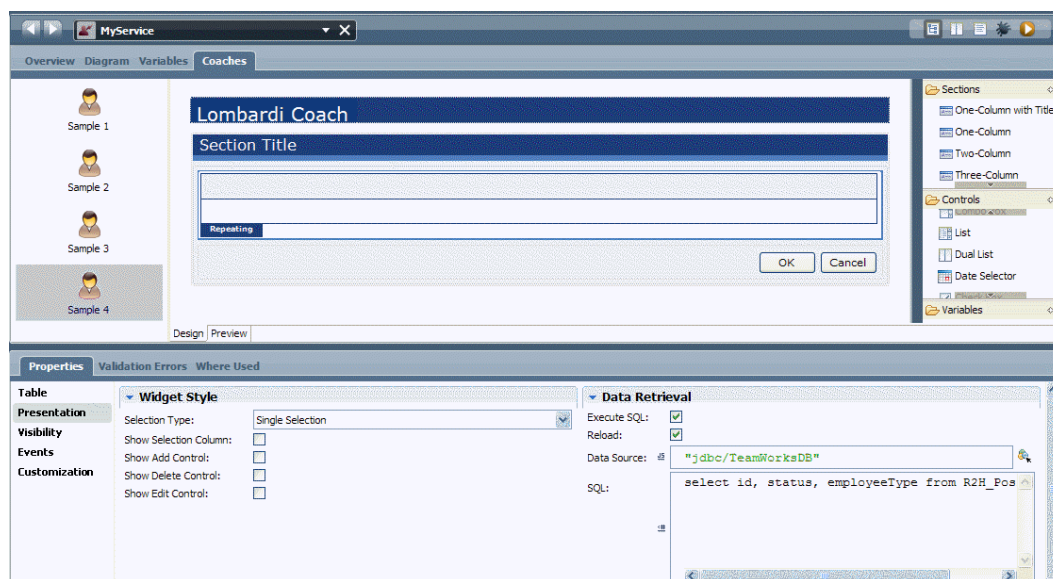
returns the value of the PRICE column to the Cost property in your custom data structure, displaying it in the column to which it is bound in the Coach. The value of the ITEM_NAME column is displayed in the Item Name column.

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Drag a Table control from the palette onto the design area.
3. While the Table control is selected, click the Presentation option in the properties.
4. Click the **Execute SQL** check box to enable it.
5. In the **Data Source** text box, type the data source from where you want to retrieve the data. By default, the data source is "jdbc/TeamworksDB", which points to Lombardi databases.

When you want to use a data source other than the jdbc/TeamworksDB data source, ensure that it is an XA data source. If you use a non-XA data source, or an emulated XA data source, you might receive an error about a database connection failure.

6. In the SQL text box, type a SQL query to select the data that you want from the data source.

The following example selects the ID, status, and employee type from a table named R2H_PositionType:



The order of the entries is in the order which the table rows are returned. Use an ORDER BY clause in your SQL statement to override this behavior.

Binding a variable to a Custom HTML component

The Custom HTML component can be used in a Lombardi Coach to accomplish a variety of runtime data presentations. The following example shows you how to render the content in an HTML block using the runtime value of a variable.

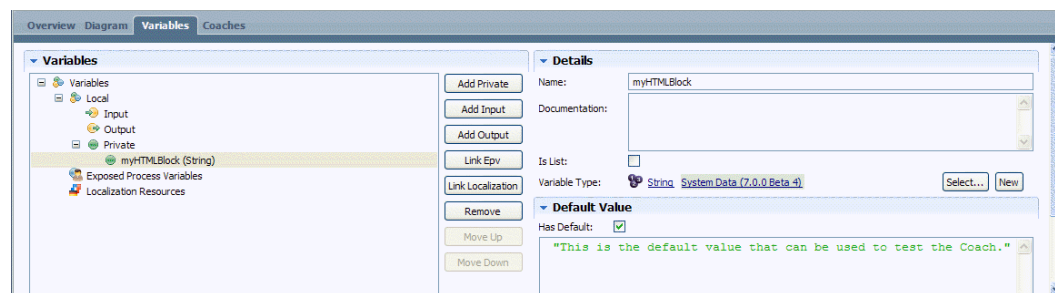
1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Drag a Custom HTML control from the palette onto the design area.
3. While the Custom HTML control is selected, click the Presentation option in the properties.
4. In the **HTML** text box, type the variable whose value will populate the HTML block at run time.

In this example, `tw.local.myHTMLBlock` is declared in the Variables tab for the service and then used to set the label of the HTML block at run time. Type the following in the HTML text box:

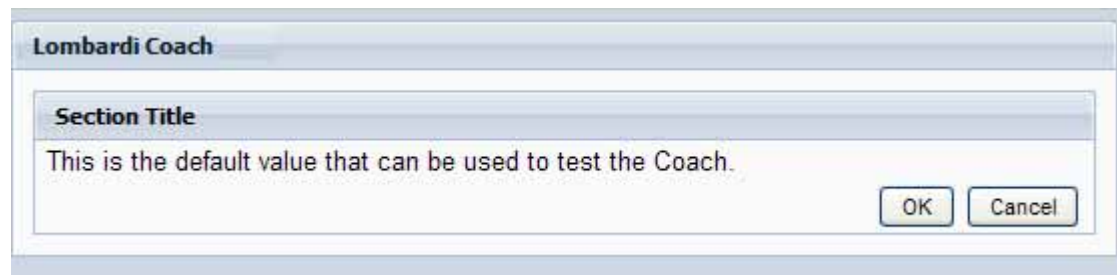
```
<p><#=tw.local.myHTMLBlock#></p>
```

5. To render the content in the HTML block, you need to run the service. When you run the service, the variable is evaluated and its runtime value is then passed to the Coach.

For a quick test, define a default value for the variable (in the Variables tab) as shown in the following image:



6. Run the service to view the Coach. The Coach displays the runtime content of the HTML as shown in the following image:



Making an input control a required field

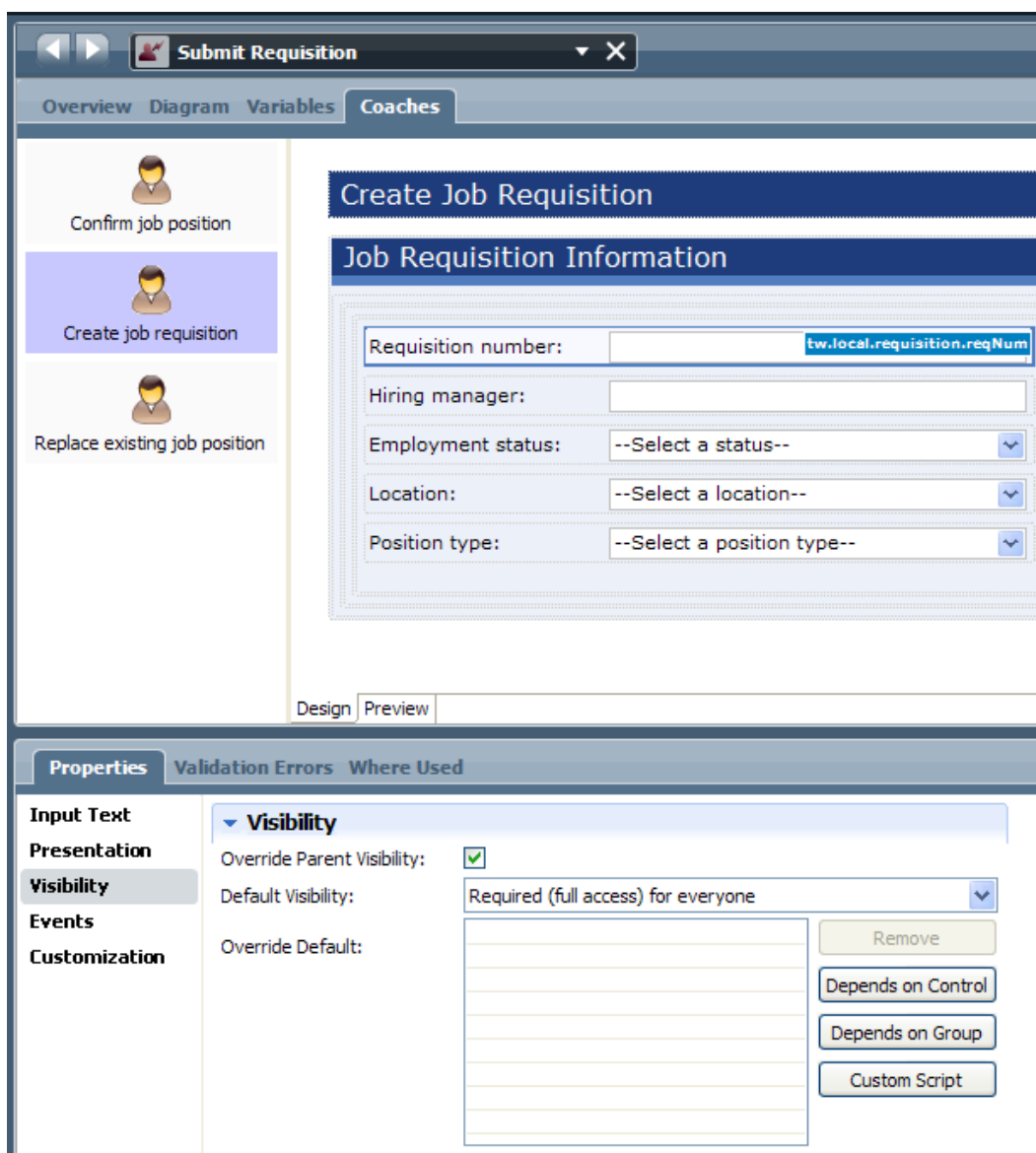
Typically, a business process requires certain input from its runtime participants. You can make input mandatory by configuring a Coach control as described in the following steps:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Click the Coach control that you want to make a required input area.
3. Click the Visibility option in the properties.

By default, input controls are visible to and can be edited by everyone.

The Visibility properties enable you to restrict what process participants see in the run-time Coach, and under what conditions they see it.

4. Click the **Override Parent Visibility** check box to enable it. Doing so allows you to change the default Visibility properties.
5. From the Default Visibility list, select the **Required (full access) for everyone** for everyone option as shown in the following image:



6. (Optional) Clear the **Override Parent Visibility** check box to set the Visibility properties back to read-only mode.
7. Run the service to test the runtime Coach:

The screenshot shows a web form titled "Create Job Requisition" with a sub-section "Job Requisition Information". The form contains several input fields and a "Next >>" button. The "Requisition number" field is highlighted in red, indicating an error. Below the form, an error message reads: "ERROR: Requisition number: is blank but must have a value".

Requisition number:*	<input type="text"/>	Date:	10-21-2009
Hiring manager:	<input type="text" value="Bob Smith"/>	Department:*	<input type="text" value="Finance"/>
Employment status:*	<input type="text" value="Full time"/>	Number of employees needed:	<input type="text" value="1"/>
Location:*	<input type="text" value="Boston"/>	Job title:	<input type="text" value="Accountant"/>
Position type:*	<input type="text" value="New"/>	<input type="button" value="Next >>"/>	

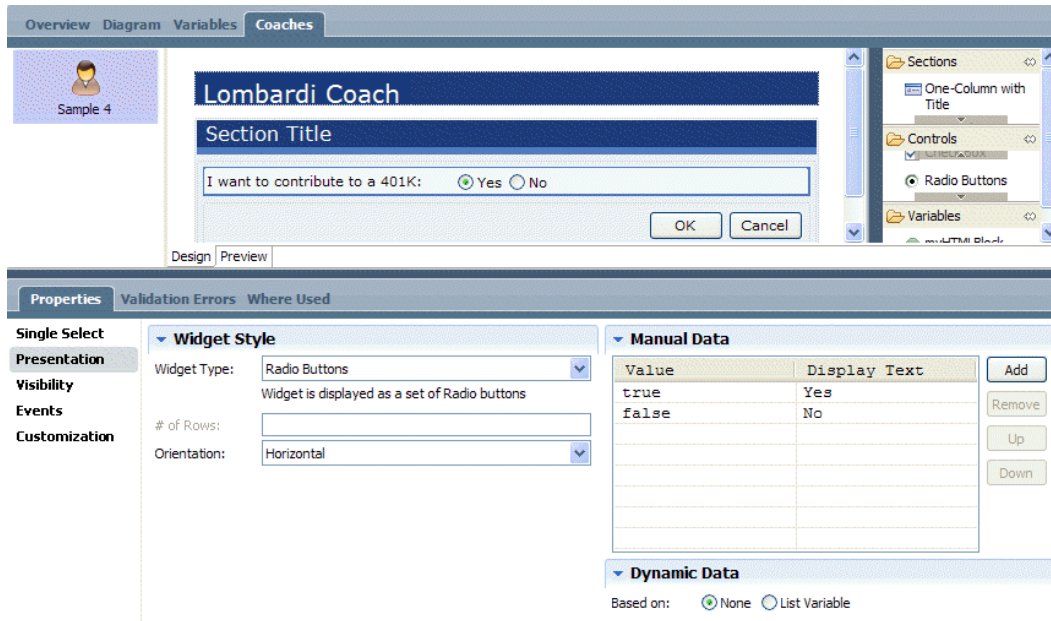
ERROR: Requisition number: is blank but must have a value

If you leave the required input text box empty and then click Next, the input text box is shown in a different color and you are not able to end the task successfully until you supply the required input.

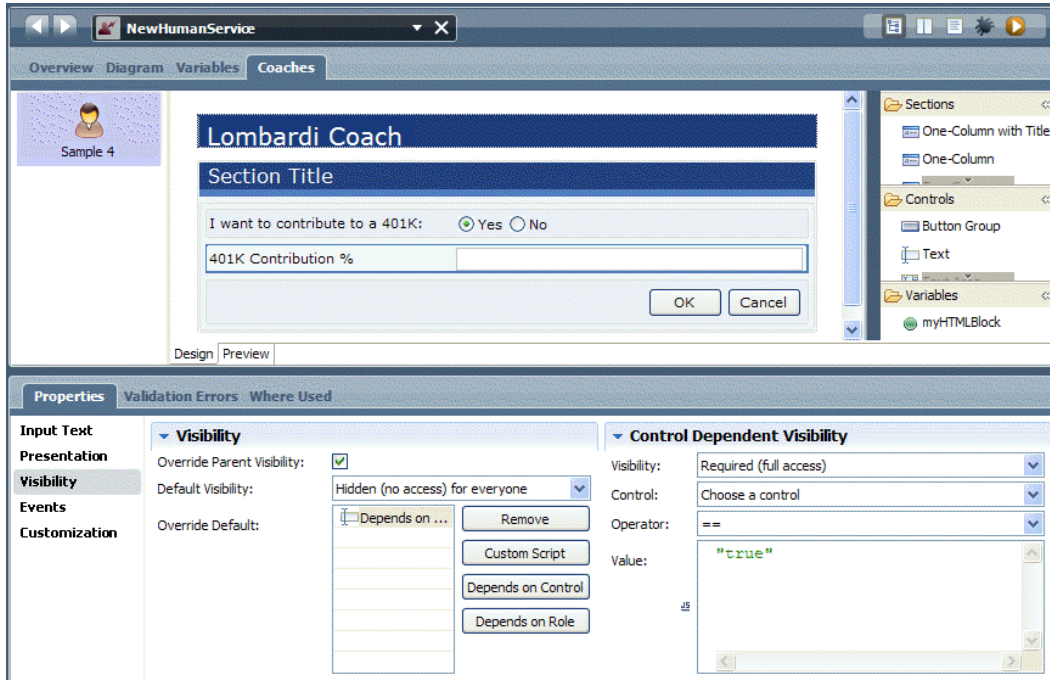
Displaying a control based on the input value of another control

The following procedure illustrates how to create a control that is displayed only when a related control is set to a specific value. The Coach in this example is used by new employees to specify the benefits that they want. If the employee decides to participate in a 401K plan, the Coach displays an input field where the employee can indicate the percentage of his pay to contribute to the plan.

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Drag a Radio Buttons control from the palette onto the design area.
3. In the properties for the control, type `I want to contribute to a 401K` in the Label text box.
4. Click the Presentation option in the properties.
5. Under Manual Data, click the **Add** button to create two Value-Display pairs, one for each radio button option as shown in the following image:



6. Drag a Text control from the palette onto the design area.
7. In the properties for the control, type `401K Contribution %` in the Label text box.
8. Click the Visibility option in the properties.
9. Click the **Override Parent Visibility** check box to enable it. Doing so allows you to change the default Visibility properties.
10. From the Default Visibility drop-down list, select the **Hidden (no access) for everyone** option.
11. Click the **Depends on Control** button. This creates a new override condition.
12. Under Control Dependent Visibility, select the **Required (full access)** option from the Visibility drop-down list.
13. From the Control list, specify the Coach control whose input value will determine if the selected control will be displayed to participants when the service runs. (In this example, the `I want to contribute to a 401K Radio Buttons` control).
14. From the Operator list, select the **==** (equals) operator.
15. In the Value text box, type `true` as shown in the following image. (This is the value that you assigned to the Yes radio button option in step 5.)



16. Save the Coach and then run the service to see how the Input control is hidden and then shown according to the visibility rules you have specified.

Displaying a control to a specific group

The following procedure describes how to display a Coach control to only those end users who are members of a particular participant group. See [Creating a participant group](#) for more information.

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Click the Coach control that you want to display to only those members of a particular group.
3. Click the Visibility option in the properties.
4. Click the **Override Parent Visibility** check box to enable it. Doing so allows you to change the default Visibility properties.
5. From the Default Visibility drop-down list, select the **Hidden (no access) for everyone** option.
6. Click the **Depends on Group** button. This creates a new override condition.
7. Under Group Dependent Visibility, click the **Select** button to choose the group that you want.
8. Under Group Dependent Visibility, select the visibility option that you want for this group from the Visibility drop-down list.
9. Save the Coach and then run the service to see how the control is hidden and then shown according to the visibility rules you have specified.

Using a custom script to control visibility

When your visibility rules are more complex than depending on group membership or depending on the value of another control, you can use a custom script to override the default visibility rules. The following

procedure illustrates how you can create private variables to use in a custom visibility script, and how the values of those variables determines visibility of the selected control.

1. Open the service that contains the Coach that you want to work with.
2. Click the Variables tab and add the private variables that you need for the custom script. For this example, add Boolean variables named `visible`, `enabled`, and `required`.
3. Click the Coaches tab.
4. Click the Coach control for which you want to add visibility control.
5. Click the Visibility option in the properties.
6. Click the **Override Parent Visibility** check box to enable it. Doing so allows you to change the default Visibility properties.
7. From the Default Visibility drop-down list, select the **Hidden (no access) for everyone** option.
8. Click the **Custom Script** button.
9. Under Custom Visibility, type the Javascript rule that you want to use to control visibility.

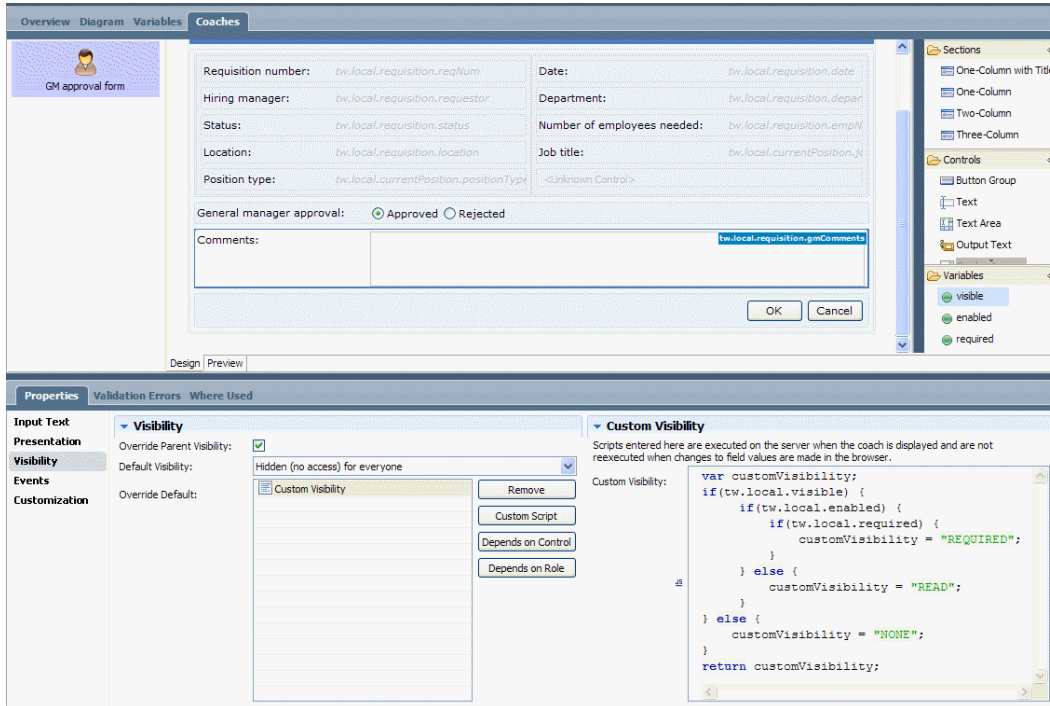
The following example uses a server-side JavaScript function, and so `return` statements are required. For custom visibility using server-side JavaScript, return one of the following values (must be in all caps):

"NONE"	Hidden
"READ"	Disabled
"FULL"	Editable
"REQUIRED"	Required

The following script causes the runtime Lombardi engine to check to see if user input is required. If user input is not required, the control is editable but not required. If the value of `tw.local.visible` is false, the control is not displayed to the end user.

```
var customvisibility;
if(tw.local.visible) {
    if(tw.local.enabled) {
        if(tw.local.required) {
            customvisibility = "REQUIRED";
        } else {
            customvisibility = "READ";
        }
    } else {
        customvisibility = "NONE";
    }
}
return customvisibility;
```

The following image shows the Coach controls and where to enter the script:



10. You can set default values for the variables added in step 2 and then run the service to test the script.

Using validation scripts for button controls

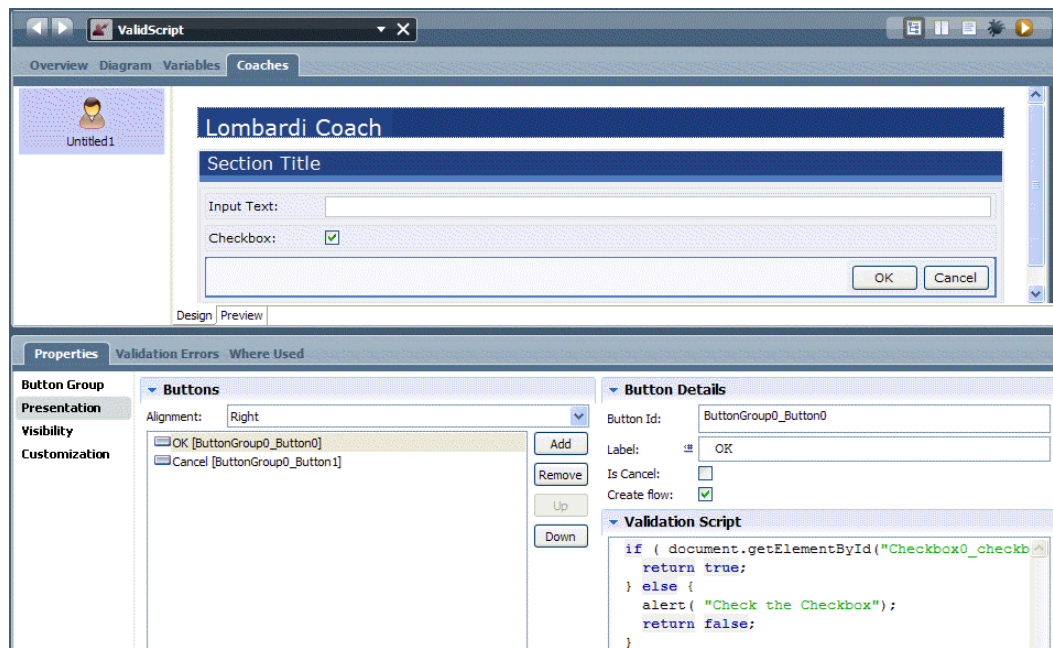
When building Coaches in Lombardi, you can add client-side validation scripts for button controls. Validation scripts ensure that end users provide all required input. If not, the script provides the appropriate feedback to prompt end users for the required information. The following procedure describes how to add a client-side validation script for a Coach control:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. In the design area, click to select the control for which you want to add a validation script.
3. Click the Presentation option in the properties.
4. If multiple buttons are included in the control, under Buttons, click the button that needs the validation script.
5. In the text box under Validation Script, type the JavaScript to validate that the required controls have been set and contain values as expected. To do this, use the control ID of each required control.

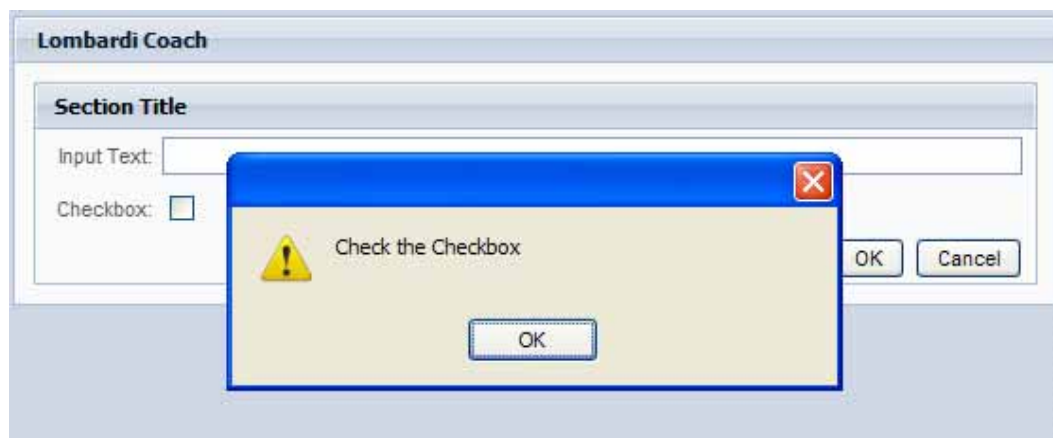
The following example is client-side JavaScript that uses the default controls included when you add a new Coach to a service. The following validation script checks to ensure that the check box is enabled (set to true). If not, the end user is prompted to check it before proceeding by clicking the OK button.

```
if ( document.getElementById("checkbox0_checkbox").checked == true ){
    return true;
} else {
    alert( "check the checkbox");
    return false;
}
```


The following image shows the Coach controls and where to enter the script:



6. Save the Coach and then run the service to test the script. You should see the following results when you click the OK button before enabling the check box:



Controlling field and other formatting in Coaches

When building Coaches, you can add field formatting capability to Input Text and Output Text controls. The pre-defined field formatting available with Coaches includes standards such as US social security number, currency in dollars and Euros, and other standards. You can also use customized formats and apply formats to variables and localization resources that are bound to Coach controls. To learn more, see the following:

To learn how to...	See...
Use the available predefined character formats	Using pre-defined formats in Coach Controls
Use custom formatting characters in the Format field	Using characters to apply custom numeric formatting
Update Lombardi configuration files to add custom formats	Adding custom format types
Use formatting with controls that are bound to variables	Using formatting with variables

To learn how to...	See...
Apply formatting to a control that is bound to a localization resource	Using formatting with language localization resources
Specify the alignment of buttons in a Button Group control	Aligning buttons
Specify the horizontal and vertical alignment of check boxes and radio buttons	Aligning check boxes and radio buttons

Using pre-defined formats in Coach Controls

The following procedure describes how to choose from the available predefined character formats:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Click the Coach control for which you want to add formatting.
3. Click the Presentation option in the properties.
4. Under Widget Style, click the **Select** button next to the Format field and choose the format that you want:

Option	Example
Currency: \$ ###,###,###.##	Enter the value 123456789, the value is formatted to \$123,456,789.00
Currency: ###,###,###.## €	Enter the value 123456789, the value is formatted to 123,456,789.00€
Currency: € ###,###,###.##	Enter the value 123456789, the value is formatted to €123,456,789.00
Integer: ###,###,###	Enter the value 123456789, the value is formatted to 123,456,789
Decimal: ###,###,###.##	Enter the value 123456789, the value is formatted to 123,456,789.00
US phone: (###) 000-0000	Enter the value 5555555555, the value is formatted to (555) 555-5555
US SSN: 000-00-0000	Enter the value 123456789, the value is formatted to 123-45-6789

5. Save your changes.

When you run the service that contains the Coach, the values typed into the control are automatically formatted as shown in the preceding examples. If a user enters non-numeric characters, those characters are removed. If a user enters only non-numeric characters, no formatting is applied.

Using characters to apply custom numeric formatting

If you want to apply numeric formatting to a control for integers and decimals, you are not required to select one of the pre-defined formats. Instead you can manually enter custom formatting characters into the Format field. For example, the # character acts as a digit placeholder. So if you type the following placeholders into the Format field in the Presentation properties for a control, you'll get the described results:

Format placeholder	Results
## (placeholder for two digits)	Since no decimal placeholder is specified, values entered into the control during run time are rounded up to the next integer. For example, if a user enters the value 34.2 into the control, the value is rounded up to 35.
##.# (placeholder for two digits and the tenths decimal position)	For additional decimal positions typed in to the control during run time, decimals less than five are rounded down, and decimals greater than or equal to five are rounded up. For example, the value 34.24 would be rounded down to 34.2, and the value 34.57 would be rounded up to 34.6.

Follow these steps to use characters to apply custom numeric formatting:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Click the Coach control for which you want to add formatting.
3. Click the Presentation option in the properties.
4. Under Widget Style, type the characters that you want to use as placeholders in the Format field. The following characters are available:

Character	Name	Description
#	Digit placeholder	A digit is copied into output. If there is no digit in this position, then nothing is stored in the output.
0	Zero placeholder	A digit is copied into output. If there is no digit in this position, a 0 is inserted into this position.
?	Padding placeholder	A digit is copied into output. If there is no digit in this position, a " " (space symbol) is inserted into this position.
.	Decimal separator	The first . character (period) in the format string determines the location of the decimal separator in the formatted value. The actual character used as the decimal separator is determined by user locale settings.
,	Thousand separator	The , character (comma) serves two purposes. First, if the format string contains a , character between two digit placeholders (0 or #) and to the left of the decimal point if one is present, then the output will have thousand separators inserted between each group of three digits to the left of the decimal separator. The actual character used as the decimal separator in the result string is determined by user locale settings. Second, if the format string contains one or more , characters immediately to the left of the decimal point, then the number will be divided by the number of , characters multiplied by 1000 before it is formatted. For example, the format string 0 , , will represent 100 million as simply 100. Use of the , character to indicate scaling does not include thousand separators in the formatted number. Thus, to scale a number by 1 million and insert thousand separators you would use the format string: # , ##0 , ,
%	Percentage	The presence of a % character in a format string causes a number to be multiplied by 100 before it is formatted. The appropriate symbol is inserted into the number itself at the location where the % appears in the format string.
E0 E+0 E-0 e0 e+0 e-0	Scientific notation	If any of the strings E, E+, E-, e, e+ or e- are present in the format string and are followed immediately by at least one 0 character, then the number is formatted using scientific notation with an E or e inserted between the number and the exponent. The number of 0 characters following the scientific notation indicator determines the minimum number of digits to output for the exponent. The E+ and e+ formats indicate that a sign character (plus or minus) should always precede the exponent. The E, E-, e or e- formats indicate that a sign character should only precede negative exponents.
;	Section separator	The ; character (semicolon) is used to separate sections for positive, negative, and zero numbers in the format string.
Other	All other characters	All other characters are copied to the result string as literals in the position where they appear.

5. Save your changes.

Adding custom format types

The predefined character formats for Input Text and Output Text controls are defined by the `<formatting-templates>` section in the `[Lombardi_home] \process-center\config\system\99Local.xml` file. To modify the formats or create additional formats, copy the `<formatting-templates>` section shown below and paste it into the `[Lombardi_home] \process-center\config\100Custom.xml` file. You can define additional formats as needed in the `100Custom.xml` file.

```
<formatting-templates>
<formatting-template comment="Currency" template="$ ###,###,###.##" />
<formatting-template comment="Currency" template="###,###,###.## €" />
<formatting-template comment="Currency" template="€ ###,###,###.##" />
<formatting-template comment="Integer" template="###,###,###" />
<formatting-template comment="Decimal" template="###,###,###.##" />
<formatting-template comment="US phone" template="(###) 000-0000" />
<formatting-template comment="US SSN" template="000-00-0000" />
</formatting-templates>
```



If you add or modify formats in your development environment by altering settings for the Process Center server, be sure to make the same changes for each Process Server in your runtime environments.

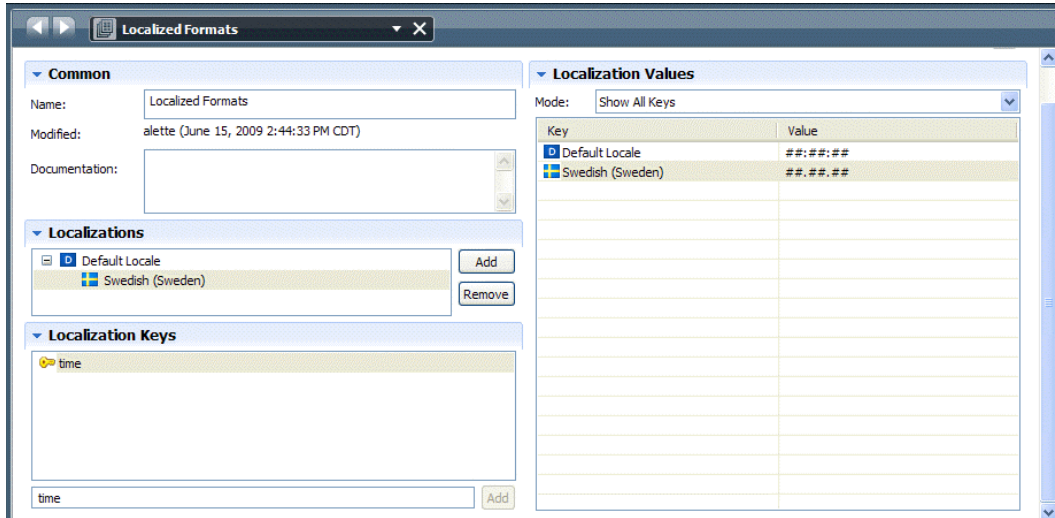
Using formatting with variables

You can apply formatting to a Coach control that is bound to a variable. All input values are treated as numbers, even if they are bound to string variables.

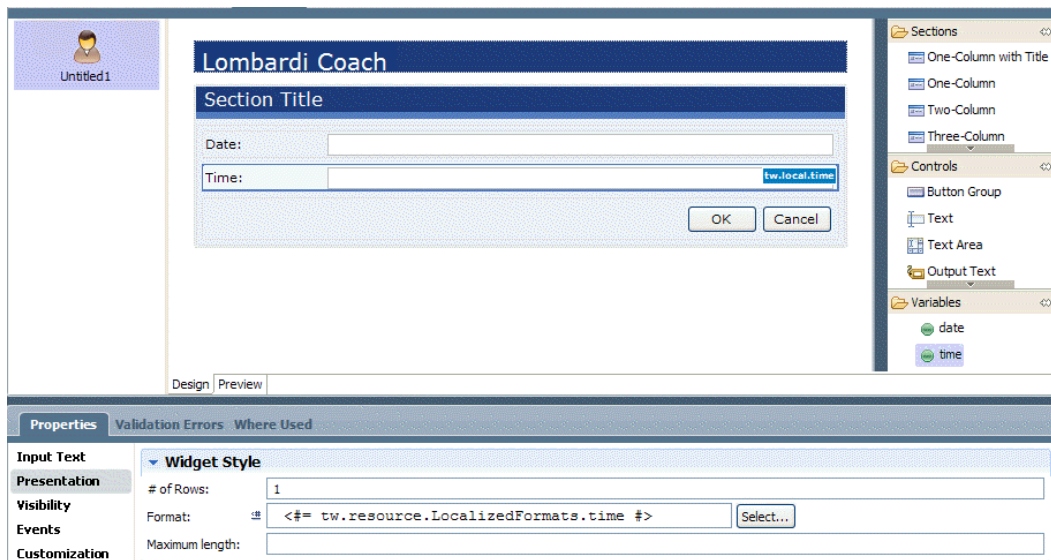
If you create a service that includes a decimal variable named `tw.local.amount` with a default value of `251000.0` and you bind a Coach control to the `tw.local.amount` variable, you can still specify the format in which the value is displayed even though the value that the control displays during run time is determined by the value of the variable to which the control is bound. For example, if the US currency (dollars/cents) format is selected for the Coach control, when you run the service the Coach control is populated with the value of the variable, and the value is formatted to `$251,000.00`.

Using formatting with language localization resources

The following procedure illustrates how you can apply formatting to a Coach control that is bound to a localization resource. The localization resource for this example (named Localized Formats) includes a localization key named `time`, which contains two locales: Default Locale and Sweden. The value of Default Locale is `##:##:##`, which is the standard format used to represent time in the majority of countries. The value of the Sweden locale is `##.##.##`, which is the standard format for Swedish time:



1. Open a service that includes several variables, and click the Variables tab.
2. Click the **Link Localization** button and select the localization resource (in this example, Localized Formats) that you want to link to the service variables as a resource bundle.
3. Create a Coach that includes an input text control named Time, and bind the control's formatting to the localization resource bundle and localization key by typing `<#= tw.resource.LocalizedFormats.time #>` into the Format field as shown in the following image:



4. Save your changes.

You can test the binding by changing the interface language to svenska in Lombardi Process Portal preferences. Then run the BPD that contains the service and run the task from the Process Portal. When you enter a 6-digit value such as 182400 into the Time field, the value should be formatted to 18.24.00, which conforms to the formatting that you specified.

Aligning buttons

When building Coaches, you can specify the alignment of buttons in a Button Group control as described in the following procedure:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. In the design area, click to select the Button Group control that you want to align.
3. Click the Presentation option in the properties.
4. Select the alignment (Left, Center, or Right) for the Button Group from the drop-down list.
5. Click the Preview tab to see how the buttons will be displayed when you run the service.

Aligning check boxes and radio buttons

When building Coaches, you can specify the horizontal and vertical alignment of check boxes and radio buttons, as described in the following procedure:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. In the design area, click to select the Dual List control that you want to align.
3. Click the Presentation option in the properties.
4. From the Widget Type drop-down list, select **Multiple Check Boxes**.
5. From the Orientation drop-down list, select **Vertical** or **Horizontal**.
6. Click the Preview tab to see how the buttons will be displayed when you run the service.



Single-selection List controls that use radio buttons provide the same alignment options in the Presentation properties.

Adding documents and reports to Coaches

Lombardi Coaches enable you to attach documents and reports to aid end users as they perform the tasks generated by Lombardi activities. Read the following topics to learn more:

- [Attaching documents to a Coach](#)
- [Embedding Lombardi reports in a Coach](#)

Attaching documents to a Coach

When you add the Document Attachment control to a Coach, documents that were previously uploaded during completion of a Lombardi task are displayed. You can configure the control to display only those documents that match properties that you set. For example, you can configure the control to display only those documents associated with a specific client or customer. Plus, you can limit the displayed documents to only those documents uploaded during the execution of the current process instance. In addition to displaying documents, you can also configure the Document Attachment control to enable end users to upload additional documents.

The following procedure describes how to display documents in your Coach using the Document Attachment control:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Drag a Document Attachment control from the palette onto the design area.
3. While the Document Attachment control is selected, click the Presentation option in the properties.
4. By default, the **Associated Process Instance** check box under Display Documents is enabled. This setting causes the control to display only those documents uploaded in previous steps of the running process instance. If you disable this check box, the control displays all documents, regardless of the instance, BPD, or process application from which they originated. Therefore, if you disable this check box, be sure to set properties that clearly identify the documents to display. If you don't, the number of displayed documents could be much larger than expected or than is useful.
5. If you want this control to display documents according to properties that you set, click the **All Properties** or **Any Properties** radio button under Display Documents. Then click the **Add** button to add the properties that will determine which documents are displayed. Each property should have a name and a value. For example, you might add a property with a name of `client` and a value of `smith`.

If you select **All Properties**, documents must match all properties that you add. If you select **Any Properties**, if documents match any one of the properties that you specify, the control displays them.



The document properties that you add should match the properties set for uploaded documents. For example, if you want to display documents that end users uploaded in an earlier step, examine the Coach for the earlier step to see the properties established for those uploaded documents. If you want to display documents from a different process, open the BPD and its services and then examine the properties established for those uploaded documents. See the following procedure for more information.

6. Save the Coach and then run the service or BPD to test your configuration. If the same service enables users to upload documents in a preceding step, you can run the service to test the configuration. If not, you must run the BPD so that the current control has access to documents to display.
7. To configure the same Document Attachment control to enable document uploads, see the following procedure. If you don't want to configure this control for document uploads clear the **Enabled** check box under Upload Documents.

The following procedure describes how to enable end users to upload documents using the Document Attachment control:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. Drag a Document Attachment control from the palette onto the design area or click on a preexisting Document Attachment control to select it.
3. While the Document Attachment control is selected, click the Presentation option in the properties.
4. By default, the **Enabled** check box under Upload Documents is enabled. This setting causes the control to display an Add Document button. When the service runs, the end user can click the Add Document button and then use the fields provided to browse for the URL or file that they want to upload, provide a title for the document, and then click OK to upload the document. The end user can upload multiple documents using the control.
5. If you want to supply a default name for all documents that the end user uploads, type the JavaScript for the default in the Default Name text box. For example, type `<#= tw.system.user_fullName #>` to make the current user name the default name for uploaded documents.

If you supply a default name, but want the end user to be able to change the default, click the **User Editable** check box to enable it.

- Click the **Add Properties** check box to enable it if you want to add properties to uploaded documents. Then click the **Add** button to add the properties that you want. Each property should have a name and a value. For example, you might add a property with a name of `client` and a value of `smith`.

The properties that you add to uploaded documents enhance the Display Documents capability of the control. All properties that you add to uploaded documents can be used to select the documents to display. See the preceding procedure for more information about using properties to display particular documents.

- Save the Coach and then run the service or BPD to test your configuration.

To see an example of how these controls appear to end users, see the *Working with documents* section in the *Lombardi Process Portal Guide* or online help.

Embedding Lombardi reports in a Coach

Lombardi reports provide valuable information about your business processes. (See [Creating and configuring reports](#) for more information.) The data available in a Lombardi report can help process participants make decisions regarding tasks and task assignments. For this reason, you may want to embed Lombardi reports in your Coaches so that end users can make informed decisions with the business data that they need directly available.

To embed a Lombardi report in a Coach:

- Open the service that contains the Coach that you want to work with and then click the Coaches tab.
- Drag a Report control from the palette onto the design area.
- While the Report control is selected, click the Presentation option in the properties.
- Under Report Settings, click the **Select** button to choose the Lombardi report that you want to embed.



You can click the **New** button to create a new report as described in [Creating and configuring reports](#).

- In the Page to Display field, use the drop-down list to choose the report page that you want to display in your Coach.
- In the Filter Parameters area, click the **Add** button to choose parameters by which you want to filter the report results. Provide the appropriate value in the Value column as shown in the following example:

The screenshot shows the Properties window for a Report control. The 'Report Settings' section includes a 'Report' field with a dropdown menu showing 'PM Process Analysis' and a 'Page To Display' dropdown menu showing 'Scoreboard Summary'. The 'Filter Parameters' section contains a table with columns for 'Name' and 'Value', and buttons for 'Add' and 'Remove'.

Name	Value
fromDate	01:01:07
toDate	01:01:08

7. Click the Report option in the properties. In the Common area, you can provide documentation for the report and add a label.
8. In the Behavior area, you can provide a unique control ID for the report and adjust the number of rows and columns that the report spans within the Coach.
9. Save the Coach and then run the service to ensure the report is displayed properly.

Customizing Coaches

In a typical Lombardi deployment, Coaches are highly customized to render a particular look and feel. For example, you may want to customize Coaches to use corporate logos and colors. In many cases, you can meet your customization requirements by configuring the presentation and visibility properties available in the Designer interface. Read the following topics to learn more about customizing Coaches:

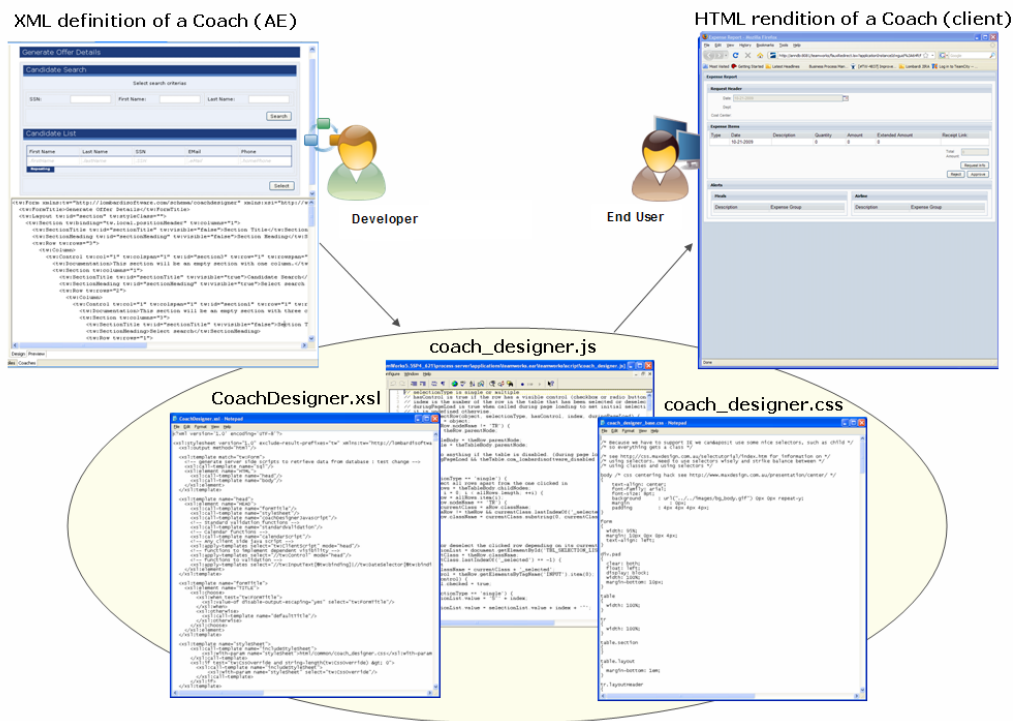
To learn...	See...
How Coaches are rendered at run time	How Coaches work
Include corporate branding and other graphic customizations in Coaches	Adding custom images to a Coach
Create a CSS override for either the label of a Coach control or for the control itself	Overriding CSS styles for selected controls and fields
Use your own Cascading Style Sheet (CSS) to customize the look and feel of an entire Coach	Specifying a custom CSS for a Coach
Override CSS styles in a Coach control to set the maximum length (in pixels) for an Input Text control	Setting the length of input text fields
Add custom attributes to Coach sections and controls to enhance the design of your interfaces	Using custom attributes
About other customization options and considerations for all options	Best practices for Coach customizations

How Coaches work

Before you customize Lombardi Coaches, you should have a good understanding of how they work to ensure that your customizations produce the desired results. The runtime rendering of a Coach involves the following key technologies:

XML	The design of a Coach is described in Extensible Markup Language (XML). As you drag sections and controls to a Coach, Lombardi automatically generates the XML definition of the Coach. You can view the XML while you're building a Coach by clicking the Code View icon in the toolbar as described in Understanding the Coach designer .
XSLT	The Extensible Stylesheet Language Transformation (XSLT) transforms the XML definition to the runtime HTML form. The XSL renders a server-side-scriptlet that is executed to generate the HTML.
HTML	The client (Web browser) renders the HTML that the Coach generates from its XML definition through XSLT processing.
CSS	The Cascading Style Sheet (coach_designer.css in the following image) instructs the client how to render the HTML output.
JavaScript	JavaScript provides the methods and functions that implement runtime Coach features, such as dynamically adding rows to a table or governing the visibility of Coach controls. JavaScript that is embedded in the XML definition of a Coach is evaluated by the runtime engine before it is rendered to HTML by the Web browser client. Both client-side and server-side JavaScript is used to render Coaches.

The following image shows how Coaches are generated:



If you have the required technical expertise, you can use the following methods to customize Lombardi Coaches:

- Override and customize Cascading Style Sheets (CSS)
- Edit the XML definition
- Edit the JavaScript files
- Edit the XSL transformation rules

The following topics describe some of the customization options most commonly used for Lombardi Coaches. If the need arises to directly alter XSL or JS files, you should be aware of the implications of making such changes as described in [Best practices for Coach customizations](#).

Adding custom images to a Coach

Coaches can include corporate branding and other graphic customizations. When you need to add custom images, add the necessary files to your process application as managed assets. See [Managing external files](#) for more information. When you add an image as a managed file, you can simply drag the image from the library in the Designer to the design area of an open Coach. Using this method, your images are included when you install snapshots of process applications on servers in test, production, or other environments.

Overriding CSS styles for selected controls and fields

You can override Cascading Style Sheet (CSS) styles to customize the appearance of a Coach. You can create a CSS override for either the label of a Coach control or for the control itself as described in the following procedure:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. In the design area, click to select the Coach control that you want to customize.
3. Click the Customization option in the properties.
4. Click the **Add Class** button.
5. Under Class Details, type a name in the Class Name text box.

By default, this field includes the name, `class`. You need to replace this text with the class name that you want.

6. To override the CSS styles for the label of the Coach control, click the **Create label override** button.

This creates a `.class_name .twLabel { }` CSS class, which is stored directly within the Coach and can be accessed as described in the following step.

7. In the design area, click on the top-level section of the Coach (named `Lombardi Coach` by default) and then click the CSS option in the properties.
8. In the CSS text box, type your CSS override settings. For example, the following CSS override changes the color of the label text to red and the typeface of the label text to bold:

```
.class_name .twLabel { color:red; font-weight:bold; }
```

9. To override the CSS styles for the Coach control itself, go back to the Customization properties for the selected control and click the **Create control override** button.

This creates a `.class_name .twControl { }` CSS class.

10. In the design area, click on the top-level section of the Coach (named `Lombardi Coach` by default) and then click the CSS option in the properties.
11. In the CSS text box, type your CSS override settings. For example, the following CSS override changes the typeface of the text in the control to italic:

```
.class_name .twControl { font-style:italic; }
```

12. Save the Coach and then run the service to test the CSS overrides.

Specifying a custom CSS for a Coach

You can use your own Cascading Style Sheet (CSS) to customize the look and feel of an entire Coach. The follow procedure describes how to specify a custom style sheet for a Coach:

1. Add the CSS file that you want to use to your current project or to a referenced toolkit as described in [Managing external files](#).
2. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
3. In the design area, click on the top-level section of the Coach (named `Lombardi Coach` by default) and then click the Coach option in the properties.
4. Click the **Select** button and choose the CSS file that you added in step 1. (You can click the **New** button and add a new CSS file as described in [Managing external files](#).)

5. Save the Coach and then run the service to test the CSS overrides.

If you want to override the styles for all Coaches, you can import your custom CSS into the `coach_designer.css` file (located in the `[Lombardi_home]/tw_web_files/html/common` directory). To override the styles for all Coaches, add the import statement `@import url("myCustom.css");` to the `coach_designer.css` file (where `myCustom.css` is the name of your custom CSS file). Then add the custom CSS to the `[Lombardi_home]/tw_web_files/html/common` directory in all Lombardi environments.

Setting the length of input text fields

Overriding CSS styles in a Coach control also enables you to set the maximum length (in pixels) for an Input Text control. Setting the maximum length for an Input Text control does not restrict the number of characters that an end user can type into the text box.

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. In the design area, click to select the Coach control that you want to customize.
3. Click the Customization option in the properties.
4. Click the **Add Class** button.
5. Under Class Details, type a name in the Class Name text box.

By default, this field includes the name, `class`. You need to replace this text with the class name that you want.

6. To override the CSS styles for the Coach control, click the **Create control override** button.

This creates a `.class_name .twControl { } CSS class.`

7. In the design area, click on the top-level section of the Coach (named `Lombardi Coach` by default) and then click the CSS option in the properties.
8. In the CSS text box, type the CSS override settings to specify the maximum input length (in pixels) for the control's input text box. For example, the following CSS override sets the length to 110 pixels:

```
.class_name .twControl { width:110px; }
```

9. Save the Coach and then run the service to test the CSS overrides.

Using custom attributes

You can add custom attributes to Coach sections and controls to enhance the layout of the interfaces that you create for end users. For example, you can add a custom attribute to enable end users to expand and collapse a section in a Coach as described in the following procedure:

1. Open the service that contains the Coach that you want to work with and then click the Coaches tab.
2. In the design area, click to select the Coach section that you want to customize.



This example requires a Coach section with a visible title that is nested within another section.

3. Click the Customization option in the properties.
4. Click the **Add Attribute** button.
5. Type `showHide` in the Name text box.

By default, this field includes the name, `key`. You need to replace this text with the key name that you want.

6. To make the section collapsible, type `true` in the Value text box.

By default, this field includes the text, `value`. You need to replace this text with the value that you want.



To make the section collapsible and collapsed by default, type `hidden` in the Value text box.

7. Save the Coach and then run the service to test the custom attribute.

The custom attributes that you add when building a Coach pass information to the XSL about how the page should be rendered. When you switch to the Coach XML view, you can see any name-value attributes in the XML. Lombardi provides the following attributes that you can use just as the `showHide` attribute is used in the preceding steps:

Attribute	Description	Applies to	Example value
<code>cssStyle</code>	Overrides a CSS style	Input Text (or equivalent)	<code>width: 100px</code>
<code>cssClass</code>	Overrides a CSS class (be sure to include the standard CSS class if you want the default functionality as well)	Input Text (or equivalent)	<code>inputText important</code>
<code>height</code>	Enables scrollable tables and sections (the table or section will always be the given height, even when there is not enough data to fill it. The header of the table scrolls with the data.)	Table or Section	<code>200px</code>
<code>width</code>	Sets the width of a table column	Cell	<code>75%</code>
<code>precision</code>	Displays numbers with the specified number of digits after the decimal	InputText or Output Text	<code>2</code>
<code>symbolPre</code>	Displays the given symbol before a value	InputText	<code>\$</code>
<code>symbolPost</code>	Displays the given symbol after a value	InputText	<code>%</code>
<code>position</code>	Specifies the location of a label, such as on top (the label must be visible)	Label	<code>top</code>

You can create completely new attributes and extend the capabilities of the Coach Designer by adding XML attributes directly to a Coach's XML. When you add attributes to a Coach's XML, you need to customize the Coach XSL to support the new attribute.

Best practices for Coach customizations

In general, you should customize Coaches using the mechanisms available in the Designer interface, such as using a custom CSS by overriding the CSS for a particular Coach or control as described in the preceding topics. If the overrides available in the Designer do not meet your needs, you can use the `cssStyleOverride` attribute, which requires a moderate comfort level with Extensible Markup Language (XML). To use a custom CSS for all Coaches, you must access and edit the default Coach CSS located in the Lombardi installation directory. For instructions, see [Specifying a custom CSS for a Coach](#).

If the CSS customization options are not a viable solution to meet your design requirements, you can edit the `CoachDesigner.xsl` transformation file to customize the rendered HTML. If you edit the `CoachDesigner.xsl` file and you notice that those changes are not being applied to the Coaches under development, open your Coach, click on the top-level section of the Coach (named Lombardi Coach by default), click the Coach option in the properties, and then click the **Transform Coach XML** button.

If you decide to modify the XSLT or JS files, you need to be aware of the following implications:

- You should not attempt to use these methods unless you have a high comfort level with Extensible Stylesheet Language (XSL) and JavaScript.
- The XSL transformation (XSLT) and JavaScript files are overwritten during the Lombardi upgrade process, without opportunity to detect or merge possible customizations. If you have made modifications to these files, you must merge your customizations manually after each upgrade.
- You must make modifications in all Lombardi environments, including development, test, production, and any others to which you will install and run processes.

Troubleshooting Coaches

If you experience problems running services that include Coaches, check for the issues described in the following table:

Issue	Potential cause and resolution
Dynamic values are not displayed in a Coach as expected.	You might be using null values in the variable (that is, the variable has no value versus the value 0). When initializing variables in the service that contains the Coach, assign default values to them.
Input values are not captured as expected at run time.	This can happen when using multiple sections in a Coach. Each section is its own HTML form; submitting data in a section does not submit data in another section, unless it is a nested section. Always design your Coaches with this rule in mind.
Unexpected behavior of nested visibility rules at run time.	An example of nested visibility rules is described in Displaying a control based on the input value of another control . If possible, design your Coaches such that you can distribute the rules across more than one Coach. Another option is to use custom JavaScript to express cascading rules. An example is provided in Using a custom script to control visibility .
A configured onClick event for a button does not take place at run time.	Ensure that the onClick event is assigned to the individual button instead of a Button Group control.

Using nested processes

Use nested processes to encapsulate activities that are related to each other within a parent process. Using nested processes enables you to manage the complexity of any business process while retaining the high-level view of the overall process represented in the parent process definition. Each activity in a BPD can have a nested process attached to it, and each activity in the nested process can have a nested process attached to it, and so forth.

When an activity that is implemented by a nested process is triggered at run time, the attached nested process is executed. After the nested process has run to its completion, the parent process resumes execution.

When you implement an activity using a nested process, the BPD diagram displays the activity with the icon shown in the following image:



The following procedure illustrates how an activity can be implemented using a nested process:

1. Open the parent business process definition (BPD) in the Designer.

If you followed the steps in [Basic modeling tasks](#) to create the Expense Reimbursement BPD, open it to use as the parent process.

2. Click the Approval activity in the BPD diagram.
3. Click the Implementation tab in the properties.
4. Under Implementation, select the Lombardi Nested Process option from the drop-down menu.
5. Click the **New** button to create a nested process.

If the nested process has already been created, you can click the **Select** button to choose the BPD from the library.

6. In the New Business Process Definition dialog, type a name for the BPD and click the **Finish** button.

For this example, type Expense Review.

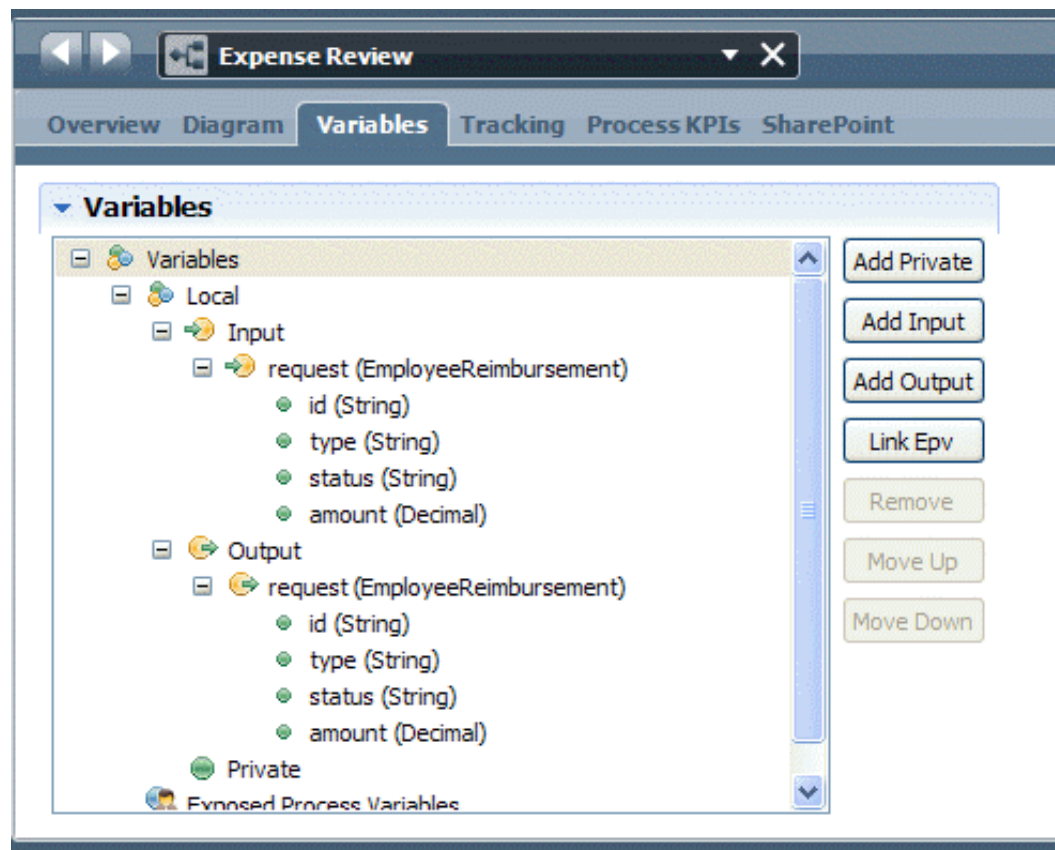
Using a nested process ensures that the review and approval step in the parent process is flexible enough to meet the needs of a growing organization. For example, using a nested process ensures that you can add lanes with additional participant groups should the review need to grow into a larger multi-stage, multi-organization implementation.

The plan for the new Expense Review BPD is to include activities where first senior management and then HR can review the submitted expenses. To implement these activities, you can build Human services with Coaches as described in [Building a Human service](#).

7. Click the Variables tab in the properties to map the variables from the parent process to the nested process.

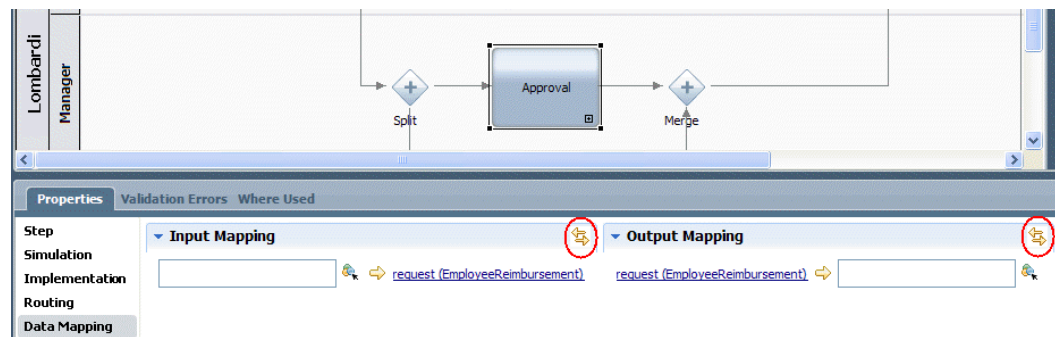
8. Click the **Add Input** button. Adding input enables the parent process to pass variable values to the nested process.
9. Replace `Untitled` in the Name field with `request`.
10. Click the **Select** button next to Variable Type and select the `EmployeeReimbursement` type from the list. (The `EmployeeReimbursement` variable type is available only if you first implement the steps in [Adding process variables to a BPD.](#))
11. Click the **Add Output** button. Adding output enables the nested process to pass variable values back to the parent process so that execution can continue.
12. Replace `Untitled` in the Name field with `request`.
13. Click the **Select** button next to Variable Type and select the `EmployeeReimbursement` type from the list. (The `EmployeeReimbursement` variable type is available only if you first implement the steps in [Adding process variables to a BPD.](#))

The following image shows the variable mapping from the parent process to the nested process:



14. Go back to the BPD for the Expense Reimbursement parent process.
15. Click the Approval activity to select it.
16. Click the Data Mapping tab in the properties.

Because you already created the input and output variables for the nested process, the Data Mapping tab for the Approval activity in the parent process includes those variables as shown in the following image:



- Under Input Mapping, click the auto-map icon highlighted in red in the preceding image and then click the auto-map icon in the Output Mapping section.

The Designer shows `tw.local.request` as the input and output variables for the Approval activity.

- Save these changes and then you can continue by adding the activities and building the implementations for the nested process if you want to test the implementation.

For more information about mapping variables, see [Managing and mapping variables](#). See the following section if you want to call a nested process dynamically.

Calling a nested process dynamically

When you choose to use a nested process as the implementation for an activity, an advanced option in the Implementation properties enables you to supply a predefined variable to dynamically call one of many nested processes, depending on your needs. To use the dynamic option for a nested process, you must first complete the following tasks:

- Create a variable in the parent process of type `String` to hold the name of the nested process that you want to run. (The variable must be of type `String`.) Your parent process must also include the logic to determine the value of this variable at run time. For example, the parent process can include logic to set the value of this variable based on user input.
- Establish the input and output variables for each potential nested process so that no matter which nested process is called, the parent process will run as expected. To meet this requirement, the variables in all potential nested processes must be the same. You can follow the steps for mapping variables from the parent process to the nested process in [Using nested processes](#) to meet this requirement.

Follow these steps to configure an activity to dynamically call one of many potential nested processes:

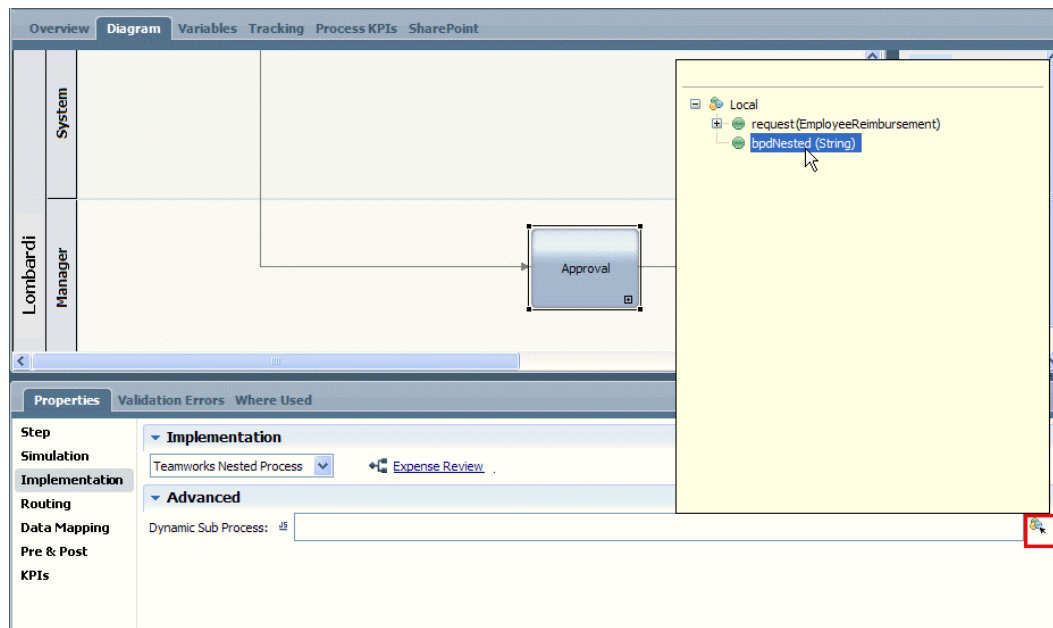
- Open the parent business process definition (BPD) in the Designer.
- Click the activity that you want in the BPD diagram.
- Under Implementation, select the Lombardi Nested Process option from the drop-down menu.
- Click the **Select** button to choose one of the predefined nested BPDs from the library.

You must initially select one of the predefined nested BPDs for the dynamic configuration to function properly.

- Click the Data Mapping tab in the properties.

Because you already created the input and output variables for the nested process, the Data Mapping tab for the activity in the parent process includes those variables.

- Under Input Mapping, click the auto-map icon in the upper-right corner and then click the auto-map icon in the upper-right corner of the Output Mapping section.
- Click the Implementation tab in the properties.
- Click the indicator next to the Advanced section heading to expand the section.
- Click the variable icon next to the Dynamic Sub-Process field to choose the previously defined variable that provides the name of the selected process as shown in the following image:



At run-time, the value of this variable cannot be null and it must exactly match the name of an existing BPD.

- Save the configuration.

Using embedded JavaScript

You can use JavaScript to implement activities in your business process definitions (BPDs). Lombardi provides full JavaScript support, including calling and creating JavaScript functions, loops, setting variables, and so on. The following resources are available for JavaScript implementations:

<p>Lombardi JavaScript API</p>	<p>Provides methods and objects to enable access to Lombardi functions and features. When implementing JavaScript in Lombardi, the type-ahead feature automatically completes your entries so that you are aware of the available objects. For example, if you type <code>tw.system</code> in a JavaScript field, Lombardi returns a list of system functions that you can choose from. To learn more about the JavaScript API, see the reference information posted to the WebSphere Lombardi Edition Documentation Wiki:</p>
--------------------------------	--

	http://wiki.lombardi.com/display/tw7/Teamworks+7+JavaScript+API . (You must have an account for the IBM Customer Support for Lombardi site to access the wiki.)
Managed JavaScript files	When you add a JavaScript (.js) file to be managed as part of your project, you can use the functions and variables within the file in the scripts and scriptlets that you develop throughout your Lombardi project. To learn how to add .js files to your project, see Managing external files .



While Lombardi supports using JavaScript to implement an activity, in most cases it is best to create a service that includes a Server Script component and then use the service to implement the activity.

Follow these steps to implement an activity using embedded JavaScript:

1. Click to select the activity that you want in the BPD diagram.
2. Click the Implementation tab in the properties.
3. Under Implementation, select the Embedded JavaScript option from the drop-down menu.
4. Type or copy the script into the text box.



You can press Ctrl+F to use find and replace functionality in the JavaScript editor, including the ability to replace all occurrences.

5. Save your changes to the activity.

Using external activities

External activities enable you to create Business Process Definitions (BPDs) that include activities that are handled by systems outside of Lombardi. For example, you can model an activity that is executed by an Eclipse RCP or Microsoft .NET application.



To create external activities in Lombardi Authoring Environment, you must enable Lombardi Developer Capability and Lombardi Advanced Features. Enable these capabilities by selecting **File > Preferences > Capabilities** from the main menu.

To use an external activity to implement a step in a business process definition (BPD), you must complete the following tasks in the order shown:

Task	See...
1. Build a custom application using the Lombardi Web API to execute an activity (step) in a BPD	Building a custom application to implement an activity
2. Create an external activity in the Designer view in Lombardi Authoring Environment	Creating an external activity
3. Select the external activity as the implementation for an activity (step) in a BPD	Selecting an external activity to implement

Building a custom application to implement an activity

If you want to build a custom application to implement an activity (step) within a process, you must use the Lombardi Web API to enable your custom application to interact with Lombardi. For example, the Lombardi Web API provides operations that enable you to pass variables from a Lombardi BPD to a custom

application and then back to the BPD for continued processing. For more information, see *Lombardi Developer's Guide*.

Creating an external activity

When you create an external activity in the Designer view in Lombardi Authoring Environment, you need to know the properties to use to identify and execute the custom application. If you did not build the custom application, you need to coordinate with the developers to ensure that you provide the appropriate properties in Lombardi Authoring Environment.

1. In the Designer view, click the plus sign next to **Implementation** and select **External Activity** from the list of components.
2. Supply a descriptive name for the new external activity.
3. Click **Finish**.
4. In the Common section of the External Activity dialog, optionally provide a description for the activity in the Documentation text box.
5. In the Custom Properties section, specify the properties to identify and execute the external activity.

For example, for an external Eclipse RCP application, you might add custom properties to pass the Java Class name of the form to use for the activity or an application-specific identifier to look up the implementation by another means. Alternatively, you might use the external activity name or system ID to find the implementation.



You can add custom properties to pass static metadata about the activity to the external application. For dynamic data, which would be different for each process instance or environment, use the Parameter Details section of the dialog as outlined in the following step.

6. In the Parameter section of the dialog, add the parameters for the activity by clicking **Add Input** or **Add Output**.

For example, if the external activity provides an interface in which a manager can either approve or reject an expense report, it might include input parameters for the expense report data and output parameters for the decision that the manager makes and the justification for his decision.

Be sure to account for all process data that the external activity requires to complete successfully and also for any data required from the external activity by subsequent activities.

7. Click **Save** in the main toolbar.

Selecting an external activity to implement

The following steps describe how to select a custom application as the implementation for an activity in a BPD:

1. Open a business process definition (BPD) in the Designer.
2. In the BPD diagram, click the activity that you want to implement using a custom application.
3. Click the Implementation tab in the properties.

4. Under Implementation, select the External Activity option from the drop-down menu.
5. Click the **Select** button to choose an external activity from the library.

If the external activity has not been created, click the **New** button and follow the steps in [Creating an external activity](#) to create a new external activity.

6. In the Task Header section, specify the following properties:

Subject	Type a descriptive subject for the task that is generated in Lombardi Process Portal when you run the BPD. You can also use the Lombardi embedded JavaScript syntax (for example, <code><#=tw.local.mySubject#></code>) to express the subject.
Narrative	Type an optional description. You can also use the Lombardi embedded JavaScript syntax to express the narrative.



For the following properties (in the Priority Settings section) you can click the JS button for an option if you prefer to use a JavaScript expression with predefined variables to establish the priority settings.

7. For the Priority field, click the drop-down list to select one of the default priority codes: Very Urgent, Urgent, Normal, Low, or Very Low.
8. For the Due In field, you can enter a value in the text box and then choose Minutes, Hours, or Days from the drop-down list. (When you choose Days, you can use the text box after the drop-down list to include hours and minutes in your specification.)

You also have the option of using the variable selector next to the text box to choose an existing variable from the library. At run-time, the variable should reflect the value that you want for the time period. Be sure to select the option that you want from the drop-down list: Minutes, Hours, or Days.

9. For the Time Period field, click the drop-down list to select one of the options. For example, select **24x7** if you want 24 hours a day, seven days a week to be the time period in which the resulting tasks from the current activity can be due.



You can leave the Schedule, Timezone, and Holiday Schedule fields set to (use default). If you do, the work schedule specified for the BPD is used. See [Setting the work schedule for a BPD](#) for more information.

10. For the Time Zone field, click the drop-down list to select the time zone that you want to apply to the tasks that result from the current activity. For example, you can select **US/Pacific** for end users who work in California.
11. For the Holiday Schedule field, you can leave the setting at (use default) as described in the preceding note or you can click the JS button if you prefer to use a JavaScript expression. Each Holiday Schedule is made up of a list of Dates.

If you choose JavaScript, you can enter either a String (or String-generated JavaScript) or JavaScript that returns a `TWHolidaySchedule` variable. If you use a String, then Lombardi looks up the holiday schedule by name according to those rules. If you use a `TWHolidaySchedule` variable, then Lombardi assumes that the holiday schedule is filled in appropriately. (Go to the System Data toolkit and open the `TWHolidaySchedule` variable to view its parameters.)

12. Click the Data Mapping tab in the properties.

Because you added the input and output parameters for the external activity when you created it, the Data Mapping tab for the activity in the BPD should include those parameters.

Under Input Mapping, click the auto-map icon in the upper-right corner, and then click the auto-map icon in the upper-right corner of the Output Mapping section. For more information about mapping variables, see [Managing and mapping variables](#).

13. Save the implementation.

Integrating with other systems

Lombardi supports both outbound and inbound integration as described in the following table:

Integration type	Description	Required Lombardi components	See...
Outbound	Lombardi communicates with an external system to retrieve, update, or insert data.	Integration service	Creating outbound integrations
Inbound	An external system calls into Lombardi to initiate a service.	Undercover Agent and Web Service	Creating inbound integrations

Creating outbound integrations

For outbound integrations, you need to create a Lombardi Integration service. Integration services can include either a Web Service Integration component or a Java Integration component. The following table describes how each integration component functions:

Integration component	Description	See...
Web Service Integration	Uses a Simple Object Access Protocol (SOAP) connection to access objects from a web service over the Internet. A Web Service Integration component hides the complexity of the underlying WSDL, SOAP request, and SOAP response and also converts inputs into the appropriate XML and outputs into the appropriate Lombardi variables.	Using the Web Service integration component in an Integration service
Java Integration	Calls methods from a Java class and interfaces with most third-party Java APIs, thus supporting a variety of integration scenarios.	Using the Java Integration component in an Integration service



When integrating with an external database, you can use the out-of-the-box Lombardi SQL Integration services available in the Lombardi System Data Toolkit. See [Using Lombardi SQL Integration services](#) to learn how to access these services.

When considering what kind of Integration component to use for outbound integrations, think about the available integration methods and protocols. SOAP integrations tend to be very easy to build and are extremely useful, especially if you are not passing volumes of information. Java integrations give you access to the features of the Java language, including published Java libraries and APIs.

Using the Web Service integration component in an Integration service

You should build an Integration service when you want to integrate with an external system to retrieve, update, or insert data in order to complete a task (i.e., outbound integration). For more information about the types of services you can build and the components available, see [Building services](#).

Integration services enable outbound integration as described in [Integrating with other systems](#). If the implementation for an activity requires that you access objects from a web service over the Internet, you can use the Web Service integration component as described in the following topics:

Topic	Description
Building an Integration service	Provides step-by-step procedures for building a sample Integration service that uses an existing Web service; also includes instructions for creating a Human service that implements the Integration service as a nested service.
Understanding outbound Web Service security, header, and other options	Describes several options that you can specify for Web Service integration components, such as the types of authentication available, as well as helpful reference information.

Understanding outbound Web Service security, header, and other options

Read the following topics to learn about all options available when using a Web Service Integration component for outbound integrations in a Lombardi Integration service. Troubleshooting tips and additional resources for building outbound Web Service integrations are also covered.

Specifying URIs

When you specify the URI for a Web Service integration component, relative URIs are also supported. When you use a relative URI, it must be relative to the current working directory of the application server: `[Lombardi_home] \ [process-center | process-server]`

You can store shared XML schemas within the `[Lombardi_home] \ process-center | process-server \ webservices \ ws` directory. For those schemas to be resolved properly, you must edit the `[Lombardi_home] \ [process-center | process-server] \ webservices \ xml-catalog.xml` file and add an entry to map the XML schema target namespace to the XML schema file as follows:

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
<group id="lombardi.teamworks.ws" prefer="public" xml:base="ws/">
<public publicId="http://www.w3.org/2001/XMLSchema" uri="XMLSchema.xsd" />
<system systemId="http://www.w3.org/2001/XMLSchema" uri="XMLSchema.xsd" />
</group>
</catalog>
```

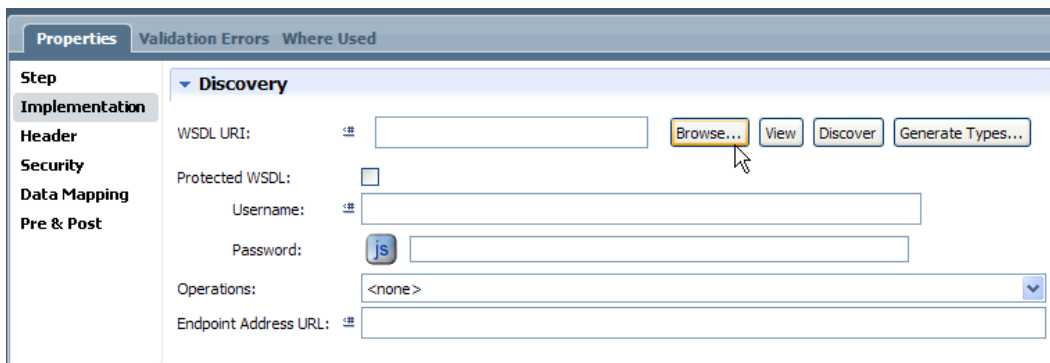
Specifying a runtime endpoint address URL

In the **Endpoint Address URL** field in the Implementation properties for a Web Service integration component, you can specify a URL that points to a different instance of the Web service. Or this address can point to a proxy server, enabling you to monitor network traffic.

Using the UDDI Explorer

Web Service integration components use a Simple Object Access Protocol (SOAP) connection to discover the ports and parameters from the Web Service Definition Language (WSDL). The Universal Description Discovery and Integration (UDDI) Explorer in Lombardi Authoring Environment enables you to use a UDDI registry with a list of Web services and gives you information about the function of each Web service.

1. To use the UDDI Explorer, click the Browse button in the implementation properties for a Web Service integration component as shown in the following image:



2. When the Explorer Wizard starts, type in the **UDDI Registry URL**.

The UDDI Explorer Wizard stores the UDDI registry URLs that you type in, enabling you to select previously used URLs from the drop-down list.

Click the **Next** button.

3. In the Name field, type a partial or complete Web service name.

You can use the % symbol as a wildcard that matches any character and returns a list of all services in the UDDI registry.

Click the **Search Services** button to generate a list of available Web services for the selected UDDI registry.

Select a Web service, and then click the **Next** button.

The wizard displays the Web service details, including the service key and the WSDL URI.

4. If you are satisfied with the WSDL URI, the service key, and the implementation, click the **Finish** button.

Lombardi populates the **WSDL URI** field in the implementation properties for the Web Service integration component.

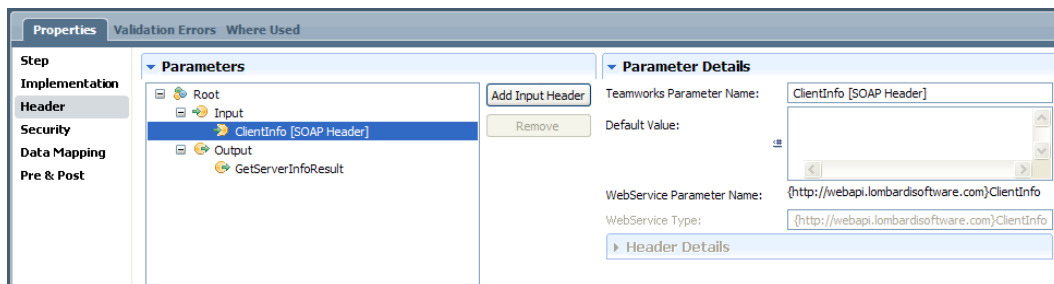
5. Click the **Discover** button to discover the WSDL.

Using SOAP headers

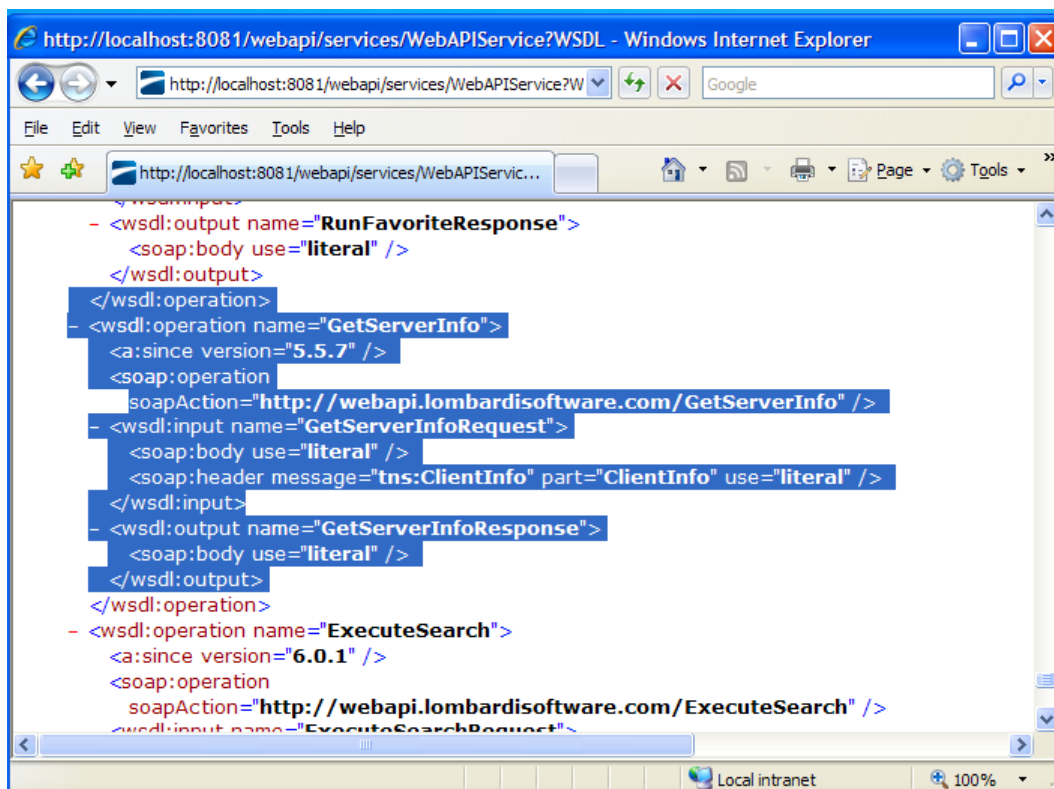
Web Service integration components use a SOAP/XML connection to call out to Web services by discovering the ports and parameters from the WSDL, enabling Lombardi to access objects from a Web service over the Internet. SOAP/XML data can also be passed through firewalls so that applications can exchange information easily without causing security risks. SOAP messages are exchanged in a request/response format. When Lombardi sends a request to a Web service, the Web service returns the requested values. These values are specified in a SOAP message, which is a block of XML code that contains several elements.

All SOAP messages must contain a SOAP envelope element, which identifies the XML code as a SOAP message. Some WSDLs require that SOAP headers also be passed with each request. A SOAP header is an element in a SOAP message that resides in the message's SOAP envelope and provides detailed information about the SOAP message. For WSDLs that require SOAP headers to be passed with each request, Lombardi supports passing such headers for SOAP operations. SOAP headers for a selected WSDL operation are identified in brackets in the Parameters section of the Header properties, as shown in the

following image. There are two types of SOAP headers: headers that are directly specified as part of the SOAP binding and headers that are not.



SOAP headers that are part of the SOAP binding are indicated specifically by the use of `<soap:header>` in the `<wsdl:input>` and `<wsdl:output>` elements in the code that you see when you open the WSDL URI in a Web browser. For example, the following figure shows the `GetServerInfo` operation in `http://localhost:8081/webapi/services/WebAPIService?WSDL`. You can see where the `<soap:header>` element is used in this operation. When Lombardi sends a request to the server that hosts the Web service, the SOAP message includes the SOAP header. For example, the SOAP header used for the `GetServerInfoRequest` input is `ClientInfo`.



SOAP headers that are not part of the SOAP binding typically are not specified in the WSDL. There is only one SOAP header section in any SOAP request. A SOAP envelope, which is passed across the Internet as the request and response, consists of a header section and a body section. SOAP headers can be input, output, or input/output, as shown in the following image, and they do not have to be specified in the WSDL. For example, you can see the SOAP headers in the `tw-outbound_webservices.log` and `tw-inbound_webservices.log` files in the `[Lombardi_home] \ [process-center | process-server] \ logs` directory.

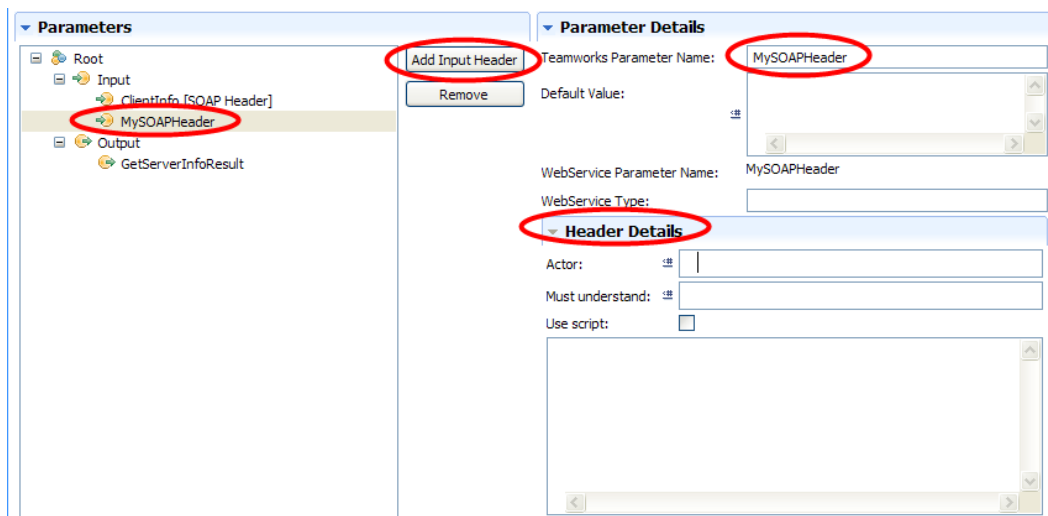
```

<wsdl:message name="EchoOutputHeader" >
  <wsdl:part name="OutputHeader" element="tns:OutputHeader" />
</wsdl:message>
<wsdl:portType name="TestInputOutputHeadersSoap">
  <wsdl:operation name="Echo">
    <wsdl:input message="tns:EchoSoapIn" />
    <wsdl:output message="tns:EchoSoapOut" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="TestInputOutputHeadersSoap"
  type="tns:TestInputOutputHeadersSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="Echo">
    <soap:operation
      soapAction="http://test.lombardisoftware.com/webservices/TestIn
      putOutputHeaders/Echo" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
      <soap:header message="tns:EchoInputHeader" part="InputHeader"
        use="literal" />
      <soap:header message="tns:EchoInputOutputHeader"
        part="InputOutputHeader" use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
      <soap:header message="tns:EchoInputOutputHeader"
        part="InputOutputHeader" use="literal" />
      <soap:header message="tns:EchoOutputHeader"
        part="OutputHeader" use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="TestInputOutputHeadersSoap12"
  type="tns:TestInputOutputHeadersSoap">

```

Lombardi also supports passing arbitrary SOAP headers using the Web Service integration component. This enables you to add a new SOAP header to a SOAP request without having to manually add the SOAP header to the WSDL. To define a new SOAP header parameter, click the **Add Input Header** button, as shown in the following image. The new SOAP header is added to the Input tree in the Parameters section. Type a name for the new header in the **Parameter Name** field. You can manually specify a Default Value and a Web Service Type for the header parameter. In the Header Details section (available only for SOAP header parameters), you can also specify the following optional properties for the SOAP header parameter:

Property	Description	Syntax
Actor	Specify a SOAP actor attribute.	soap:actor="URI"
Must understand	Specify a SOAP mustUnderstand attribute.	soap:mustUnderstand="0" soap:mustUnderstand="1"
Use script	When you enable this edit box, you can manually write a script that generates the SOAP/XML content for the SOAP header. If you select this option, the Default Value, Web Service Type, Actor, and Must understand fields are disabled.	N/A



To delete a header from the Input tree, select the header and then click the **Remove** button.

Serializing Lombardi objects for use in Web services

Authors can add metadata to Lombardi objects to control how those objects are serialized into XML elements in a SOAP request for Web services. When sending Complex types as input parameters to Web services, Lombardi often needs hints as to how to structure the data. These methods are built on the structure because a structure is, by definition, a Complex type.



Serializing Lombardi objects for use in Web services is primarily for advanced users, and is now required only when using Record objects with Web services instead of the generated types.

The following methods have been added to the `tw.object.Record()` object to assist in forming the SOAP request:

- `setSOAPElementName(string elementName)`
- `setSOAPElementNS(string namespace)`
- `defineSOAPProperty(string propertyName, string schemaName, string typeName, boolean isAttribute, string namespace)`

Example:

You are passing an object of type `NameUpdateRequest` as an argument to a Web service. This object is defined in the namespace `http://www.lombardisoftware.com/schemas/NameUpdateRequest`. The `NameUpdateRequest` object contains two properties, `First` (string) and `Last` (string).

Following is example code to generate the XML that is part of the SOAP call:

```
<#
out = new tw.object.Record();
out.setSOAPElementNS("http://www.lombardisoftware.com/schemas/
NameUpdateRequest");
out.setSOAPElementName("NameUpdateRequest");
out.defineSOAPProperty("First", "http://www.w3.org/2001/
XMLSchema", "string", false, "");
```

```
out.defineSOAPProperty("Last", "http://www.w3.org/2001/
XMLSchema", "string", false, "");
out.First = "John";
out.Last = "Smith";
#>
```

This generates the following XML element:

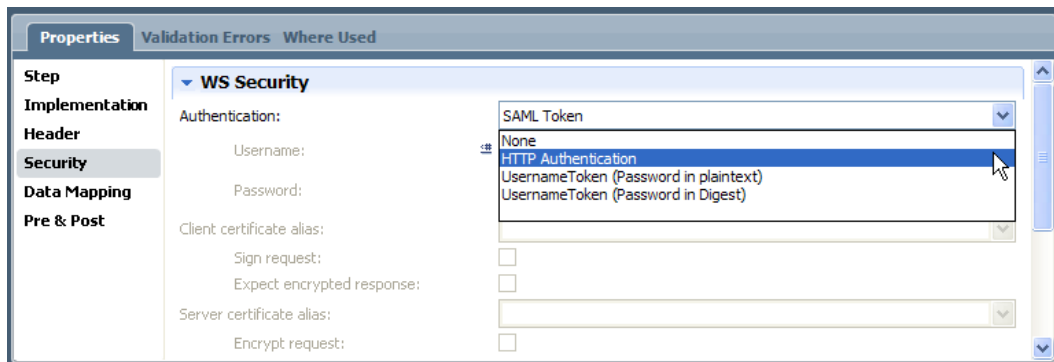
```
<NameUpdateRequest xmlns="http://www.lombardisoftware.com/schemas/
NameUpdateRequest">
<first xmlns="">John</first>
<last xmlns="">Smith</last>
</NameUpdateRequest>
```

Calling a Web service that requires authentication

Lombardi supports the following runtime authentication mechanisms that do not require client certificates:


- Protected WSDL
- HTTP Basic Authentication (described in [RFC 2627](http://tools.ietf.org/html/2617) [http://tools.ietf.org/html/2617])
- UsernameToken Authentication (described in [Web Services Security UsernameToken Profile 1.0](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf) [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf])

For Protected WSDL authentication, you can simply enable the Protected WSDL check box in the implementation properties for the Web Service integration component, and then provide the user name and password in the provided fields. The other authentication options are available in the security properties for Web Service integration components as shown in the following image:



The following table describes the information that you must provide for each option:

Option	Description
HTTP Authentication	Requires a user name and password. Lombardi never stores the password in plain text in its database or export files, and no plain text passwords are required in Lombardi configuration files.
UsernameToken	When using UsernameToken authentication in Lombardi, a user name and password are passed to a Web service in the header of the SOAP request. UsernameToken allows for different algorithms and formats for passing the password. Lombardi supports passwords in the following formats: Plain text and Digest

Option	Description
	 <p>UsernameToken authentication is an alternative to using a certificate on the client. If you select either of the UsernameToken options, the Sign request option is disabled. If you select the Sign request, the UsernameToken option is disabled.</p> <p>The specification for UsernameToken authentication with Password in Digest describes two optional elements that can be supplied:</p> <ul style="list-style-type: none"> • <code>wsse:Nonce</code> • <code>wsu:Created</code> <p>The Lombardi implementation of this standard always provides <code>wsse:Nonce</code> and <code>wsu:Created</code>. This is compatible with the behavior of Microsoft WSE 2.0 Toolkit when using UsernameToken digest authentication.</p>

Calling a Web service that is secured by SSL

Lombardi supports standards for calling Web services that are secured by the SSL protocol. The procedure for modeling the integration between Lombardi and a Web service that is secured by SSL is generally the same as for non-secured Web services, if the certificate comes from a trusted certificate authority.

However, if the certificate that is used for SSL is not from one of the certificate authorities that Java recognizes by default, you need to install the certificate in the SSL certificate store that Java uses. If not, you get an error stating `No trusted certificate is found` when you click the Discover button in the implementation properties for the Web Service integration component.

Calling a Web service that uses client and server certificates

If you call a Web service that uses X509 client and server certificates, you must provide the certificate aliases when you select the HTTP Authentication option in the security properties for a Web Service integration component.



A developer or administrator must add the certificates to the Lombardi keystore and make appropriate configuration changes.

When you select the HTTP Authentication option in the security properties for a Web Service integration component, the following fields are available for certificate aliases:

Client certificate alias	Identify the certificate that can be used to sign a Web service request or to decrypt a Web service response.
Server certificate alias	Identify the certificate that can be used to encrypt Web service requests or to verify signatures on Web service responses.

Troubleshooting outbound Web Service integrations

The following table describes some common problems that you might encounter when creating services that include outbound Web Service Integration components:

Issue	Error message when you click Discover	Possible resolutions
Incorrect WSDL URI value	PARSER ERROR: Problem parsing '[path_name]WebAPIService.':The	You have incorrectly typed the URI value. Navigate to the URI using a Web browser to

Issue	Error message when you click Discover	Possible resolutions
	markup in the document following the root element must be well formed.	<p>verify that you have the correct WSDL. A common problem is that the ?wsdl argument is missing from the end of the URI.</p> <p>For file protocol URIs, the URI does not exist on disk. If you are unable to validate the location of the URI on disk, contact your network administrator.</p>
Nonexistent host	Unknown Host Exception	<p>You have incorrectly typed the host value. Navigate to the URI using a Web browser to verify that you have the correct host information.</p> <p>The server that hosts the URI is offline (not running). Contact your network administrator to determine if this is causing the problem.</p> <p>The network is experiencing connectivity problems. Contact your network administrator to determine if this is causing the problem.</p>
Authentication required for WSDL access	Runtime Exception: Unauthorized access to '[path_name]WebAPIService?WSDL'	<p>The WSDL URI is protected by an authentication mechanism. If you have permission to access the Web service, check the Protected WSDL check box, and then enter the Username and Password.</p> <p>Navigate to the WSDL URI using a Web browser and save the text of the WSDL to a file so that you can use the file location as the target WSDL URI.</p>

Recommended reading for outbound Web service integrations

To learn more about Web service standards and how they might apply to your Lombardi environment, the following Web resources are recommended:

- <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- <http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html>
- <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>
- <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>
- <http://msdn2.microsoft.com/en-us/library/77hkfh8.aspx>

Using the Java Integration component in an Integration service

You should build an Integration service when you want to integrate with an external system to retrieve, update, or insert data in order to complete a task (i.e., outbound integration). For more information about the types of services you can build and the components available, see [Building services](#).

Integration services enable outbound integration as described in [Integrating with other systems](#). If the implementation for an activity requires calling methods from a Java class, you can use the Java Integration component as described in the following procedure.



Before creating the Integration service, add the JAR file that contains the classes that you need as described in [Managing external files](#). After you add the JAR file as a server file, you can select the class and method that you want to use for your service. If the JAR files that you need are included in a Lombardi toolkit, you can add a dependency to that toolkit to access those files. See [Creating a toolkit dependency in the Designer view](#) for instructions.

1. Create an Integration service as described in [Creating a service](#).
2. Drag a Java Integration component from the palette to the service diagram and then use sequence lines to connect the component to the Start and End Events.
3. Click on the Java Integration component in the diagram and then click the Definition option in the properties.
4. Click the **Select** button next to the Java Class field to choose the JAR file and the class on the JAR file that you want to call.

The JAR files listed are the ones added as managed server files as described in [Managing external files](#). By default, the classes in the Lombardi Java package are available in the `integration.jar` file, which is included in the System Data toolkit. If your current project has dependencies on other toolkits that include JAR files, those files are also available.

5. Use the drop-down list to choose the method that you want to call on the class that you selected in the preceding step (as shown in the following image):

The screenshot displays the WebSphere Designer interface. The main workspace shows a process diagram with a 'Start' event, a 'Java Integration' component, and an 'End' event, all connected by sequence lines. The 'Properties' panel at the bottom is open to the 'Definition' tab. Under the 'Discovery' section, the 'Java Class' field is set to 'teamworks.Users' with a 'Select...' button to its right. The 'Method' field is set to 'getUser(String)' with a dropdown arrow. The 'Translate JavaBeans' checkbox is checked.

6. Enable the Translate JavaBeans check box if you want the result of the Java method that is invoked to be serialized and returned to the Integration service as an XML element. The content of the element is based on the properties of the object's class.

For example, if you choose the `teamworks.Users` class from the `integration.jar` file (Lombardi Java packages) and then select the `getUser` method with the Translate JavaBeans check box enabled, the result looks like this:

```
<userinfo type="com.lombardisoftware.core.UserInfo" description="UserInfo">
<calendarid type="com.lombardisoftware.client.persistence.common.ID" description="calendarId" />
<fullname type="java.lang.String" description="String">tw_author</fullname>
<qualifiedName type="java.lang.String" description="String">tw_author</qualifiedName>
<sendToAddress type="com.lombardisoftware.core.routing.Address" description="Address">
<name type="java.lang.String" description="String">tw_author</name>
<toGroup type="java.lang.Boolean" description="Boolean">false</toGroup>
<toUser type="java.lang.Boolean" description="Boolean">true</toUser>
</sendToAddress>
<userData type="java.util.HashMap" description="HashMap">
<entry key="Full Name" description="Map Entry">
<key type="java.lang.String" description="String">Full Name</key>
<value type="java.lang.String" description="String">tw_author</value>
</entry>
</userData>
<userId type="com.lombardisoftware.client.persistence.common.ID#NumericID" description="ID#NumericID">
<id type="java.math.BigDecimal" description="BigDecimal">2</id>
<type type="com.lombardisoftware.client.persistence.common.FOType#User" description="FOType#User">
<deleted type="java.lang.Boolean" description="Boolean">false</deleted>
<exportable type="java.lang.Boolean" description="Boolean">false</exportable>
<factoryName type="java.lang.String" description="String">com.lombardisoftware.client.persistence.UserFactory</factoryName>
<id type="java.lang.Integer" description="Integer">2048</id>
<libraryItem type="java.lang.Boolean" description="Boolean">false</libraryItem>
<name type="java.lang.String" description="String">User</name>
<tableName type="java.lang.String" description="String">LSW_USR_XREF</tableName>
</type>
</userId>
<username type="java.lang.String" description="String">tw_author</username>
</userinfo>
```



When you enable the Translate JavaBeans check box, the variable type that you select in the Integration service for the value returned from the Java method should be `XMLElement` or `ANY`.

If you do not enable the Translate JavaBeans check box, the Java method can only return objects of the following types:

<code>java.lang.String</code>	<code>java.lang.Double</code>	<code>java.lang.ArrayList</code>
<code>java.lang.Long</code>	<code>java.lang.Float</code>	<code>java.lang.HashMap</code>
<code>java.lang.Integer</code>	<code>java.lang.Boolean</code>	<code>org.jdom.Document</code>
<code>java.lang.Short</code>	<code>java.lang.Character</code>	<code>org.jdom.Element</code>
<code>java.lang.Byte</code>	<code>java.lang.Calendar</code>	<code>com.lombardisoftware.core.XMLNodeList</code> and <code>com.lombardisoftware.core.TWOObject</code>

7. Click the Variables tab for the Integration service to add any input variables that the service needs to receive and any output variables that the service needs to make available for subsequent steps in the service or BPD.
8. Click on the Java Integration component in the service diagram and then click the Data Mapping option in the properties to configure the input and output mappings for the component.
9. Nest the Integration service in another service or select it as the implementation for an activity, depending on the requirements of the overall process.

Using Lombardi SQL Integration services

To integrate with an external database, you can use the SQL Integration services available in the Lombardi System Data Toolkit. During Lombardi installation, the System Data toolkit is imported into the Process Center repository so that each process application and toolkit that you create has access to Lombardi system data. The System Data toolkit includes SQL Integration services to enable you to easily integrate with external databases.

The SQL Integration services support common database interactions, including support for parameterized queries. In addition, these services can automatically map query results directly into a desired variable type. The SQL Integration services enable you to develop implementations to:

- Read existing data from a database
- Update existing data in a database
- Write new data to a database

Plus, when passing data between Lombardi and a connected database, the SQL Integration services enable you to specify certain types of data (like integers, BLOBs, and CLOBs).

The SQL Integration services are Java-based integrations that bind to a specific method in the `teamworks.SQLConnector` Java class. Although you cannot alter the SQL Integration services, you can open them in the Designer in Lombardi Authoring Environment to see the method implemented by each one and the available input and output variables as outlined in the following procedure.

1. Open a process application in the Designer in Lombardi Authoring Environment.
2. Click the indicator next to the Toolkits category to see a list of toolkit dependencies for the current process application.
3. Click the indicator next to the System Data toolkit to see its contents.
4. Click the **Implementation** category and then double-click one of the listed SQL services.
For example, double-click the SQL Execute Statement service to open it.
5. In the service diagram, click the Java Integration component to select it.
6. Click the **Definition** option in the properties to display the Java Class and method implemented by the service.
7. Switch from the diagram view of the service by clicking the Variables tab.
8. Click on an Input or Output variable to see its details, such as its type and default values (where applicable).

To use a SQL Integration service in an implementation, you can:

- Select a SQL Integration service as the implementation for an activity as described in [Implementing activities](#).
- Nest a SQL Integration service in another service by dragging it from the library to the diagram of the parent service.

Creating inbound integrations

For inbound integrations that involve an external system or application calling into Lombardi to kick off a service, you need to build several Lombardi components and corresponding services. See the sections listed in the following table for more information:

Lombardi component	Description	See...
All items required for an inbound integration	Several components must work together to complete an inbound integration. You can use the procedures in the referenced section to build and test a complete integration.	Building a sample inbound integration
Message events	Use a message event to represent a point in your process where an incoming message is received from an external system.	Modeling message events
Undercover Agent (UCA)	When you include a Message Event in a BPD, you must attach a UCA to the event to call the service that you specify. For example, when a message event is received from an external system, a UCA is needed to invoke the appropriate service in response to the message.	Understanding and using Undercover Agents
Web Service	Lombardi can publish Web services in the same way that it consumes Web services. Using a SOAP connection, external applications can call the Lombardi Web Service to initiate a particular service or set of services.	Publishing Lombardi Web Services

Building a sample inbound integration

To implement an inbound integration in Lombardi, you need to build several components that work together. The following table lists the steps required to build a sample inbound integration. For general and introductory information, see [Integrating with other systems](#).



You can build the required components in the order that works best for you, the steps below simply provide a suggested approach.

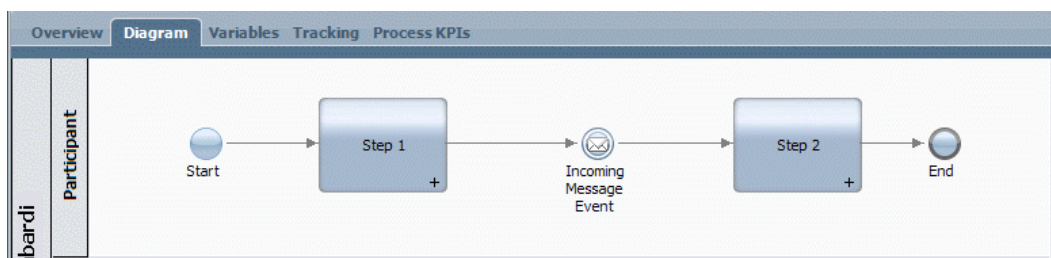
Task	See..
1. Add a message event to a BPD	Adding a message event to a BPD
2. Create a service to pass information and a correlation ID to the BPD (handler service)	Creating a handler service
3. Create an event-based UCA to wrap the service from step 2	Creating a UCA
4. Attach the UCA to the message event from step 1	Attaching the UCA to the message event
5. Create a service with appropriate inputs to call the UCA to send the event (caller service)	Creating a caller service
6. Create an inbound Web Service	Creating an inbound Web Service
7. Add the caller service from step 4 to the inbound Web Service	Creating an inbound Web Service
8. Test the completed inbound integration	Testing the integration

The steps in the referenced sections in the preceding table describe how to create simple components so that you can easily follow along and also easily test your initial implementation. References to more detailed descriptions of the implementation options for each component are provided in the relevant sections.

Adding a message event to a BPD

To build a sample inbound integration, you can start by adding a message event to a simple BPD as described in the following steps.

1. Create a simple BPD that contains two activities like the one shown in the following image. (If you need detailed instructions, see [Creating a BPD.](#))
2. Drag an Intermediate Message Event component from the palette onto the BPD diagram so that it falls between the two activities.
3. Use the Sequence Flow tool to connect the BPD components as shown in the following example.



We'll come back to the message event to set its implementation properties after creating a handler service (for the required UCA) and the UCA.

4. Save your work.

Creating a handler service

The UCA that you attach to the message event needs a handler service to pass the parameter values from the run-time message to the BPD.



For intermediate message events like the one in this sample inbound integration, even if no parameter values are being passed from the message event to the BPD, you must map a UCA input variable to a local variable to ensure that the correct running instance of the BPD resumes execution upon receipt of the message event.

1. Create a General System service and name it `My UCA Handler` or something similar. (If you need detailed instructions, see [Creating a service.](#))
2. Use the Sequence Flow tool to connect the Start Event and End Event components in the service diagram.

Because this service is used to simply pass values, you do not need to add any service components to the diagram.

3. Click the Variables tab.
4. Click the **Add Input** button and replace `Untitled` in the Name field with `someString`.
5. Leave the variable type for the input variable set to `String`.
6. Click the **Add Output** button and replace `Untitled` in the Name field with `someString`.
7. Leave the variable type for the output variable set to `String`.

8. Save your work.

Creating a UCA

After you create the handler service, you can create the UCA to wrap that service. The UCA tells Lombardi what service to run when the message is received. The message can be triggered by Lombardi itself or by an external system as in this example.

1. In the Designer view, click the plus sign next to Implementation and then select **Undercover Agent** from the list.
2. In the New Undercover Agent dialog, enter the following information and then click the **Finish** button.

Name	Type <code>My UCA</code> or something similar for the name.
Schedule Type	Select On Event from the drop-down list.
Attached Service	Click the Select button and choose the handler service, <code>My UCA Handler</code> , that you created in the preceding procedure.

3. In the Details section, the queue for processing this UCA is set to Async Queue by default. This setting is fine for the sample integration. (To learn more about the queue options, see [Understanding and using Undercover Agents](#).)
4. In the Parameter Mapping section, you can see that the UCA is automatically mapped to the `someString` variable from the attached service, `My UCA Handler`.
5. Save your work.

Now that the UCA is available, you can attach it to the message event as described in the following section.

Attaching the UCA to the message event

After you create the UCA, you should go back to the message event in the BPD and attach the UCA as described in the following steps.

1. Open the BPD that includes the message event.
2. Click the message event in the BPD to select it.
3. Click the Implementation option in the properties.
4. In the Message Trigger section, click the **Select** button next to the Attached UCA field and pick `My UCA` that you created in the preceding steps.
5. Leave the Consume Message and Durable Subscription check boxes enabled. (For more information about these options, see [Modeling message events](#).)
6. Notice that the UCA Output Correlation is automatically set to the `someString` variable from the UCA.

In the field next to the variable, type `tw.system.process.instanceId`. This sets the value of the `someString` variable to the ID of the running instance, which enables you to test the implementation in the Inspector as instructed in [Testing the integration](#).

7. Save your work.

Creating a caller service

The next step in completing this sample inbound integration is to create an Integration service to call the UCA to send the event when the inbound Web Service (that you create in the following section) is called.

1. Create an Integration service and name it `Inbound WS Handler` or something similar. (If you need detailed instructions, see [Creating a service.](#))
2. Drag an Invoke UCA component from the palette and place it between the Start Event and End Event in the service diagram.
3. Use the Sequence Flow tool to connect the service components on the diagram.
4. Click the Invoke UCA component in the diagram and then select the Step option in the properties.
5. Type `My UCA` in the Name field.
6. Select the Implementation option in the properties.
7. Click the **Select** button next to the Attached Undercover Agent field and pick the Undercover Agent that you created in the preceding procedure, `My UCA`.
8. Select the Data Mapping option in the properties.
9. Notice that the Input Mapping is automatically set to the `someString` variable from the UCA.

In the field next to the variable, type `tw.local.someString`. This sets the input value of the UCA to the local value of the `someString` variable, which enables you to test the implementation in the Inspector as instructed in [Testing the integration](#).

10. Save your work.

Creating an inbound Web Service

Now you need to provide a way for an external system or application to call into Lombardi. The recommended method for accomplishing this is to create and publish a Web Service so that external applications can initiate a particular Lombardi service or set of services. By invoking a SOAP call, external applications can call the Web Service.

1. Select the plus sign next to the Implementation category and then select **Web Service** from the list.
2. In the New Web Service dialog, type `KickTheBPD` in the Name field and then click the **Finish** button.
3. In the Operations section, click the **Add** button and select the `Inbound WS Handler` Integration service that you created in the preceding procedure from the list.
4. In the Operation Detail section, change `Untitled` in the Operation Name field to `doKick` or something similar.
5. Notice the WSDL URI in the Behavior section. You can use this URI to test the sample integration as instructed in [Testing the integration](#).
6. Save your work.

Testing the integration

After you build and link the input and output of the required components as instructed in the preceding procedures, you can test the completed inbound integration using the Inspector in Lombardi Authoring Environment and a utility such as soapUI.

1. Open the simple BPD that you created per the instructions in [Adding a message event to a BPD](#).
2. Click the Run icon in the upper right corner of the BPD diagram. (If you need detailed instructions for using the Inspector, see [Stepping through a process](#).)
3. When Lombardi prompts you to change to the Inspector interface, click **Yes**.



Click the check box if you want Lombardi Authoring Environment to change interfaces without prompting for approval.

4. In the Process Instances tab, click the received task for Step 1 and then click the Run task icon as shown in the following image:

Instance Name	Snapshot	Process	Status	Due Date	Instance Id	Status	Owner	Subject	Priority	Due Date	Task Id
Do Stuff:4	Tip	Do Stuff	Active	May 14, 2...	4	Received	(ROLE) Al...	Step: Step 1	Normal	Apr 30, 2009 4...	5
Do Stuff:3	Tip	Do Stuff	Completed	May 14, 2...	3						

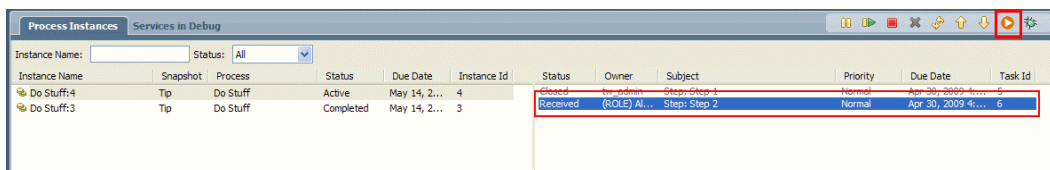
5. The coach for the activity, which is the default Human service, opens in a browser. Click the **Done** button in the Coach to complete the first step.
6. Click the Refresh icon in the toolbar as shown in the following image.

You can see that the process instance is waiting for the incoming message event:

Instance Name	Snapshot	Process	Status	Due Date	Instance Id	Status	Owner	Subject	Priority	Due Date	Task Id
Do Stuff:4	Tip	Do Stuff	Active	May 14, 2...	4	Closed	tw_admin	Step: Step 1	Normal	Apr 30, 2009 4...	5
Do Stuff:3	Tip	Do Stuff	Completed	May 14, 2...	3						

7. Using your SOAP utility, such as soapUI, point to the WSDL URI for the `clickTheBPD` inbound Web Service that you created per the instructions in [Creating an inbound Web Service](#).

8. In your SOAP utility, initiate a request to the `doKick` method. In the `someString` parameter for the method, provide the Instance ID for the currently active instance, which you can see in the Process Instances tab in the Inspector. For example, in the preceding image, the instance ID of the active instance is 4.
9. Click the Refresh icon in the Inspector toolbar. Now you can see that Step 2 has been received, meaning that the message event was successfully processed and the instance is able to move to the next step.
10. Click the Step 2 task to select it and then click the Run task icon as shown in the following image:



11. The coach for the activity, which is the default Human service, opens in a browser. Click the **Done** button in the Coach to complete the second step.
12. Click the Refresh icon in the Inspector toolbar. You can see that the task for Step 2 is closed and the process instance has a status of Complete, indicating that the BPD instance has completed successfully.

Understanding and using Undercover Agents

A UCA is started by an event. Events can either be triggered by a message or on a specific schedule. When a UCA is started, it invokes a Lombardi service in response to the event.

Incoming messages can originate from a Web Service that you create or from a message that you post to the JMS Listener. If you want to create Web Services to initiate inbound requests from external systems, see [Publishing Lombardi Web Services](#).

If you want to post a message to the JMS Listener, the Event Manager has a defined XML message structure that it must receive from an external system. See [Posting a message to Lombardi Event Manager](#) for more information about the required message structure.

The Event Manager is the part of the Process Server that handles event scheduling and queuing as described in *About the Event Manager* in *Lombardi Administration Guide* and the online help for the Process Admin Console. The Event Manager needs a UCA to find the service that it should run. So, if you want to invoke a service when an incoming message event is received or if you want to invoke a service as the result of an event that occurs on a regular schedule, you should create a UCA as described in the following procedures.

Creating an Undercover Agent for an incoming message event

Follow these steps to create a UCA that invokes a particular service as the result of a incoming message event from an external application:



See [Building a sample inbound integration](#) to learn how to build a sample integration that includes this type of UCA.

1. Start Lombardi Authoring Environment and open the appropriate process application in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. In the Designer view, select the plus sign next to Implementation and then select **Undercover Agent** from the list.

3. In the New Undercover Agent dialog, enter the following information and then click the **Finish** button.

Name	Type a name for the new UCA.
Schedule Type	Select On Event from the drop-down list.
Attached Service	Click the Select button to select the service to invoke when the event is received. You can select any type of service except for Human services.

4. In the Common section, you can type a description of the UCA in the Documentation text box.
5. In the Scheduler section, you can see the type of schedule for the current UCA. After you have configured and saved the UCA, you can click the **Run Now** button if you want to test the UCA and monitor it as described in *Maintaining and monitoring Lombardi Event Manager* in *Lombardi Administration Guide* and the online help for the Process Admin Console.
6. Under Details, click the drop-down list next to Queue Name to select the queue that you want from the following options:

Async Queue	Allows Event Manager jobs to run at the same time.
SYNC_QUEUE_1	Forces one job to finish before the next job can start. By default, three synchronous queues are available.
SYNC_QUEUE_2	Forces one job to finish before the next job can start. By default, three synchronous queues are available.
SYNC_QUEUE_3	Forces one job to finish before the next job can start. By default, three synchronous queues are available.



For more information about Event Manager jobs, monitoring those jobs, and creating and maintaining Event Manager execution queues, see *Maintaining and monitoring Lombardi Event Manager* in *Lombardi Administration Guide* or the online help for the Process Admin Console. When you install and run the current process application on a Process Server in a test or production environment, the queue that you select must exist in that environment in order for the UCA to run.

7. Ensure the service shown as the Attached Service is the one that you want to invoke when the external message is received. If not, click the **Select** button to choose a different service.
8. Ensure that the Enabled check box is checked/enabled.



If this check box is not enabled, the Event Manager does not run the UCA when the external message is received. (The Event Manager monitor may show that the Event Manager has executed the UCA, but if this check box is not enabled, the execution does not occur.)

9. In the Parameter Mapping section, click the Use Default check box if you want to use the default value of the input variable in the attached service. If the input variable of the attached service does not have a default value, this check box is disabled.

Type a value in the text box if you want to map a constant value to the input variable of the attached service. For example, you might use a constant for testing purposes.

In most cases, the required values are included in the incoming message event and no action is required here.

10. In the Event section, Lombardi provides a default ID that is unique in the Message Event field. This ID represents the Message Event for Lombardi processing.

If you are posting a message to Lombardi Event Manager from an external system, the ID in this field is the event name that you need to include in the XML message. See [Posting a message to Lombardi Event Manager](#) for more information about the message structure.

If you are using a Web Service to enable an external application to call into Lombardi, you should not alter this ID. Lombardi seamlessly uses this ID if you create an inbound integration as described in [Building a sample inbound integration](#).

11. Open the BPD that includes the message event to which you want to attach the UCA. For example, if you want a particular process to start when a new customer record is created in your SAP system, you can associate the Start Message Event in the BPD with a UCA that handles that incoming event.
12. Click the message event in the BPD to select it.
13. Click the Implementation option in the properties.
14. In the Message Trigger section, click the **Select** button next to the Attached UCA field and pick the UCA created in the previous steps.

Creating an Undercover Agent for a scheduled event

Follow these steps to create a UCA that invokes a service as the result of an event that occurs on a regular schedule:



Scheduled UCAs do not run on the Process Center Server except via the **Run Now** button. So, if you are testing a BPD that includes scheduled UCAs and you want to ensure that the UCAs execute on time, you should run the BPD on a Process Server in one of your runtime environments such as your staging or testing environment.

1. Start Lombardi Authoring Environment and open the appropriate process application in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. In the Designer view, select the plus sign next to Implementation and then select **Undercover Agent** from the list.
3. In the New Undercover Agent dialog, enter the following information and then click **Finish**.

Name	Type a name for the new UCA.
Schedule Type	Select Time Elapsed from the drop-down list.
Attached Service	Click the Select button to select the service to invoke when the event is received. You can select any type of service except for Human services.

4. In the Common section, you can type a description of the UCA in the Documentation text box.
5. In the Scheduler section, you can see the type of schedule for the current UCA. After you have configured and saved the UCA, you can click the **Run Now** button if you want to test the UCA and monitor it as described in *Maintaining and monitoring Lombardi Event Manager* in *Lombardi Administration Guide* or the online help for the Process Admin Console.
6. Under Details, click the drop-down list next to Queue Name to select the queue that you want from the following options:

Async Queue	Allows Event Manager jobs to run at the same time.
SYNC_QUEUE_1	Forces one job to finish before the next job can start. By default, three synchronous queues are available.
SYNC_QUEUE_2	Forces one job to finish before the next job can start. By default, three synchronous queues are available.
SYNC_QUEUE_3	Forces one job to finish before the next job can start. By default, three synchronous queues are available.



For more information about Event Manager jobs, monitoring those jobs, and creating and maintaining Event Manager execution queues, see *Maintaining and monitoring Lombardi Event Manager* in *Lombardi Administration Guide*. When you install and run the current process application on a Process Server in a test or production environment, the queue that you select must exist in that environment in order for the UCA to run.

7. Ensure the service shown as the Attached Service is the one that you want to invoke according to the specified schedule. If not, click the **Select** button to choose a different service.
8. Ensure that the Enabled check box is checked/enabled.



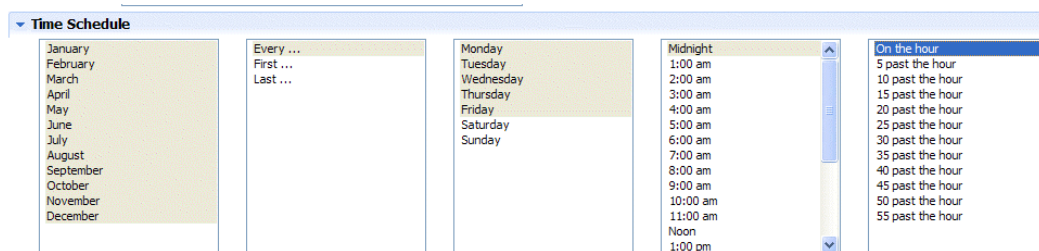
If this check box is not enabled, the UCA will not run.

9. In the Parameter Mapping section, click the Use Default check box if you want to use the default value of the input variable in the attached service. If the input variable of the attached service does not have a default value, this check box is disabled.

Type a value in the text box if you want to manually map a constant value to the input variable of the attached service. For example, you might use a constant for testing purposes.

10. In the Time Schedule section, use the available options to establish a schedule.

For example, if you want to start the attached service every week day at midnight, set the options shown in the following image:



You can select multiple contiguous items by pressing the Shift key, clicking the first in the series, and then clicking the last in the series. To select multiple non-contiguous items, press the Ctrl key each time you click an item.

11. Open the BPD that includes the message event to which you want to attach the UCA. For example, if you want a particular process to start at the same time each day, you can associate the Start Message Event in the BPD with a UCA that establishes the required schedule.

12. Click the message event in the BPD to select it.
13. Click the Implementation option in the properties.
14. In the Message Trigger section, click the **Select** button next to the Attached UCA field and pick the UCA created in the previous steps.

Publishing Lombardi Web Services

You can create and publish a Web Service to enable external applications to initiate a particular Lombardi service or set of services. Using a SOAP integration, external applications can call the Web Service.



See [Building a sample inbound integration](#) to learn how to build a sample integration that includes a Web service.

Follow these steps to create a Web Service that external applications can call:

1. Start Lombardi Authoring Environment and open the appropriate process application in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. In the Designer view, select the plus sign next to the Implementation category and then select **Web Service** from the list.
3. In the New Web Service dialog, type a name for the service and then click the **Finish** button.
4. In the Common section, you can type a description of the Web service in the Documentation text box.
5. In the Operations section, click the **Add** button to choose an existing service to add.

Because the services that Lombardi initiates from a Web service do not have an associated task, do not add services that include the following components: Coach, Modify Task, and Browser Script. See [Understanding service components](#) for more information about these components.

6. Under Operation Details, type a name for the selected service in the Operation Name text box. (Change `Untitled` to the name that you want.)
7. Optionally, you can type a description of the operation in the Documentation text box.
8. Click the **Add** button in the Operation section to continue adding services.
9. In the Behavior section, enable the Protected check box if you want access to the WSDL URI to be password-protected. If password-protected, a user must supply a user name and password to access the operations described in the WSDL URI.

If the Protected check box is not enabled, the `tw_webservice` default user account is invoked. See *Lombardi Administration Guide* for more information about the `tw_webservice` default user.

10. In the Behavior section, click the WSDL URI to view the XML description of the resulting Web service and its elements and operations.

You can supply this URI to the developers who need to call the operations within the Web service.

11. Include a Message Event in your BPD for the incoming request as described in [Modeling events](#).
12. Create an Undercover Agent (UCA) that calls this Web Service as described in [Understanding and using Undercover Agents](#) and then attach the UCA to the event from the preceding step.



When you install a process application on a test or production server, all Web services are included in the list of exposed items in the Process Admin Console as described in [Configuring exposed processes and services](#).

Posting a message to Lombardi Event Manager

If you want to post a message from an external system to Lombardi Event Manager, you must use the message structure described in the following sections. You can use Java Message Service (JMS) to post a message to the Event Manager. See *About the Event Manager* in *Lombardi Administration Guide* to see how the Event Manager processes incoming requests.

Understanding the message structure

The message that you post to the Event Manager must contain an event name (the message event ID generated when you create an Undercover Agent) and it can contain parameter names and values in key-value pairs. ([Creating an Undercover Agent for an incoming message event](#) describes the message event ID that you should use for the event name.) The message must be structured as XML as shown in the following example:

```
<eventmsg>
<!-- The process app acronym and event name are required: The snapshot and UCA
name are optional -->
<event processApp="[acronym]" snapshot="[snapshot_name]"
ucaname="[UCA_name]">[event_name]</event>
<!--Optional: The name of the queue for the event to run in-->
<queue>[queue name]</queue>
<!--Any parameters the UCA may require--
<parameters>
<parameter>
<key>param1</key>
<value>![CDATA[value1]]</value>
</parameter>
</parameters>
</eventmsg>
```

If you do not include the snapshot name in the message, the Event Manager uses the default snapshot on the target Process Server for start message events. For intermediate message events, if you do not include the snapshot name in the message, all active snapshots receive events. To learn more about active and default snapshots, see [Configuring installed snapshots](#).

Note that if the value of the `<value>` element contains XML elements or similar content, you need to wrap the value in a CDATA tag as shown in the preceding example. For information about passing parameter values for each complex variable type, see the following section.

Passing complex variable types to Undercover Agents

You can use Undercover Agents (UCAs) to instantiate services automatically, without human interaction by a Lombardi participant. When you include a Message Event in a Lombardi BPD, you must assign a UCA to it for the Message Event to execute at run time. The Event Manager, which is part of the Lombardi Process Server, handles the scheduling and queuing of UCAs. For more information, see [Understanding and using Undercover Agents](#).

In addition to user-created complex variable types, the following complex variable types can be used to invoke UCAs at run time:

Record	XMLDocument
Date and Time	XMLElement
Boolean	XMLNodeList
Map	ANY (default)

For example, to invoke a UCA using an XML message, let's say your run-time process contains a Message Event that waits for an incoming message. This message contains an input parameter whose value includes the Customer Name, Description, and Age.

First, create the service and then associate the service with an Undercover Agent (the service describes what happens when the UCA is invoked at run time). Then, send the message with the following syntax:

```

<eventmsg>
<event processApp="[acronym]" snapshot="[snapshot_name]"
ucaname="[UCA_name]">[event name]</event>
<parameters>
<parameter>
<key>customerParam</key>
<value>
<Name>John</Name>
<Description>John Description</Description>
<Age>30</Age>
</value>
</parameter>
</parameters>
</eventmsg>

```

The following sections provide examples of how to pass the content of the `<value>` element. The conversion from the event XML format to a complex type is handled automatically by the Lombardi engine.

When the `Any` type is used to pass a parameter value, the actual Lombardi type must be supplied using the `type` attribute of the corresponding element. The `type` attribute can be omitted only when Lombardi knows the exact type or when the type is `String`. The value of the attribute must be an existing Lombardi type—or in case of an array, a Lombardi type concatenated with the `[]` string at the end.

Passing Lombardi Structured types

All structured objects are passed as XML structures, where every object property corresponds to an XML element.

For example:

Variable type: Customer - Name: String (John Doe) - Description: String (Single) - Age: Integer (30)

is mapped to:

```

XML:
<value>
<Name>John Doe</Name>
<Description>Single</Description>
<Age>30</Age>
</value>

```

Keep the following important rules in mind:

- Every object property is mapped to an XML Element with the same name as the property name. The element name is case sensitive. For example, if the property is `Name`, the element name must be `<Name>`, not `<name>`.
- All XML element attributes are reserved. If you pass an attribute (excluding `type`), an error is returned because the passed Document is not valid.
- When an array is passed, it is mapped as a sequence of XML elements with the same name. There are two possibilities:

Passing root level arrays: Every object array item must be placed as content of an `<item>` element. For example:

```
<value>
<item>
<Name>John Doe</Name>
<Description>Married</Description>
<Age>30</Age>
</item>
<item>
<Name>Jane Doe</Name>
<Description>Married</Description>
<Age>31</Age>
</item>
</value>
```

Passing array properties: Every object in the object array is converted to XML using the object property name as an XML Element name. For example:

```
<value>
<Name>John Doe</Name>
<Description>Single</Description>
<Age>30</Age>
<Address>
<Street>10506 Jollyville Rd</Street>
<City>Austin</City>
</Address>
<Address>
<Street>10507 Research Blvd</Street>
<City>Austin</City>
</Address>
</value>
```

- If an object property is `null`, the corresponding element is skipped.

Passing Record type

The Record type is serialized in the same way as Structured types. However, because all values are considered of type `ANY`, the type information must also be passed (using the `type` attribute) in order for the correct objects to be instantiated during de-serialization.

Passing Date/Time types

The format for passing dates is `yyyy/MM/dd HH:mm:ss.S z`.

Example:

- `<value>2004/03/11 14:02:20.0 PST</value>`
- `<value>2000/02/20 11:10:20.0 GMT+6:00</value>`

When the value is converted to the Calendar Java object, it preserves the time zone, and no other modifications (such as adjusting it to the server time zone) is performed.

Passing Boolean type

The valid values for the Boolean type are `true` or `false` (case is not considered).

Example:

```
<value>TRUE</value>
```

Passing Map type

A Map type is passed to a UCA using the following structure:

```
<value>
<entry>
<key> ... </key>
<value> ... </value>
</entry>
</value>
For example:
<value>
<entry>
<key>TX</key>
<value>Texas</value>
</entry>
<entry>
<key>CA</key>
<value>California</value>
</entry>
</value>
```

Because all values and keys in this case need to be of the `ANY` type, the type information must also be passed in order for the correct objects to be instantiated during deserialization. If the object is of the `String` type, the type does not need to be specified.

Passing XMLDocument type

An XML Document is passed as an XML escaped string.

Example:

```
<value>
<![CDATA [
<?xml version="1.0"?>
<Customer>
<Name>John Doe</Name>
<Description>Married</Description>
<Age>30</Age>
```

```

</Customer>
]]>
</value>

```

Passing XMLElement type

An XML Element is passed as an XML escaped string.

Example:

```

<value>
<![CDATA[
<Customer>
<Name>John Doe</Name>
<Description>Married</Description>
<Age>30</Age>
</Customer>
]]>
</value>

```

Passing XMLNodeList type

Every node is passed as an XML escaped string. The array of the nodes is encoded as a sequence of `<item>` elements.

Example:

```

<value>
<item>
<![CDATA[
<Customer>
<Name>John Doe</Name>
<Description>Married</Description>
<Age>30</Age>
</Customer>
]]>
</item>|
<item>
<![CDATA[
<Customer>
<Name>Jane Doe</Name>
<Description>Married</Description>
<Age>31</Age>
</Customer>
]]>
</item>
</value>

```

Passing ANY type

When the type of an input parameter to a UCA is declared as `ANY`, the information about the actual type must be passed as part of the XML.









Example:






Define a process with one input parameter, `Name`, of type `ANY`. When the data is encoded in XML, the actual type must be supplied as the value for the `type` attribute. If the type is not passed, the `String` type is assumed.

```
<value type="String">
John Doe
</value>
```

Modeling events

Lombardi enables you to model events that can occur at the beginning, during, or at the end of a run-time process (as opposed to activities that are performed by participants within the process). You can include the following events in your Lombardi Business Process Definitions (BPDs):

Event	When to use...	For more information, see...
 Start Event	<p>Use to model the start of a process if you want to enable process participants to start a process manually from Lombardi Process Portal. Also use to model the start of nested processes. Use the Start Message Event if you want an incoming message or event to kick off a process.</p>  <p>A Start Event is automatically included each time you create a BPD. A BPD can include one Start Event and multiple Start Message Events if you need to be able to start the process more than one way.</p>	Creating a BPD
 Timer Event	<p>Use to model escalation paths or delays in your BPDs. Using a timer event, you can specify a time interval after or before which some activity is performed.</p>	Modeling timer events
 Intermediate Tracking Event	<p>Use to indicate a point in a process at which you want Lombardi to capture the run-time data for reporting purposes.</p>	Creating a basic custom report and Creating a more advanced custom report
 End Event	<p>Use to end process execution.</p>  <p>An end event is automatically included each time you create a BPD.</p>	Creating a BPD
 Start Message Event	<p>Use to model the start of a process if you want an incoming message event to start a new instance of the process. The start is triggered by the completion of the Undercover Agent (UCA) that you select.</p>  <p>A BPD can include more than one Start Message Event.</p>	Modeling message events

Event	When to use...	For more information, see...
 Start Ad-hoc Event	Use when you need to include ad-hoc actions that can be executed at any time during process execution. For example, you can include an ad-hoc event to enable end users to cancel a customer order at any time during the ordering process.	Using ad-hoc events
 Intermediate Message Event	Use to model a message event received while a process is running. The event is triggered by the Undercover Agent (UCA) that you select.	Modeling message events
 Terminate Event	Use to close all running tasks associated with the process and to cancel all outstanding timers. The process shows a status of Terminated in the Inspector.	Modeling timer events and Modeling message events
 Intermediate Exception Event	Use to catch process execution exceptions and handle exceptions with an error handler activity or further process flow.	Handling exceptions
 End Exception Event	Use to throw an exception to parent processes.	Handling exceptions



Intermediate events (Timer Events, Intermediate Message Events, and Intermediate Exception events) can be attached to activities within your BPDs or they can be included in the process flow, connected with sequence lines. Other events are simply included in the process flow.

Modeling timer events

You can use timer events to specify a time interval after or before which some activity is performed, or another path in your process is taken. You can attach a timer event directly to an activity in your BPD or connect a timer event to other process steps using sequence lines. The following table describes the options:

Timer event type	Description	Use to model...
Attached timer event	Attached directly to an activity in a BPD	Escalation paths as described in Using attached timer events
Intermediate timer event	Connected to other process steps using sequence lines	Delay timers as described in Using intermediate timer events

Using attached timer events

When a running process instance reaches an activity with an attached timer event, a timer is started. The time interval for the timer is calculated according to the configuration that you specify in the implementation properties for the timer event. When the specified time interval has elapsed, the process follows the sequence line that flows from the attached timer event to a subsequent activity.

The following example shows how to model an escalation path using an attached timer event. In this example, if the activity takes longer to complete than the defined period of time, the timer event is triggered and the process follows the path from the attached timer to the escalation activity.

For the following example, you can use the HR Open New Position BPD included in the Quick Start Tutorial process application. (If you do not see the Quick Start Tutorial process application in your list of applications in the Process Center Console, ask your Lombardi administrator to give you access.) To do so, clone the

Quick Start Tutorial process application so that your edits do not affect other users of Lombardi Authoring Environment. When a BPD includes review and approval activities, you may want to include escalation paths like the one in the following example to help ensure timely completion of the overall process.

1. Open the BPD in the Designer and click the Diagram tab.

2. Drag a Timer Event component from the palette to the Approve/reject position activity.

The event is anchored to the activity. To verify this, select the activity. If the activity's outline includes the event, the event is properly attached.

3. Select the Timer Event component in the BPD diagram and then click the Implementation option in the properties.

4. In the Attached Event Details section, disable the Close Attached Activity check box.

This check box closes the attached activity when the timer event is triggered, which is not the required behavior in this example. In this example, the end user should complete the activity when he receives the escalation notice.

5. From the Trigger On drop-down list, select **After Due Date**.

This selection causes the event to trigger when the due date for the activity has elapsed. The due date is calculated according to the work schedule for the BPD and the priority settings for the activity. For more information, see [Setting the work schedule for a BPD](#).



If you choose to trigger before or after a custom date, you can enter the JavaScript to determine the custom date in the Custom Date text box. Your script must return a Date object, specifying when the timer is to be executed.

6. In the Before/After Difference text box, type 1 and then select **Days** from the associated drop-down list.

This causes the example event to trigger one day after the activity's due date.



To use a variable to specify this value, click the variable icon next to the text box and select the variable that you want.

7. Leave the value in the Tolerance Level text box at 0.

The tolerance level enables you to delay the timer event for a specified amount of time. For example, you could specify a tolerance level of one hour if you wanted the escalation to occur one day and one hour after the activity's due date.

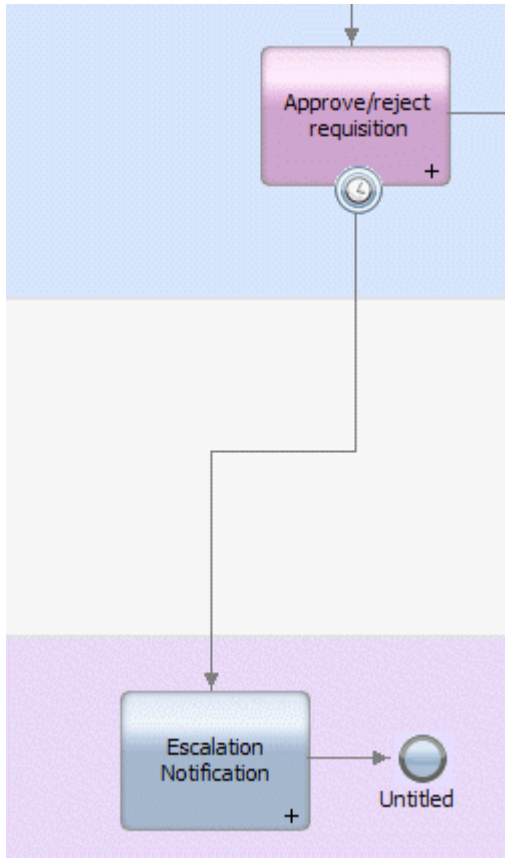


To use a variable to specify this value, click the variable icon next to the text box and select the variable that you want.

8. Drag an activity component from the palette into the System lane.

9. While the activity is still selected, in the Step tab in the properties, enter the name: `Escalation Notification`.

10. Drag an End Event component from the palette into the System lane and position it directly after the activity created in the preceding steps.
11. From the palette, click to select the Sequence Flow tool and then add a sequence line from the attached timer event to the new Escalation Notification activity, and from that activity to the End Event component as shown in the following image:

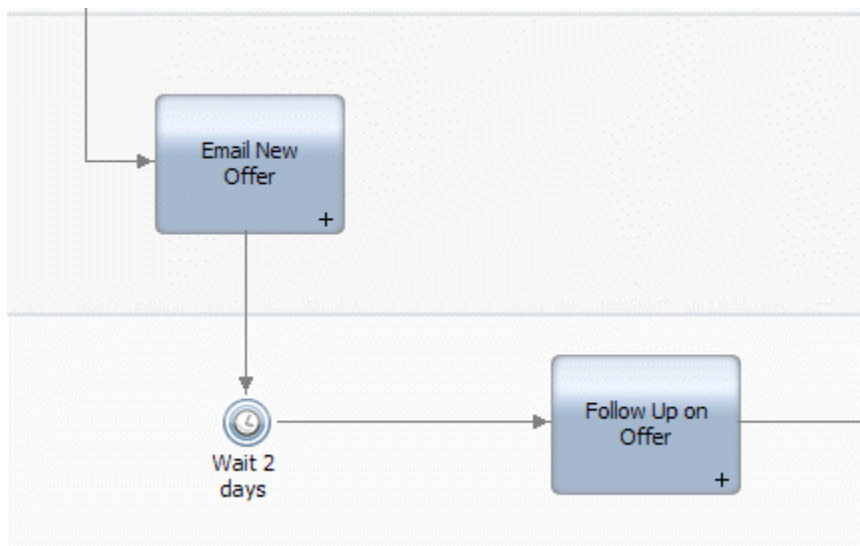


12. You can complete the escalation path by building an underlying service to implement the Escalation Notification activity. You can use the Send Alert component for the underlying service.
13. You can attach more than one timer event to an activity. Using this example, you can attach another timer event to the activity that triggers two days after the due date has passed, thereby providing multiple notifications should the end user fail to perform the task after the first notification is received.

Using intermediate timer events

If a timer event is not attached to an activity, it causes a delay. The process waits for the timer to trigger before proceeding to the next activity. For example, if your BPD includes an activity that emails offers to customers and you want your internal sales team to follow up two days after the offers are mailed, you can place a timer between the two activities as shown in the following image.

The following image shows a sample delay timer event. In the sample, the delay ensures that a specific amount of time passes between the completion of one activity and the start of another.





You can set the properties for the intermediate timer event in the same way as outlined for attached timer events in the preceding section. The only difference is that there are no Attached Event Details options.

Modeling message events

Use a message event to represent a point in your process where an incoming message is received. Incoming messages can originate from a Web Service that you create, a message that you post to the JMS Listener, or simply by calling a UCA in a Lombardi service. If you want to create Web Services to initiate inbound requests from external systems, see [Publishing Lombardi Web Services](#).

If you want to post a message to the JMS Listener, the Event Manager has a defined XML message structure that it must receive from an external system. See [Posting a message to Lombardi Event Manager](#) for more information about the required message structure.

You can include the following types of message events in your BPDs:

Event	When to use...	For more information, see...
 Start Message Event	Use to model the start of a process if you want an incoming message event to kick off the process. A BPD can include more than one Start Message Event.	Using start message events
 Intermediate Message Event	Use to model a message event received during execution of the activities in a process. Intermediate Message Events can be attached to activities within your BPDs or they can be included in the process flow, connected with sequence lines.	Using intermediate message events

Before including any type of message event in a BPD, you should be aware of the following:

- Messages can be received by any running process that contains one or more message event.
- By default, when a message is delivered to a running process, the message is consumed by the first message event in the BPD that can accept it (as determined by the UCA that is attached to the message event). When a message is consumed, it will not be processed again by that message event, or any other message event in the BPD instance that can accept it, should the execution of the BPD instance

loop back and reach the same message event(s). If a new instance of the message is delivered to the process instance, this message is available for consumption again and is accepted by the message event.

- Message events can be used to enable roll-forward scenarios in which the same message needs to be passed through multiple steps until it reaches the appropriate step in the process where it is to be consumed. To enable rolling a message forward through multiple message events, enable the Consume Message option only for the last message event in the chain of roll-forward message events. You can also use conditions to further control message consumption.
- Occasionally, you may need to set conditions on the processing of incoming messages. If the condition that you specify evaluates to true, the message is accepted and processing continues—otherwise, it is stopped. Because the message condition is evaluated before the message values can be passed to the input variables of the process definition, the message values are passed to the condition in a special namespace, `tw.message`. If the message condition evaluates to true, the values are passed from the `tw.message` namespace to the BPD input variables.

Using start message events

If you want a process to start when a message is received, use a Start Message Event in your BPD. For example, you may want an employee on-boarding process to start when a record for each new employee is created in your SAP HR system. When the record is created, the SAP systems sends an event to Lombardi. Lombardi captures the event and starts the follow-on steps for each new employee such as setting up the necessary space and computer equipment, requesting and creating a security badge, and so on. The following example describes how to model the Start Message Event in a BPD for this type of process.

When including start message events in a BPD, you should be aware of the following:

- The general information that applies to all types of message events covered in [Modeling message events](#).
- When a message is received by a start message event (specifying that an incoming message is to start a process at run time), a new instance of the BPD is created and a unique BPD instance ID is assigned to it.
- If you use multiple start message events in a single BPD, use a separate UCA for each. If you use the same UCA for multiple start message events, Lombardi instantiates multiple instances of the BPD.

1. Open the BPD in the Designer and click the Diagram tab.
2. Drag a Start Message Event component from the palette onto the diagram.
3. Click the Implementation option in the properties.

4. In the Message Trigger section, click the **Select** button next to the Attached UCA field to select a preexisting Undercover Agent.

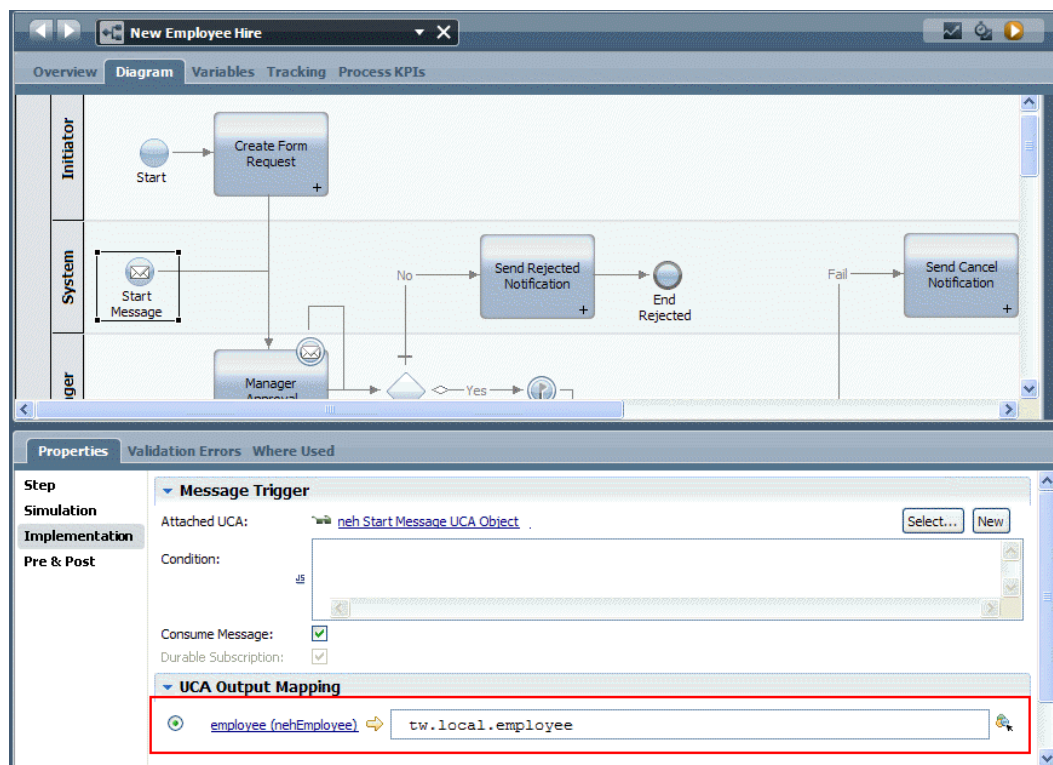
To create a new UCA, click the **New** button and see [Understanding and using Undercover Agents](#) for more information.

5. In the Condition text box, type a JavaScript expression if you want to define conditions under which the message event is processed.

If you do specify a condition and the condition evaluates to true, the message is accepted and processing continues. If the condition evaluates to false, processing stops. In most cases, special message conditions are not necessary.

6. The **Consume Message** check box is selected by default. Clear the Consume Message check box if you do not want the incoming message to be consumed after it has been received by the message event. Refer to the bulleted list in [Modeling message events](#) to learn more about message consumption.
7. The **Durable Subscription** check box is not available for start message events, only for intermediate message events as described in the following section.
8. In the UCA Output Mapping section, map one or more of the listed UCA output variables to appropriate input variables when you want their run-time values passed to the BPD instance.

For example, if the Start Message Event starts an instance of an on-boarding process when an employee record is created in your SAP HR system, you can map the employee information from the UCA to a local variable in the BPD as shown in the following image:



Using intermediate message events

Include an intermediate message event in your BPD when you want to model a message event received during execution of the steps in a process. Intermediate message events can be attached to activities within your BPDs or they can be included in the process flow, connected with sequence lines.



To build a sample inbound integration that includes an intermediate message event, see [Building a sample inbound integration](#).

When including intermediate message events in a BPD, you should be aware of the following:

- The general information that applies to all types of message events covered in [Modeling message events](#).

- By default, intermediate message events have a durable subscription. When a message arrives before a process has executed to a point where the event can accept the message, the durable subscription causes the message to be stored until the message event is reached. Only the most recently received message is stored.
- When the process execution reaches an intermediate message event, further execution along that path is blocked until an incoming message arrives.
- When you use an attached or intermediate message event, you must map an appropriate UCA output variable to a local variable in the BPD to ensure that the parameter values of the run-time message are passed to the correct BPD instance.

1. Drag an Intermediate Message Event component from the palette onto the BPD diagram so that it is attached to an activity.

The event is anchored to the activity. To verify this, select the activity. If the activity's outline includes the event, the event is attached.



For a sample of an intermediate message event that is included in the process flow, connected with sequence lines, see [Building a sample inbound integration](#).

2. In the Attached Event Details section, the **Close Attached Activity** check box is enabled by default. This setting closes the attached activity when the message event is triggered. You want this behavior in cases where the receipt of the message event signals completion of the activity. Otherwise, clear this check box.
3. In the Message Trigger section, click the **Select** button next to the Attached UCA field to select a preexisting Undercover Agent.

To create a new UCA, click the **New** button and see [Understanding and using Undercover Agents](#) for more information.

4. In the Condition text box, type a JavaScript expression if you want to define conditions under which the message event is processed.

If you do specify a condition and the condition evaluates to true, the message is accepted and processing continues. If the condition evaluates to false, processing stops. In most cases, special message conditions are not necessary because you should implement each message event with a separate UCA.

5. The **Consume Message** check box is selected by default. Clear the Consume Message check box if you do not want the incoming message to be consumed after it has been received by the message event. Refer to the bulleted list in [Modeling message events](#) to learn more about message consumption.
6. The **Durable Subscription** check box is selected by default for intermediate message events.

This setting enables the message event to receive an incoming message, even when the message event is not in an active state. (When a token is on a step, that step is in an active state. See [Understanding tokens](#) for more information.)

7. In the UCA Output Correlation section, you must map an appropriate UCA output variable to a local variable in the BPD to correlate the message event with the BPD instance (Lombardi only requires one variable mapping to correlate the event).

Using ad-hoc events

If you want to enable end users to start an ad-hoc process during the execution of another process, you can do so by using a Start Ad-Hoc Event component in your BPD. For example, you may want to enable end users to cancel an order, determine the status of an order, or perform some other ad-hoc function during the normal processing of an order. Because an ad-hoc process is executed in the context of the regular process instance, it has access to all the data of the regular process instance and can also manipulate the flow of the regular process instance, depending on the logic that you include.



To understand how end users access available ad-hoc processes, see the relevant section in *Lombardi Process Portal User Guide* or online help.

Building a sample ad-hoc process

The following example shows how to model an ad-hoc event that enables end users to view the content of a request for a new employee position at any time during normal processing of the requisition.

For the following example, you can use the HR Open New Position BPD included in the Quick Start Tutorial process application. (If you do not see the Quick Start Tutorial process application in your list of applications in the Process Center Console, ask your Lombardi administrator to give you access.) To do so, clone a snapshot of the Quick Start Tutorial process application so that your changes do not affect other users of Lombardi Authoring Environment.

1. Open the BPD in the Designer and click the Diagram tab.
2. Drag a Lane component from the palette to the diagram.
3. Right-click the new lane and select **Move Lane Down** until the new lane is the last lane in the BPD (below the System lane).
4. Click on the new lane in the diagram (named `Untitled_1` by default) and in the **Name** field in the properties, type `Ad-hoc process`.
5. Notice in the Common section of the properties that the default lane assignment is **All Users**. To assign to one of the tutorial participant groups, click the **Select** button.

For example, if only HR Managers should be able to view information about pending requisitions, select HR Managers from the list of available participant groups.

6. Drag a Start Ad-Hoc Event component from the palette onto the BPD diagram so that it is positioned in the new `Ad-hoc process` lane.
7. In the Step tab in the properties, type `Show Requisition Data` for the event name.
8. Drag an activity from the palette into the Ad-hoc process lane.
9. In the Step tab in the properties, type `Show Data` for the activity name.
10. Drag an End Event component from the palette onto the BPD diagram so that it is positioned after the `Show Data` activity in the `Ad-hoc process` lane.
11. Using the Sequence Flow tool, connect the Start Ad-Hoc Event, Show Data activity, and End Event on the BPD diagram.
12. Right-click the Show Data activity and select **Activity Wizard** from the list of options.

13. In the Activity Wizard - Setup Activity dialog, make the following selections:

Service Type	Human Service (Service with human interaction)
Service Selection	Create a New Service

In the **New Service Name** field, type `Show Data` for the new service. (For this example, name the new Human service the same as the corresponding activity in the BPD.)

14. In the Activity Wizard - Setup Activity dialog, click the **Next** button.
15. In the Activity Wizard - Parameters dialog, choose the process variables from the regular process to use as input and output for the new service for the ad-hoc process.

For the private variable named `requisition`, leave the **Input** field set to **true** and change the **Output** field to **false**. These settings reflect the fact that our sample ad-hoc process simply displays the requisition data and does not pass back modified data. For other variables, click to change the setting from true to false under the Input and Output field. Click the **Finish** button.

The new service is created and attached to the activity. The new service includes a single Coach.

16. Double-click the Show Data activity in the Ad-hoc process lane in the BPD.

The new service opens in the Designer and you can see the diagram.

17. Click the **Coaches** tab and then click the listed Coach to see its controls.

Because we used the Activity Wizard, the Coach includes a form element for each of the parameters in the `requisition` variable.

18. Save your work and then follow the instructions in [Running a sample ad-hoc process](#).

Running a sample ad-hoc process

Before you begin to test the sample ad-hoc process, open Lombardi Process Portal in a Web browser as described in *Lombardi Process Portal User Guide*. And, ensure that you log in to Lombardi Process Portal as a member of the participant groups who receive and can complete the tasks generated by the activities in the sample BPD.

1. In the Designer in Lombardi Authoring Environment, open the BPD to which you added an ad-hoc event as described in [Building a sample ad-hoc process](#).
2. Click the Run icon in the upper right corner of the BPD diagram.
3. Lombardi Authoring Environment switches to the Inspector where you should see a new `Submit requisition` task.
4. Go to Lombardi Process Portal and click the Run icon for the new `Submit requisition` task in your inbox.
5. Fill out the Job Requisition information, click the **Next** button, and then click the **Submit** button on the Confirm Job Position form.
6. When the next task for the process instance (`Approve/reject requisition`) displays in your inbox in Lombardi Process Portal, click the instance name or task subject to open the details page.



If the task does not display, reload the browser page.

- Click the Actions menu in the toolbar, and select the name of the ad-hoc process. (The name of the process is the name that you assign the ad-hoc event that kicks off the process in the BPD diagram in Lombardi Authoring Environment. For this sample, the name is `Show Requisition Data`.)

This causes Lombardi to generate a `Show Data` task in the Tasks area of the details page.

- Click the Run icon for the Show Data task. Lombardi displays the data that you entered in step 5.
- Click the **OK** button.

Now you can continue with normal processing by completing the next task in the process instance, `Approve/reject requisition`. You can invoke the ad-hoc process again, after completion of the `Approve/reject requisition` task, to see whether the requisition has been approved.

The ad-hoc event that you added to the BPD diagram enables you to view the requisition information at any time during execution of the regular process.

Managing and mapping variables

In Lombardi, variables capture the business data that is passed from step to step in a process. When you develop Business Process Definition (BPDs) in Lombardi, you should spend time creating data models for those BPDs during the design phase. To create a data model, compile a list of all data that a process requires, determine how to best represent that data in Lombardi, and develop an understanding of how to effectively pass that data from one step to the next in your process. Understanding how variables work, the available variable types, and other topics covered in this section will enable you to develop a comprehensive and effective data model for your Lombardi process.

You can think of variables in Lombardi as names that point to values in memory. For example, a process you are designing may need a customer account number to properly execute. If so, you can declare a variable in your process called `accountNumber`. When the `accountNumber` variable is called during process execution, it points to the value of the account number in memory. For example, if an end user provides 50891 as the value for `accountNumber` in a Coach form, the rest of the steps in the process have access to that value.

Variable types in Lombardi

During Lombardi installation, the System Data toolkit is imported into the Process Center repository so that each process application and toolkit that you create has access to Lombardi system data. The System Data toolkit includes assets that all Lombardi projects require, including variable types. The Lombardi System Data toolkit provides the following types of variables:

Type	Description	Includes types such as...
Base types	You can create a custom variable type using a base type	String, Integer, and Decimal
System types	You cannot change or customize	ANY, Record, and XMLDocument

Custom types are variable types that you create from a base type, and then further constrain based on pattern, length, or value. For more information, see [Creating custom variable types](#).

In Lombardi, every variable name is associated with one type of variable. It is the variable type that determines what values are legal for the associated variable. If you create a variable called `myInteger`, for example, and you associate it with the `Integer` variable type, `myInteger` cannot legally store alphabetic characters. When Lombardi encounters a value that is illegal for a type, it attempts to convert the value to a correct type using variable conversion rules before throwing an exception. If you do not know the type of the variable during development when you are declaring variables, you should use the `ANY` type.

The following table provides more information about the base variable types provided in Lombardi System Data toolkit:

Base variable type	Description
String	Allows alpha-numeric characters to be entered into the variable.
Integer	Accepts digits without a decimal place, such as 45 or 20.
Decimal	Allows you to enter numbers with a decimal place, such as 45.3 or 20.139.
Date	Allows date and time formats to be entered into the variable.
Time	Allows date formats to be entered into the variable as times. The user enters a time, and before the variable is entered into the symbol table, it is converted to and behaves like a date.
Selection	Allow you to provide a list of possible entries to an end user, of which the user can select only one. A selection is a list of different values; each value is typed as a string. A selection type appears at run time on a Coach form as a drop-down list or as radio buttons.
Boolean	Accepts either <code>true</code> or <code>false</code> as values. It appears at run time on a Coach form as a check box.
Structure	To use a Structure type, you need to create a custom Structure type because a usable structure must have defined properties. A structure is simply a way of grouping business data that is related to the same subject. For example, a <code>Customer</code> structure might contain elements such as <code>lastName</code> , <code>firstName</code> , <code>homeNumber</code> , <code>streetAddress</code> , and so forth.

For the system variable types provided in Lombardi System Data toolkit, you can open the variable type in the Designer in Lombardi Authoring Environment to learn when and how to use the type. For example, to open the `Record` variable type included in the System Data toolkit, follow these steps:

1. Open a process application in the Designer in Lombardi Authoring Environment.
2. Click the indicator next to the Toolkits category to see a list of toolkit dependencies for the current process application.
3. Click the indicator next to the System Data toolkit to see its contents.
4. Click the **Data** category and then double-click the **Record** variable type to open it.
5. The Documentation field provides information about the variable type.

The description informs you that a record is a group of `ANY` types and that you do not need to declare the number of `ANY` typed variables that you want to go into the `Record`. So, the `Record` type is similar to a `Structure` type, except you do not need to declare the type or the number of variables it contains.

To learn about other system variable types included in Lombardi System Data toolkit, open the type as described in the preceding steps, read the documentation provided, and examine the parameters and parameter properties, when applicable.

Variable scope in Lombardi

In Lombardi, the variables that you declare for a Business Process Definition (BPD) or service are local variables. Local variables are only accessible to the currently executing process instance or service. When a running process instance or service reaches an exit point, the variable values are no longer available in memory. Because of the local nature of process variables, you must explicitly define the variables for each BPD and service that implements your overall process, including nested BPDs and services. Because variables are unique to an individual BPD or service, you can use a variable of the same name in a nested BPD or service and there are no conflicts at run time.

All Lombardi variables are JavaScript objects. Lombardi uses namespaces to organize these objects and their functions and methods. The following table describes the namespaces most commonly used during process design and development in Lombardi:

Namespace	Description
tw	Top-level Lombardi namespace
tw.object	Access Lombardi JavaScript objects and variable types
tw.local	Access and update BPD and service-level variables
tw.system	Access system features and functionality
tw.system.org	Access security functionality
tw.epv	Access exposed process values (EPVs)
tw.env	Access environment variables



To learn more about the JavaScript API, see the reference information posted to the Lombardi Documentation Wiki: <http://wiki.lombardi.com/display/tw7/Teamworks+7+JavaScript+API>. (You must have an account for the IBM Customer Support for Lombardi site to access the wiki.)

The namespaces described in the preceding table are available to you as you develop processes in the Designer. This enables you to:

- Easily assign a system ID to a local variable as described in [Setting variables in pre and post assignments](#)
- Use methods as described in [Using JavaScript methods on variables](#)
- Reference an EPV in a decision gateway and any JavaScript code in your BPD as described at the end of the procedure in [Creating exposed process values \(EPVs\)](#)
- Use environment variables in a script as described at the end of the procedure in [Setting environment variables](#)

Creating custom variable types

In Lombardi, you can create a custom variable type by using a base variable type or by defining a new complex structure. For more information, see [Variable types in Lombardi](#).



When you create a new custom variable type in a process application, that variable type is available for all BPD and services included in the process application. If you want to share a custom variable type across process applications, create or store the custom type in a toolkit and then create a dependency on that toolkit from the process applications that require the variable. For more information, see [Creating a toolkit dependency in the Designer view](#).

The following procedure describes how to create a custom variable type. To complete the following steps, you must have write access to a process application or toolkit in the Process Center repository. Access to process applications and toolkits is controlled by users who have administrative rights to the repository. For more information, see [Managing access to the Process Center repository](#).

1. Start Lombardi Authoring Environment and open the appropriate process application or toolkit in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. Click the plus sign next to the **Data** category and select **Variable type** from the list.
3. In the New Variable type dialog enter a name for the variable type and click the **Finish** button.

Name variable types so that each word is capitalized like so: MyType. This naming convention enables you to easily distinguish between the variable types that you create and the variables that you declare. Names of variable types are case sensitive.

4. In the Behavior section, choose a Definition Type from the drop-down list:

Simple Type	Create a new variable type using an existing base type such as String, Decimal, and Integer and further constrain based on pattern, length, or value as described in the following steps.
Complex Structure Type	Create a new complex type by specifying the parameters for the structure and the type of each parameter. (See Adding process variables to a BPD for the steps involved in creating a complex structure type. See Initializing complex variables and lists to learn how to perform necessary initialization for complex structures.)

5. In the Simple Type section, choose the base type that you want.

The following table describes the options as well as how you can further constrain allowed values:

String	Constrain by its length or to a pattern using a regular expression.
Integer	Constrain by a minimum or maximum value, by precision, or by a regular expression.
Decimal	Constrain by a minimum or maximum value, by precision and scale, or by a regular expression.
Date	Constrain the format of the date for users. However, the date type is not stored in the format so dates can be passed easily from process to process.
Time	Constrain the format of the date for users. (The user enters a time, and before the variable is entered into the symbol table, it is converted to and behaves like a date.)
Selection	Provide the value and display text for each possible entry.

6. In the Validation section, establish the validation rules to further constrain allowed values.

For example, if you choose the Date base type, select the Date Validation that you want from the drop-down list next to the Format field.

If you choose `yyyy.MM.dd`, this variable only accepts dates in the format `2009.08.14` (for August 14, 2009).

7. In the Error Message text box (in the Simple Type section), type the text that you want to display when a date is not provided in the chosen format.

For example, a good error message would be: `Dates must be in yyyy.MM.dd format`

- Expand the Advanced Properties section and set the XML serialization options if you want to save the custom variable type to an XML file.

For complex structure types, you can also set XML serialization options for each parameter.

- Click **Save** in the main toolbar.

Now you can select the new custom variable type for the variables that you create as described in [Declaring and passing variables](#).

Declaring and passing variables

As described in [Variable scope in Lombardi](#), variables for a Business Process Definition (BPD) or service are local variables and as such are only available to the currently executing process instance or service. Because of this fact, BPD-level variables are not affected by the services and sub-processes that implement the activities in a BPD unless you properly declare and then pass the required variables. In general, to properly declare variables and manipulate variable values throughout the steps of an entire BPD you need to:

- Declare variables at the BPD level
- Pass those variables as inputs to the sub-processes and services that require them for their implementation
- Pass the variables from sub-process and services back up to the main BPD as outputs when you want the main BPD to be aware of changes made to the variables in the sub-process or services

When passing variables as inputs and outputs, you should:

- For each sub-process and service, declare variables using the same name (or a similar one) and variable type as the variables that are required from the main BPD.



Because variables are unique to an individual BPD or service, you can use a variable of the same name in a nested BPD or service and there are no conflicts at run time.

This enables the sub-process or service to display and manipulate the values that it receives from or pushes out to the main BPD.

- Use the Data Mapping tab to set the input and output mapping for each activity in a BPD diagram.

This ensures that the values received and generated by the sub-processes and services that implement the activities map to the variables from the main BPD.



When you use the Activity Wizard to create a service to implement an activity, you can pick the variables from the main BPD to use as input and output. When you do, the Activity Wizard automatically declares the variables for the resulting service and completes the data mapping for the activity. You can use the Activity Wizard to create Human, Rule, and General System services.

The following sections provide instructions for defining basic variables for a BPD and then passing those variables to a service that implements an activity in the BPD:

- [Declaring variables for a BPD](#)
- [Declaring variables for a service](#)

- [Mapping input and output data for an activity](#)
- [Testing declared variables and data mapping](#)

In addition to the considerations covered by the sections in the preceding list, you also need to be aware of how Lombardi passes a variable as described in [Passing variables by value or by reference](#).

Passing variables by value or by reference

Whether Lombardi passes a variable by value or by reference is determined by several factors as described in the following table:

From	To	Pass by
BPD activity	Nested BPD	Value, if simple variable type
BPD activity	Nested BPD	Reference, if complex variable type
BPD activity	Service	Value
Service	Nested service	Value, if simple variable type
Service	Nested service	Reference, if complex variable type

Also, when passing variables, you should be aware of the following:

- If a variable is passed by value, the BPD or service that receives the value can manipulate it and the change does not affect the original value unless the receiving BPD or service returns the variable as an output.
- If a variable is passed by reference, the changes made by the BPD or service that receives the reference do affect the original value, even if the receiving BPD or service does *not* return the variable as an output.

Because of the way Lombardi handles variables, you should follow these guidelines:

- If the variable is a simple type, declare the variable as an input and an output in nested BPDs, services, and nested services.
- If the variable is a complex type, you must declare the variable as an input. And, although the output declaration is not required (because complex types are passed by reference), it is a good idea to also declare the variable as an output. This ensures that other developers will be aware that the nested BPD, service, or nested service returns a complex variable.
- Always use an identical name and data type for a set of input and output variables for data that is passed in, processed, and then passed back.

Declaring variables for a BPD

For each BPD that you create, you need to declare variables to capture the business data that is passed from step to step in your process. You can add the following variables to your BPDs:

Variable	Description
Private	Private variables are local variables that are only used within the process.
Input	Input variables are mapped to values that you can pass into the current process.
Output	Output variables are mapped to values that you can pass out from the current process to a parent process.



Exposed process values (EPVs) are a special type of variable that you can create to enable end users to set or alter values while instances of a process are running. EPVs allow end users to adjust specific variable values as constants, thereby affecting the flow of running process instances, task assignments, and so on. If EPVs have been created, you can link them to multiple processes and services from the Variables tab in the Designer. For more information see [Creating exposed process values \(EPVs\)](#).

The following steps describe how to declare private variables for a BPD. The variables are basic so that you can easily see how to pass them to a service that implements an activity in the BPD.

1. In Lombardi Authoring Environment, create a BPD as described in [Creating a BPD](#).
2. Click the Variables tab in the Designer.
3. Click the **Add Private** button.
4. In the details for the variable, enter `employeeId` in the Name field.



All variable names should start with a lowercase letter, with subsequent words capitalized like so: `myVar`. Do not use underscores or spaces in variable names. Variable names are case sensitive.

5. In the Documentation text box, you can type a description of the variable.
6. By default, the **Is List** check box is disabled. For the `employeeId` variable, leave the check box disabled. You should enable this check box only when a variable can accept list values (arrays).
7. Click the **Select** button next to Variable Type and select the `Integer` type from the list.

Variable types from the System Data Toolkit are listed as well as any custom variable types that you have created or to which you have access via a toolkit. See [Creating custom variable types](#) for more information.

8. Leave the **Has Default** check box disabled for the `employeeId` variable.

In some cases, you should enable this check box and provide a default value. For example, if your process includes logic where a null value for the variable in question is not a valid input at run time. You might also want to establish a default temporarily so that you can test your process and the data flow.

9. Leave the **Available in Search** check box disabled for the `employeeId` variable.

Enable this option when you want to include a variable in the business data that end users can view in Lombardi Process Portal. Provide an alias for the data represented by this variable in the Search Alias text box. The alias is the name that is visible to Process Portal users.



If a variable is used in parent and nested processes, use the same search alias if you want search results to include all related processes.

10. Leave the **Performance Tracking** check box disabled for the `employeeId` variable.

Enable this option to include the values for this variable in the data that is collected and used for Lombardi reports. For more information, see [Tracking Lombardi performance data](#).

11. Repeat steps 3 through 7 to create an additional private variable named `employeeTitle` of type `String`.
12. Click **Save** in the main toolbar.
13. See the following section to learn how to declare these variables for a service that implements an activity in the BPD.

Declaring variables for a service

Declaring the variables for a service enables the service to display and manipulate the values that it receives from (input) and then pushes back up (output) to the main BPD.



If you use the Activity Wizard to create a service to implement an activity, you can pick the variables from the main BPD to use as input and output. When you do, the Activity Wizard automatically declares the variables for the resulting service and the steps outlined in this section are not necessary. See [Building a Human service](#) for an example. You can use the Activity Wizard to create Human, Rule, and General System services.

The following procedure describes how to declare both input and output variables for a service as a complete example. Depending on the logic of your BPD, a service may only require input or output variables and not both. And a service can include private variables for processing that occurs only within the service and does not involve any other activities in a process.

1. In Lombardi Authoring Environment, open the BPD for which you declared variables in the preceding section.
2. Drag an activity from the palette to the Participant lane in the BPD diagram.
3. While the activity is still selected in the diagram, click the Implementation option in the properties.
4. Click the **New** button and in the New Human Service dialog, type `MyService` in the **Name** text box and then click the **Finish** button.

The Designer opens the service diagram for the new service.

5. Drag a Coach component from the service palette to the diagram and then connect it to the Start and End events using the Sequence Flow tool.
6. Click the **Variables** tab in the Designer.
7. Click the **Add Input** button.
8. In the details for the variable, enter `employeeId` in the Name field.
9. Click the **Select** button next to Variable Type and select the `Integer` type from the list.
10. Click the **Add Input** button.
11. In the details for the variable, enter `employeeTitle` in the Name field.
12. Leave the variable type set to `String`, the default.
13. Click the **Add Output** button.
14. In the details for the variable, enter `employeeId` in the Name field.

Click the **Select** button next to Variable Type and select the `Integer` type from the list.

15. Click the **Add Output** button.
16. In the details for the variable, enter `employeeTitle` in the Name field.
Leave the variable type set to `String`, the default.
17. Click **Save** in the main toolbar.
18. Click the **Coaches** tab and notice that the variables you added are now available in the palette, which enables you to drag them to the Coach to automatically create related controls. (See [Building Coaches](#) for more information.)
19. See the following section to learn how to map these variables as inputs and outputs to the activity that is implemented by this service.

Mapping input and output data for an activity

The following steps describe how to use the Data Mapping tab to set the input and output data mapping for an activity in a BPD diagram. When developing processes in Lombardi, you must set the input and output mapping for each activity included in a BPD so that the variable values received and generated by the sub-processes and services that implement the activities map to the variables from the main BPD. The following procedure assumes that you are working with the same BPD, activity, and service that you created by following the instructions in the preceding two sections.

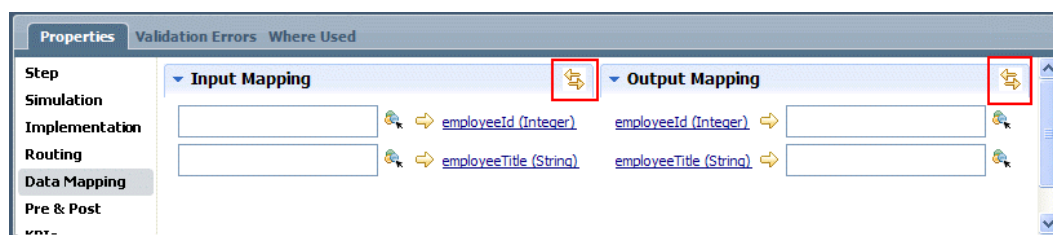


If you use the Activity Wizard to create a service to implement an activity, you can pick the variables from the main BPD to use as input and output. When you do, the Activity Wizard automatically maps the input and output data for the resulting service and the steps outlined in this section are not necessary. See [Building a Human service](#) for an example. You can use the Activity Wizard to create Human, Rule, and General System services.

The following procedure describes how to map both input and output data for an activity as a complete example. Depending on the logic of your BPD, an activity may only require input or output data and not both.

1. In Lombardi Authoring Environment, open the BPD that includes the service for which you declared variables in the preceding section.
2. Click on the activity in the BPD diagram and then click the Data Mapping tab in the properties.
3. Because you have already declared the variables for the service that implements this activity, the Data Mapping tab displays the variables that are available.

To complete the data mapping, click the auto-map icon in both the Input Mapping and Output mapping sections as shown in the following image:



This completes the required data mapping for the activity.



Auto-mapping works only when variable names and types match exactly. As discussed previously, you should always use an identical name and data type for a set of input and output variables that are passed in, processed, and then passed back.

When the mapping is complete, you can see that the input data mapping for this activity is set as follows:

BPD variable	Maps to service variable (service that implements activity)
<code>tw.local.employeeId</code>	<code>employeeID</code>
<code>tw.local.employeeTitle</code>	<code>employeeTitle</code>

And you can see that the output data mapping for this activity is set as follows:

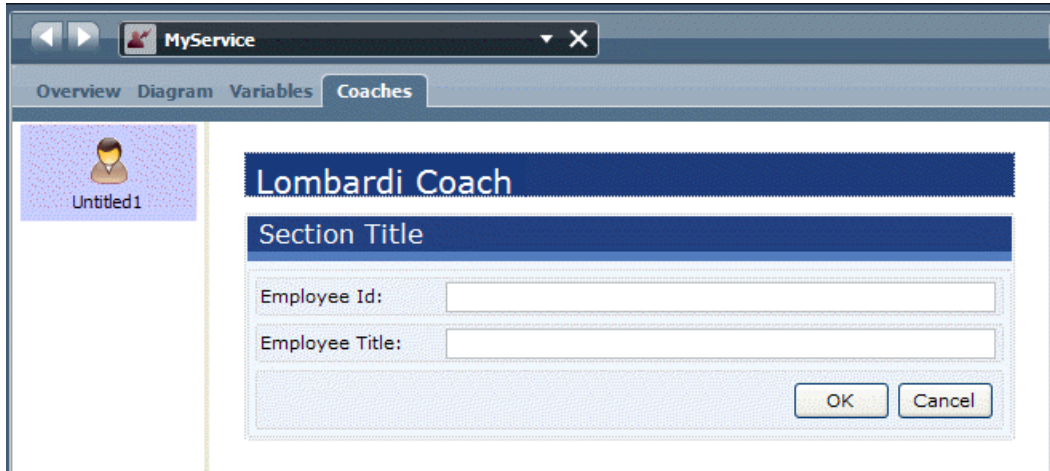
Service variable (service that implements activity)	Maps to BPD variable
<code>employeeID</code>	<code>tw.local.employeeId</code>
<code>employeeTitle</code>	<code>tw.local.employeeTitle</code>

4. Save your work.
5. See the following section to learn how to test the variables and the mappings you've established.

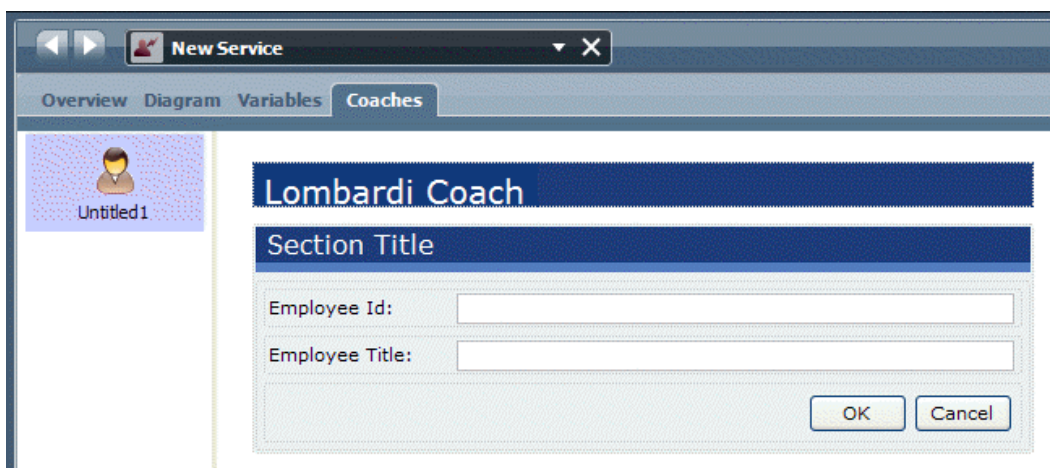
Testing declared variables and data mapping

To test the variables and data mapping established in the preceding sections, follow these steps:

1. In Lombardi Authoring Environment, open the BPD that includes the activity for which you established data mapping in the preceding section.
2. Open the service named `MyService`.
3. Click the **Coaches** tab and then click the only Coach listed to display its controls.
4. Remove the default controls.
5. Drag the input variables, `employeeId` and `employeeTitle` to the Coach form to automatically create controls as shown in the following image:



6. Open the BPD diagram, drag another activity from the palette to the Participant lane, and place it after the original activity.
7. While the new activity is still selected in the diagram, click the Implementation option in the properties.
8. Click the **New** button and in the New Human Service dialog, type `NewService` in the **Name** text box and then click the **Finish** button.
9. Click the **Variables** tab and add two input variables: `employeeId` of type `Integer` and `employeeTitle` of type `String`.
10. Click the **Diagram** tab.
11. Drag a Coach component from the service palette to the diagram and then connect it to the Start and End events using the Sequence Flow tool.
12. Click the **Coaches** tab and then click the only Coach listed to display its controls.
13. Remove the default controls.
14. Drag the input variables, `employeeId` and `employeeTitle` to the Coach form to automatically create controls as shown in the following image:

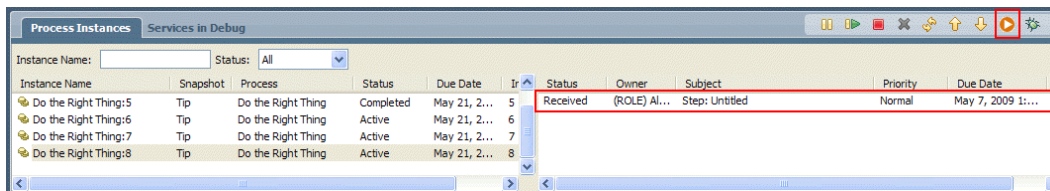


15. Open the BPD diagram.
16. Click the new activity and select the **Data Mapping** option in the properties.
17. Use the auto-map icon to map the BPD variables to the input variables that you declared for `NewService` (service that implements this activity).
18. Select the Sequence Flow tool from the BPD palette and establish the flow from the Start event through the activities and then to the End event.
19. Save your work.
20. Click the Run icon in the upper right to run the BPD in the Inspector.

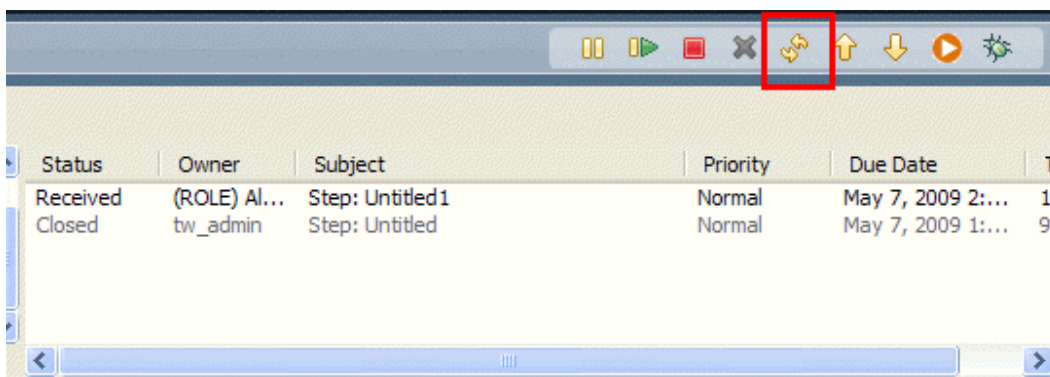


If prompted to switch to the Inspector, click the **Yes** button.

21. In the Process Instances tab in the Inspector, click the new task and then click the Run task icon as shown in the following image:



22. When prompted, select the user account that you want to use to run the task. (By default, activities in the Participant lane are assigned to the All Users group, which is why you are prompted to select a user account.)
23. When the coach for the first activity in the BPD opens in a browser, type an integer for the Employee ID and a string for the Employee Title and then click the **OK** button.
24. Click the Refresh icon in the Inspector toolbar as shown in the following image:



The task generated by the second activity is displayed as shown in the preceding image.

25. In the Process Instances tab, click the newly received task and then click the Run task icon.

26. When prompted, select the user account that you want to use to run the task. (By default, activities in the Participant lane are assigned to the All Users group, which is why you are prompted to select a user account.)
27. When the Coach for the second activity in the BPD opens in a browser, verify that the data that you entered into the first coach is displayed and then click the **OK** button.
28. Click the Refresh icon in the toolbar and you can see that this process instance completed in the Process Instances tab in the Inspector.

By successfully running the BPD, you have verified that the data is passed from the first activity to the second activity, enabling the process to complete.

Creating exposed process values (EPVs)

In the Designer in Lombardi Authoring Environment, you can create exposed process values (EPVs) to enable the end users that you specify to set or alter variable values while instances of a process are running. For example, if you create a process to handle expense reimbursement, you may want to enable supervisors to change the allowed amounts for daily expenditures, or the dollar amount that coincides with various levels of approvers. By creating EPVs, you can provide this type of flexibility, allowing end users to adjust specific variable values as constants, thereby affecting the flow of all running process instances, task assignments, and so on.

After you create an EPV, you must link it with a business process definition (BPD) or service as outlined in the following steps. Designated end users can set or alter EPVs in the Process Admin Console. See *Lombardi Administration Guide* or the online help for the Process Admin Console for more information.

1. Start Lombardi Authoring Environment and open the appropriate process application or toolkit in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. Click the plus sign next to the **Data** category and select **Exposed Process Value** from the list.
3. In the New Exposed Process Value dialog, enter a name for the value and click the **Finish** button.

The new EPV opens in the Designer, where you can configure the required settings.

4. In the Common section, enter a description in the Documentation text box so that other developers will understand the purpose of this EPV.
5. In the Details section, type an email address in the Feedback E-mail Contact text box so that end users can send their feedback regarding this EPV.

The Manage Exposed Process Values page in the Process Admin Console includes a Feedback link that uses this email address.

6. In the External Description text box, type the text to describe this EPV to end users.

The description that you provide here is displayed in the Manage Exposed Process Values page in the Process Admin Console.

7. In the Exposed Process Value Variables section, click the **Add** button to add a variable to this EPV.

For example, if you want to enable end users to adjust the dollar amounts that correspond with various levels of approvers for an expense reimbursement process, you should add a variable for each available level. For this sample scenario, you can add variables named `Level11`, `Level12`, and `Level13`.

- In the Variable Details section, supply the details for the variable most recently added or currently selected.

In the following example, the details for the Level 1 variable have been set:

- In the External Name text box, type the name of the variable as you want end users in the Process Admin Console to see it.

This name appears in the Variable List for this EPV in the Process Admin Console.

- In the Variable Name text box, type the name of the variable as you want it represented internally for processing.



Variable names should start with a lowercase letter, with subsequent words capitalized like so: myVar. Do not use underscores or spaces in variable names. Variable names are case sensitive.

- In the External Description text box, type the text to describe this variable to end users.

This description appears in the Variable List for this EPV in the Process Admin Console.

- In the Default Value text box, type a valid default for this variable.
- Enable the check box named In-Progress Tasks Use New Values if you want tasks that are currently running to use an updated value when the end user makes a change in the Process Admin Console.
- Click the **Select** button next to Variable Type and select the type that you want from the list.

Variable types from the System Data Toolkit are listed as well as any custom variable types that you have created or to which you have access via a toolkit. You can also click the New button to create a new variable type. See [Creating custom variable types](#) for more information.

- Repeat steps 9 through 14 for each variable that you add.

16. In the Exposing section, click the **Select** button to choose the participant group whose members can manage this EPV and adjust its variable values.
17. Click **Save** in the main toolbar.
18. Open the BPD or service to which you want to link this EPV.
19. Click the Variables tab.
20. Click the Link EPV button and select the EPV (that you created in the preceding steps) from the list.
21. Click **Save** in the main toolbar.
22. After you link the EPV to a BPD or service, you can refer to it anywhere in the BPD or service.

You can reference the name of the EPV and its variables like so:

`tw.epv.[epv_name].[epv_variable_name]`. The auto-complete feature in the Designer assists you with this syntax.

You can use the EPV in a decision gateway to control the flow of a process. You can also reference the EPV from any JavaScript code in a linked BPD, such as the code within a Server Script service component.

Now when you run the BPD or service to which the EPV is linked, you can go to the Process Admin Console to adjust the variable values in the Manage Exposed Process Values page. See *Lombardi Administration Guide* or the online help for the Process Admin Console for more information.



You can also add EPVs to a report to enable end users to adjust variable values directly from a report. To add EPVs to a report, simply open the report in the Designer in Lombardi Authoring Environment and select the EPVs that you want using the **Add** button in the Exposed Process Values section.

Setting variables in pre and post assignments

You can set pre and post assignments for variables when you want to assign a value to a variable immediately before or after an activity or event executes in a running process. You can also set pre and post assignments for components in a service, such as a Coach or a Server Script component.

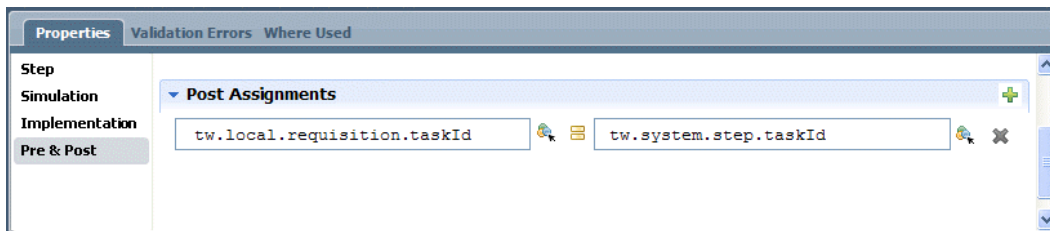
For example, if you want to send an email message to end users as soon as an activity is active and can be completed, you can attach a timer event to the activity and use a post assignment to place the task ID into a variable so that it can be passed to the follow-on activity that sends the email message. The task ID is needed so that the email message sent to end users includes information about the task to complete.

1. In the Designer in Lombardi Authoring Environment, open a BPD that includes an activity or event that requires a pre or post assignment.
2. Click the activity or event in the BPD diagram and then select the Pre & Post option in the properties.
3. To add an assignment, click the plus sign for either Pre Assignments or Post Assignments.

For this example, click the plus sign for Post Assignments to assign the task ID to a variable as described in the preceding example scenario.

4. In the text box on the left, click the variable icon to choose the variable that you have declared in the current BPD to contain the task ID.

- In the text box on the right, type `tw.system.step.taskId`. (The type-ahead feature in the Designer enables you to see all available Lombardi JavaScript objects.)



With the post assignment, the ID for the task is set and available as input for the follow-on activity and the service that you build to implement it.

Using JavaScript methods on variables

As described in [Variable scope in Lombardi](#), all Lombardi variables are JavaScript objects, which enables you to use standard JavaScript methods on Lombardi variables. The methods used must correspond to the variable type.

For example, you can call the `toUpperCase()` or `toLowerCase()` JavaScript method on a Lombardi variable of type `String`. In the following example, the results of the call are used to assign a new value to the variable:

```
tw.local.myString = tw.local.myString.toUpperCase();
```

Initializing complex variables and lists

Any structures that you declare in Lombardi are considered complex variables. In Lombardi, all complex variables and all lists (arrays) must be initialized before you use them in a BPD or service. If you do not initialize a complex variable or list, you may receive run-time errors or notice that the Coach controls to which the variables are bound do not behave as expected.

Before using a complex variable, initialize it by using a script like the following:

```
tw.local.requisition=new tw.object.Requisition();
```

In the preceding script, the name of the variable that is being initialized is `tw.local.requisition`. The name of the complex variable type is `Requisition`.



If your complex variable type includes element that are themselves complex variables, then you must initialize them before you use them.

You must also initialize lists before you use them. If you have a list of `Strings`, you can initialize them using a script like the following:

```
tw.local.yourStringList = new tw.object.listOf.String();
```

To see a sample of how complex types are properly initialized, open the Quick Start Tutorial process application in the Designer. (For instructions, see the *Quick Start Tutorial Guide* or online help.) Open the Submit Requisition service and in the service diagram, click the server script named Initialize Output to select it. Then click the Implementation option in the properties. You can see the scripts used to initialize the complex variables that are included in the BPD.

Making variables available for Process Portal searches

If you want process participants in Lombardi Process Portal to be able to search for particular business data across process instances, follow the steps in this procedure to properly configure the variables that need to be available.

1. In the Designer in Lombardi Authoring Environment, open the BPD that includes the variables you want to configure.
2. Click the **Variables** tab in the Designer.
3. For each variable whose runtime values you want to search, enable the **Available in Search** check box in the Business Data Search section. For complex variables, be sure to enable the check box for each parameter you want to search.
4. Type a name for the variable in the **Search Alias** text box. This is the name to use when performing searches in Lombardi Process Portal.



If a variable is shared by multiple BPDs (for example, a parent process and its nested processes) and you want the variable to be searchable in all of those processes, you must define the same search alias for the variable in each of the BPDs where it is used.

5. Save your changes.

Now when Lombardi runs instances of the BPDs that contain the configured variables, you can search for process instances that include these variables in Lombardi Process Portal.

Handling exceptions



When the processes that you develop in Lombardi include integrations with external systems, server scripts, and other complex implementations, you need to anticipate potential exceptions and create the components required to handle those exceptions when they occur. For example, if a BPD includes an integration with a database system, that database may not be available when each new instance of the BPD executes. So, when you develop the integration in Lombardi, you need to build in exception handling to detect errors and recover in a predictable manner.

You can build error handling capabilities into both BPDs and services using the available exception components. For more information, see:

- [Using exception events in BPDs](#)
- [Using exception components in services](#)

Using exception events in BPDs

For BPDs, you can include the following exception events:

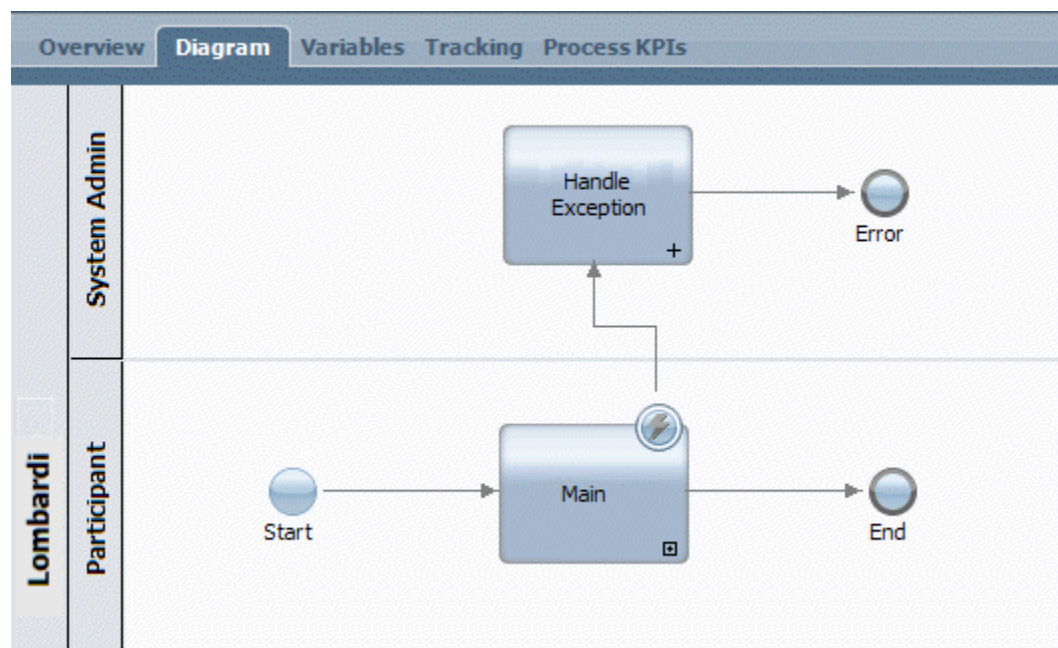
BPD event	Description
 Intermediate Exception Event	Use to catch process execution exceptions and handle exceptions with an error handler activity or further process flow.
 End Exception Event	Use to throw an exception to parent processes.

When including exception events in your BPDs, keep the following points in mind:

- You should attach intermediate exception events to an activity.
- Do not include intermediate exception events in the process flow as generic error handlers. Generic error handlers do not work in BPDs.
- You can attach only one intermediate exception event to an activity.
- If an exception occurs while a process is executing an activity with an attached exception event, the process flows along the sequence line that is attached to the exception event.
- If an exception occurs and there is no exception event attached to the activity where the exception occurs, the exception is propagated up the BPD call stack until it reaches a nested process that contains an activity with an exception event attached to it, or the top of the stack is reached. If an exception is not caught before it reaches the top of the stack, the BPD execution fails.
- Lombardi creates an XML representation of process exceptions that occur. An exception is captured as an `XMLElement` in `tw.system.error`.

The best way to implement error handling is from the bottom up. Build each nested process and service so that exceptions can be captured and corrected. If a correction is not possible at the lowest level implementation, you can allow the exception to bubble up (by not including an exception event) or include an exception event to re-throw the error to the calling service or process. ([Using exception components in services](#) shows how to design a service so that active exceptions are thrown.)



For example, to ensure that any exceptions that might occur during process execution are captured, you can create a high-level BPD that includes an activity (to call the main process as a nested process) and then one additional activity (with an underlying service) to implement error handling as shown in the following image:



This type of design ensures that any exceptions thrown from underlying processes and services are propagated up and handled appropriately. See [Using exception components in services](#) to learn more about catching and throwing exceptions in services.

Using exception components in services

For services, you can include the following exception components:

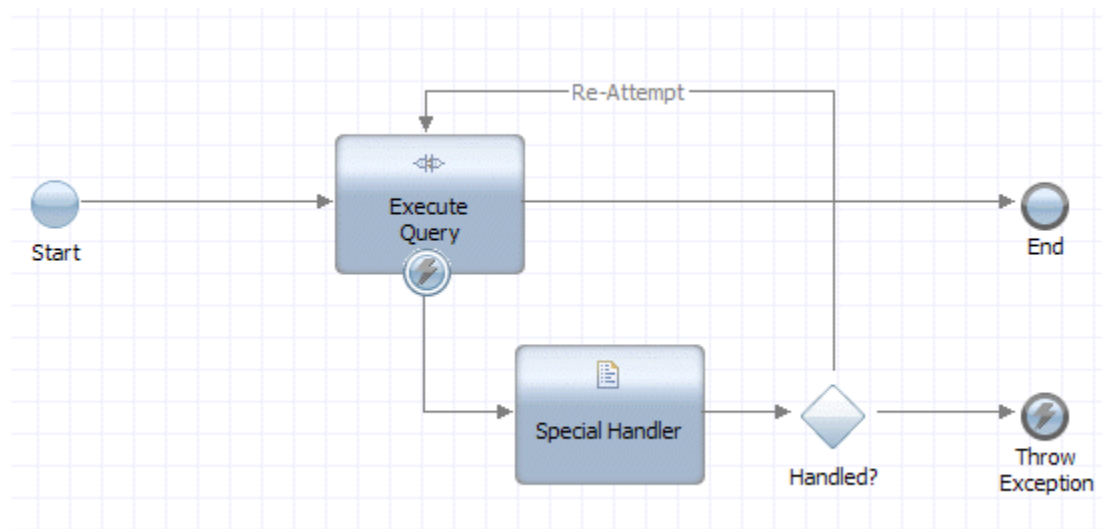
Service component	Description
 Catch Exception	Listens for exceptions from the service component to which it is attached.
 Throw Exception	Use to purposely throw an error and end processing. You might, for example, use a Throw Exception component if you return too many rows from a database (over a limit that is normal and would bog down the server).

When including exception components in your services, keep the following points in mind:

- You should attach Catch Exception components to other components in your service. It is possible to include Catch Exception components in the service flow so that they can act as global error handlers within the service. However, this practice is not recommended.
- You can attach only one Catch Exception component to another component in a service, such as a script or integration component.

When building services that include integrations with external systems or other complex implementations, you should use the Catch Exception component to ensure that errors are captured and end users (such as administrators) are informed and, when possible, given an opportunity to correct the problems. Be sure to include the appropriate logic and a Throw Exception component in your services to throw exceptions in cases where they still exist after attempts to handle them. Throwing the exceptions ensures that the errors are passed up to parent processes and services and are eventually handled by a high-level process like the one shown in [Using exception events in BPDs](#).

For example, the following image shows a service that catches an exception for a database query, attempts to handle the exception with a special script, and then (if the handler script fails) throws the exception for further processing by higher level services or processes.



Creating loops

Lombardi provides several ways to create and implement loops. For example, you can include a script component in a service that iteratively processes records that you retrieve from a database until all records have been processed. Since you can include JavaScript throughout your implementations, you can easily develop the logic required to repeat an action until a certain condition is true.

In addition to implementing loops with scripts, the activities that you add to your BPDs can be configured for simple and multi-instance looping as described in the following table. When you want the run-time task that results from an activity to be performed more than once, you can configure looping behavior for that activity.

Loop type	Description
Simple loop	When you model an activity with simple looping, the required number of instances is dynamically created, up to the loop maximum value that you specify. A simple-looping activity is executed sequentially until the last instance of the activity has been performed. When you run an activity configured for simple looping, a single token is generated and used for each instance of the activity, which, in effect, recycles the run-time task.
Multi-instance loop	Multi-instance looping provides a way to dynamically execute multiple unique instances of the same activity sequentially or in parallel. When you run an activity configured for multi-instance looping, a unique token is created for each instance of the activity. (For more information about tokens, see Understanding tokens .)

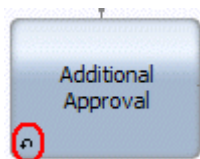
Configuring an activity for simple looping

1. Click to select the activity that you want to configure in the BPD diagram.
2. Click the Step option in the properties.
3. Under Behavior, select the **Simple Loop** option from the `Loop Type` drop-down list.
4. Under Simple Looping, type a value in the `Loop Maximum` text box. This value sets the maximum number of instances that can be created at run time.

If you have declared a variable that can be used for this setting, click the variable icon to select it or type the variable name into the `Loop Maximum` text box.

5. In the `Loop Condition` text box, type an optional JavaScript condition to dictate the run-time looping behavior. A condition is evaluated before any instances are created from the activity. If the condition is not met, the looping does not occur.
6. Save your changes.

When you configure an activity for simple looping, the activity includes the indicator shown in the following image:



Configuring an activity for multi-instance looping

1. Click to select the activity that you want to configure in the BPD diagram.
2. Click the Step option in the properties.
3. Under Behavior, select the **Multi Instance Loop** option from the Loop Type drop-down list.
4. Under Multi Instance Looping, type a value in the Start Quantity text box. This value sets the number of instances created at run time.

If you have declared a variable that can be used for this setting, click the variable icon to select it or type the variable name into the Start Quantity text box.

5. From the Ordering drop-down list, select one of the following options:

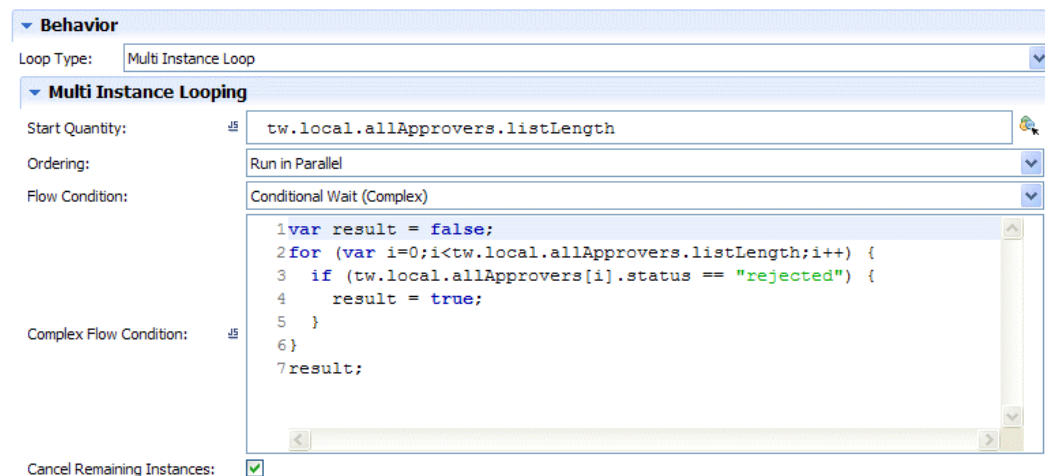
Run Sequential	Resulting instances are executed sequentially until the last instance of the activity has been performed.
Run in Parallel	Resulting instances are executed at the same time until all instances are finished or until the condition that you specify has been met.

6. For parallel ordering, select one of the following options from the Flow Condition drop-down list:

Wait for all to finish (All)	Looping continues until all resulting instances of the activity are finished.
Conditional Wait (Complex)	Looping continues until the condition that you specify in the following step is met.

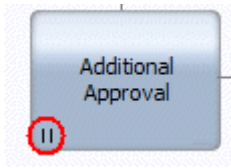
7. For complex flow conditions, type the JavaScript to implement that condition in the Complex Flow Condition text box.
8. Enable the Cancel Remaining Instances checkbox if you want active instances of the activity to be canceled when the preceding condition is met.

The following image shows an activity configured for a multi-instance loop that uses a variable for its start quantity and a JavaScript expression for the complex flow condition:



9. Save your changes.

When you configure an activity for simple looping, the activity includes the indicator shown in the following image:



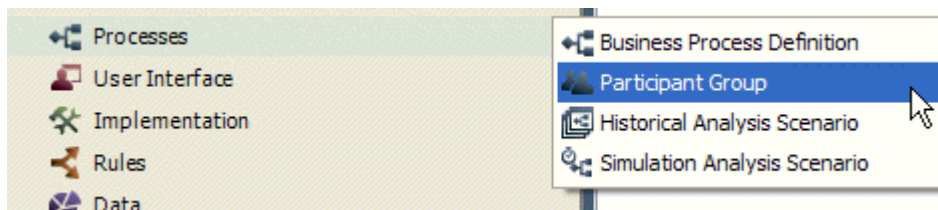
Helpful reference information

Creating a participant group

Participant groups represent the groups of users in your enterprise.


To create a participant group and add users to that group, follow these steps:

1. In the Designer view, click the plus sign next to **Processes** and select **Participant Group** from the list of components:



2. In the New Participant Group dialog, enter a name for the group and click **Finish**.
3. Lombardi Designer displays the properties for the participant group. Supply requested information, including:

Dialog area	Field or link	Description
Common	Name	This field displays the name you provided in step 2.
	Documentation	Optionally provide a description of the group in this field.
Simulation Properties	Capacity	Choose either Use Provider Users or Use Estimated Capacity. If you select Use Provider Users, the capacity is determined by the number of users in the group. If you choose to estimate, provide the maximum number of users that this group can include.
	Availability	For simulation purposes, specify the percentage of this group's working hours that are available to complete Lombardi tasks.
	Efficiency	For simulation purposes, specify the efficiency of this group as a percentage.
	Cost per Hour	For simulation purposes, provide the cost (in dollars and cents) to your organization for each hour of work performed by this group.
Members	Select	Click the drop-down list to choose how you want to define the members of this group. You can choose Standard Members or Using Expression. The choice you make determines the information required.
	If you select Standard Members...	Use the buttons provided to add the users and groups that you want. You can add existing participant groups as members. You can also add users and groups previously defined in the Process Admin Console.

Dialog area	Field or link	Description
		See <i>Managing Lombardi users</i> in the <i>Lombardi Administration Guide</i> for more information.  When adding users and groups, type in part of the name of the account that you want and Lombardi displays all users or groups that match.
	If you select Using Expression...	First establish whether you want to select members based on a match with any or all rules that you define. Then use the sentence editor to establish the rule or rules that you want. See Defining Participant Group rules to learn how to define rules. If no users match the expression, the resulting participant group is empty.

4. Click **Save** in the main toolbar.

Defining Participant Group rules

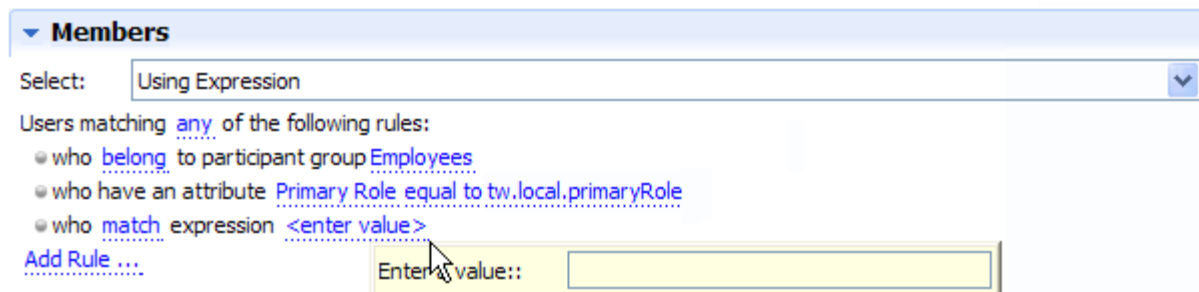
When you choose to establish participant group members by using an expression, you can define rules to determine those members. To define rules, follow these steps:

1. Click **Add Rule** to choose the type of rule you want.

When defining rules, you have the following types from which to choose:

Participant Rule	Allows you to select users according to participant group membership.
User Attribute Rule	Allows you to select users based on user attributes. To learn how to create user attribute definitions, see Creating a user attribute definition .
Expression Rule	Allows you to select users who match a particular expression that you provide.

The following figure shows an example of the types of rules you can define:



2. Supply the necessary information for the type of rule that you choose.
 - For a Participant Rule, supply the input that you want for the following specification:

Who *belong* to participant group *select participant*.

<i>belong</i>	Click <i>belong</i> to choose either belong or do not belong .
<i>select participant</i>	Click <i>select participant</i> to choose an existing participant group.

- For a **User Attribute Rule**, supply the input that you want for the following specification.



To learn how to create user attribute definitions, see [Creating a user attribute definition](#).

Who have an attribute *select user attribute equal to enter value*.

<i>select user attribute</i>	Click <i>select user attribute</i> to select an existing user attribute definition.
<i>equal to</i>	Click <i>equal to</i> to choose from: equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to .
<i>enter value</i>	Click <i>enter value</i> to display a field in which you can enter either a Lombardi variable or a JavaScript expression that produces the value that you want to compare. Be sure to surround any strings in the expression with double quotation marks.

- For an **Expression Rule**, supply the input that you want for the following specification:

Who *match* expression *enter value*.

<i>match</i>	Click <i>match</i> to choose either match or do not match .
<i>enter value</i>	Click <i>enter value</i> to display a field in which you can enter either a Lombardi variable or a JavaScript expression that produces the value that you want to compare. Be sure to surround any strings in the expression with double quotation marks. The expression must evaluate to a specific user name.

Creating a user attribute definition

User attribute definitions allow you to associate unique capabilities or qualities with one or more users. You can then use the defined attributes when you create participant groups based on a user attribute rule. See [Defining Participant Group rules](#) for more information.

To create a user attribute definition:

- In the Designer view, click the plus sign next to **Data** and select **User Attribute Definition** from the list of components.
- In the New User Attribute Definition dialog, provide a unique name for the attribute, and click **Finish**.
- Supply requested information about the user attribute definition, including:

Dialog area	Field or link	Description
Common	Name	This field displays the name you provided in step 2.
	Documentation	Optionally provide a description of the attribute in this field.
Type	Variable type	The default variable type is String. If you want a different type, click Select to choose the type you want. Click New if you want to define a new variable type.
Obtain current value from...	Source	The source is Lombardi.
Obtain possible values from...	Source	Click the drop-down list to choose the source for the possible values of the user attribute. You can choose Any Allowed or List . The choice you make determines the information required.

Dialog area	Field or link	Description
	If you select Any Allowed...	This means that the possible values for the attribute are limited only by its variable type.
	If you select List...	Enter each possible value in the Value field and click Add . You can remove values from the list by clicking Remove or change the order of the values that will be displayed by using the Up and Down buttons.

- Click the **Save** icon in the main toolbar.

Lombardi Authoring Environment saves the user attribute definition, which allows you to use the attribute when creating participant groups.

Routing activities

For any activity with a Lombardi service implementation, you can designate the end users who should receive the run-time task using the Routing option in the activity's properties. You can also control how the run-time task is distributed to designated users.



The routing options are available only for those activities with a Lombardi service implementation.

- In the Designer view, click an activity in a BPD diagram to display its properties.
- Click the **Routing** option.
- From the **Assign To** drop-down list, chose one of the following options:

Last User in Lane	Routes the run-time task to the user who completed the activity that immediately precedes the selected activity in the swim lane. Do not select this option for the first activity in a lane unless the activity is a service in a top-level BPD and a Start Event is in the lane. In this case, the run-time task is routed to the user who started the BPD.
Lane Participant	Routes the run-time task to the participant group assigned to the swim lane in which the selected activity resides. This is the default selection.
Routing Policy	Routes the run-time task according to the policy that you establish. See Setting up a routing policy for more information.
List of Users	Routes the run-time task to an ad-hoc list of users. See Routing to an ad-hoc list of users for more information.
Custom	Routes the run-time task according to the JavaScript expression that you provide in the corresponding text field. To select one or more variables for your expression, click the variable selection icon next to the text field. The JavaScript expression should produce results such as <code>USER:<user_name></code> or <code>ROLE:<group_name></code> , where <code>user_name</code> is the name of a Lombardi user (such as <code>tw_author</code>) and <code>group_name</code> is the name of a Lombardi security group (such as <code>tw_authors</code>).

- From the **User Distribution** drop-down list, choose one of the following options:

None	Lombardi assigns the run-time task to all potential users. This is the default setting.
Load Balance	From the potential users who can receive the run-time task, Lombardi assigns to the users who have the fewest number of open tasks.

Round Robin	From the potential users who can receive the run-time task, Lombardi assigns to users in a round-robin fashion. For example, if the users in the Call Center participant group should receive the run-time task, Lombardi assigns each task (created by each process instance) in a series-to one user in the group after another.
-------------	--

Setting up a routing policy

One of the options when routing the run-time tasks that are generated by activities, is to establish a routing policy. When you define a policy, the users who receive the run-time task are determined by the conditions that you specify.



Before you can configure a routing policy, you must declare variables for your BPD. See [Declaring and passing variables](#) for more information.

1. In the Designer view, open the diagram of your BPD and select the activity that you want to route.



The activity that you choose must already have an attached service.

2. Click the **Routing** option in the properties.
3. From the **Assign To** drop-down list, select **Routing Policy**.
4. Under Routing Conditions (If), click **Add a column** to select the variable to use to begin specifying conditions.

When defining routing conditions, you create a table in which each column represents a variable and each row represents a rule.

5. From the displayed list, choose the variable for which you want to specify a condition.
6. In the first field in row 1, enter the value to compare to the variable.

For example, if you choose a variable called `customer` (String) in the preceding step and that variable holds customer names, enter a customer name in the field.

7. If you want to add another variable to the routing condition, click **Add a column** and choose a variable from the displayed list. Enter the value to compare to the second variable.

If you add two columns (variables) and two rows (rules), your routing conditions table resembles the following figure:

	customer	loanAmount	Adv	Assign To
1	"Jones Hardware"	>5000	<input type="checkbox"/>	Swimlane
2	"Smith Lumber"	<200000	<input checked="" type="checkbox"/>	Swimlane
3			<input type="checkbox"/>	

If no condition matches, Teamworks will assign to [Swimlane](#)

The following examples illustrate the syntax for possible values in the variable columns:

Enter...	To match...
"ok"	The exact string, ok (no quotes)
"A" "B"	Either of the strings A or B
>100	Any number greater than 100
<100	Any number less than 100
!=3	All numbers except 3



You can use && for AND conditions and | for OR conditions in the variable columns.

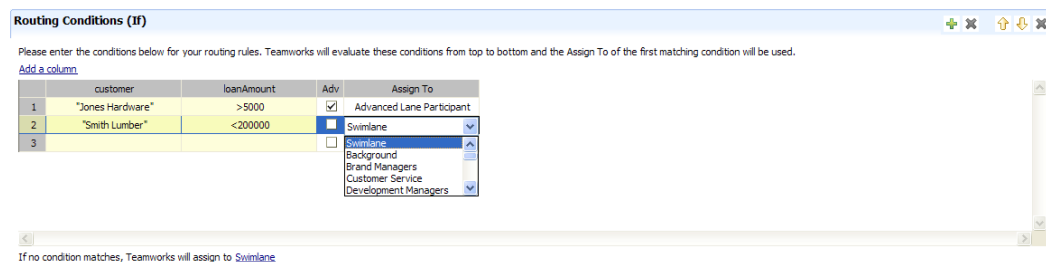
- If you want to establish advanced routing rules, click the **Adv** check box to enable it. When you establish routing rules, you create specifications that determine who will receive the run-time task, such as only those users who have a previously defined user attribute.

To establish rules, click **Add Rule** in the Advanced Assigned To (Then) section of the Routing properties and see [Defining rules](#) for instructions.



Lombardi creates only one set of rules under Advanced Assigned To (Then) for the entire Routing Conditions table. If you want to remove a rule after you define it, click on Advanced Lane Participant in the Assign To field in the routing conditions table to display the rule or rules for that routing condition. Under Advanced Assigned To (Then), click the bullet before the rule that you want to remove and then click Remove.

- If you do not click the **Adv** check box, the Assign To field in the routing table shows the default assignee, Swimlane, which means that the run-time task is routed to the participant group assigned to the swimlane in your BPD. If you have multiple participant groups defined in your current process application, you can click the drop-down indicator in this field to pick one of those groups as show in the following figure:



When you define routing conditions, Lombardi evaluates the conditions in the table from top to bottom. Lombardi implements the first condition that matches. If no conditions match, Lombardi assigns the activity to the default assignee, Swimlane. (As described in the preceding step, you can change the default assignee to any participant group in the current process application.)

When working with the routing conditions table, the set of icons on the left control the columns (variables) and the icons on the right control the rows (rules). See the following figure for more detail.

Control Columns **Control Rows**

Routing Conditions (If)

Please enter the conditions below for your routing rules. Teamworks will evaluate these conditions from top to bottom.

[Add a column](#)

	customer	loanAmount	Adv	Assign To
1	"Jones Hardware"	>5000	<input checked="" type="checkbox"/>	Advanced Lane Participant
2	"Smith Lumber"	<200000	<input type="checkbox"/>	Swimlane
3			<input type="checkbox"/>	

Defining rules

When you establish a Routing Policy for an activity, you can establish rules to determine the users who receive the activity as a run-time task. Using rules, your task assignments can be dynamic, which helps guarantee that activities are routed to the proper individuals.

When defining rules, you have the following types from which to choose:

Swimlane	Allows you to route activities to users based on whether they are the default lane participants.
Participant Rule	Allows you to select users according to participant group membership.
User Attribute Rule	Allows you to select users based on user attributes.
Expression Rule	Allows you to select users who match a particular expression that you provide.

- Under **Advanced Assigned To (Then)**, click *all* in the following statement:

Advanced Lane Participants are users who match *all* of the following rules:

Choose **all** or **any**.

If you choose **all**, the run-time task is assigned to users who match all specified rules. If you choose **any**, the run-time task is assigned to users who match at least one of the specified rules.

If none of the conditions that you specify are met, Lombardi assigns the task to the Swimlane for the rule to which this Advanced Assignment applies.

- Click **Add Rule** to choose the type of rule you want.

The following figure shows three routing rules established for an Advanced Assignment:

▼ Advanced Assign To (Then)

Advanced Lane Participant are users who match all of the following rules:

- who belong to participant group Finance Managers
- who have an attribute Level 1 Loans greater than tw.local.loanAmount
- who match expression tw.local.Recipient

[Add Rule ...](#)

3. Supply the necessary information for the type of rule that you choose.

- For a **Swimlane** rule, supply the input that you want for the following specification:

Who *belong* to lane participant.

<i>belong</i>	Click <i>belong</i> to choose either belong or do not belong .
---------------	--

- For a **Participant Rule**, supply the input that you want for the following specification:

Who *belong* to participant group *select participant*.

<i>belong</i>	Click <i>belong</i> to choose either belong or do not belong .
<i>select participant</i>	Click <i>select participant</i> to choose a participant group from the library.

- For a **User Attribute Rule**, supply the input that you want for the following specification:

Who have an attribute *select user attribute equal to enter value*.

<i>select user attribute</i>	Click <i>select user attribute</i> to select a user attribute definition from the library.
<i>equal to</i>	Click <i>equal to</i> to choose from: equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to .
<i>enter value</i>	Click <i>enter value</i> to display a field in which you can enter either a Lombardi variable or a JavaScript expression that produces the value that you want to compare. Be sure to surround any strings in the expression with double quotation marks.

- For an **Expression Rule**, supply the input that you want for the following specification:

Who *match* expression *enter value*.

<i>match</i>	Click <i>match</i> to choose either match or do not match .
<i>enter value</i>	Click <i>enter value</i> to display a field in which you can enter either a Lombardi variable or a JavaScript expression that produces the value that you want to compare. Be sure to surround any strings in the expression with double quotation marks. The variable or expression must evaluate to a specific user name.

Routing to an ad-hoc list of users

When you need to route an activity to a group of users that is defined dynamically when a BPD instance is running, Lombardi enables you to route to an ad-hoc list of users. The ad-hoc group (or list) of users is maintained as long as the process instance exists on the runtime Process Server.



To route to an ad-hoc list of users, you must create an activity with an underlying service (that is upstream from the activity to be routed) to dynamically generate the list of users. The activity that generates the list of users must include an output variable to identify the list of users to the follow-on activity that you want to route.

1. In the Designer view, open the diagram of your BPD and select the activity that you want to route.
2. Click the **Routing** option in the properties.
3. From the **Assign To** drop-down list, select **List of Users**.
4. Click the **Select** button next to Binding.
5. Choose the variable to use to bind the list to the activity to be routed.

For example, you can define a new complex variable that is a list (array) to pass the list of users from the service that generates the list to the activity to be routed as shown in the following example:

Service Variables

Using the complex variable shown in the preceding example, you can pick the `id` parameter to bind the list to the activity that you are currently routing as shown in the following image:

To learn more about working with variables in Lombardi, see [Managing and mapping variables](#).

Example gateways

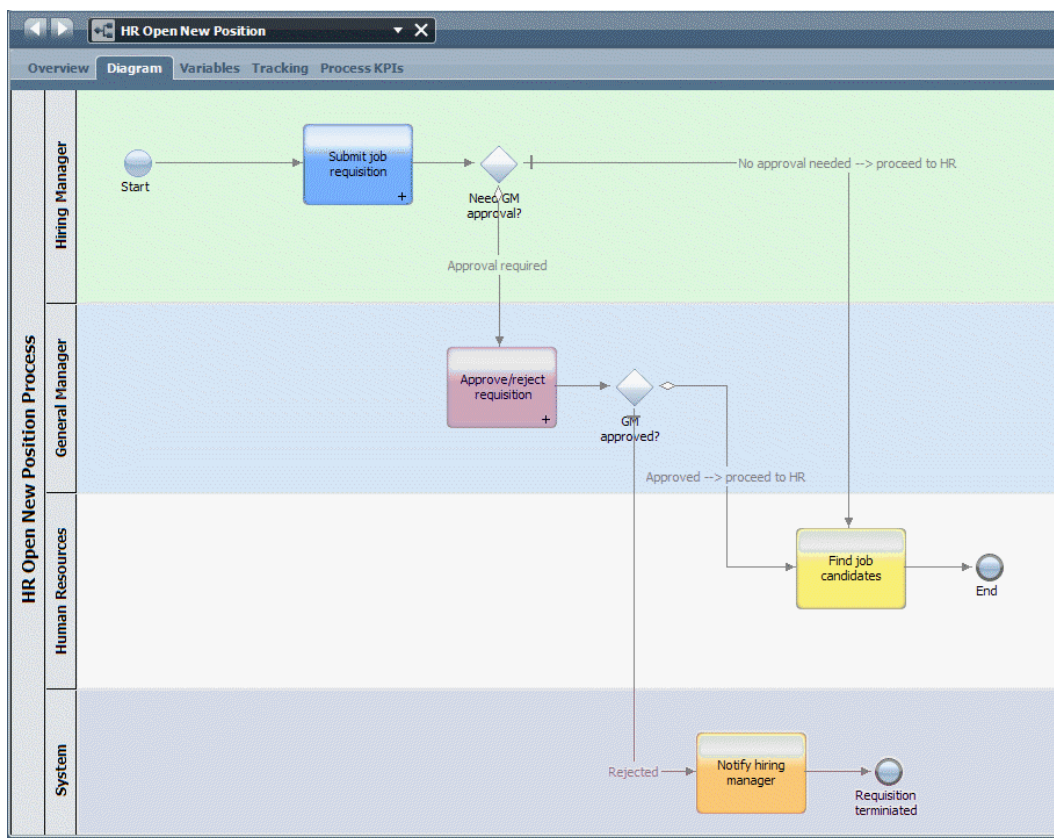
When modeling processes in Lombardi, you have several options for implementing gateways. See [Using gateways](#) to understand the available options and to see a sample implementation of a simple split. Review the following samples to learn more about decision gateways and conditional splits.



To implement decision gateways and conditional splits in a BPD, you need to declare variables for that BPD as described in [Declaring and passing variables](#).

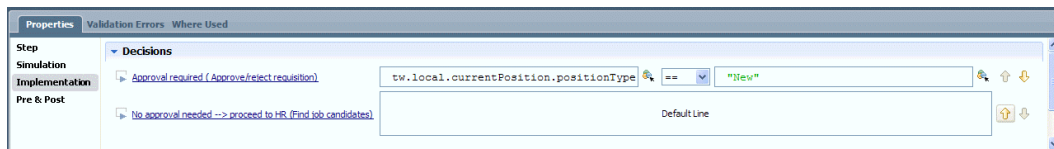
Sample decision gateways

You should use a decision gateway in your BPD when you need to model a point in the process execution where *only one* of several paths can be followed, depending on a condition. For example, the gateways shown in the following BPD are both decision gateways:



You can access the BPD shown in the preceding image (named `HR Open New Position`) in the Quick Start Tutorial process application. For more information, see the *Quick Start Tutorial Guide* or online help.

The first gateway, named `Need GM Approval?`, determines which path to follow based on whether the submitted job requisition requires approval. To see how this works, click the `Need GM Approval?` gateway in the BPD diagram to select it and then click the `Implementation` option in the properties:



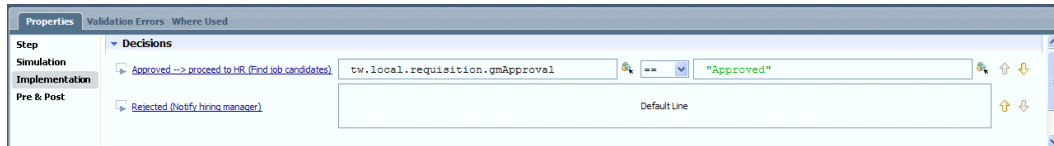
The `Approval required` path is followed to the `Approve/reject requisition` activity only when the `tw.local.currentPosition.positionType` variable is equal to `"New"`. This logic ensures that those requisitions from Hiring Managers for new headcount are approved by General Managers before HR

processing. If a position is not new, the process follows the default path to the `Find job candidates` activity.



For conditional and decision gateways, decisions in the Implementation properties are evaluated from top to bottom and the path for the first decision that evaluates to true is followed. If no decisions evaluate to true, the default path is followed.

The second gateway, named `GM Approved?`, determines which path to follow based on whether a new position has been approved. To see how this works, click the `GM Approved?` gateway in the BPD diagram to select it and then click the Implementation option in the properties:



The `Approved --> proceed to HR` path is followed to the `Find job candidates` activity only when the `tw.local.requisition.gmApproval` variable is equal to `"Approved"`. This logic ensures that those requisition requiring approval are actually approved before HR processing. If a requisition is not approved, the process follows the default path (`Rejected` path) to the `Notify hiring manager` activity.

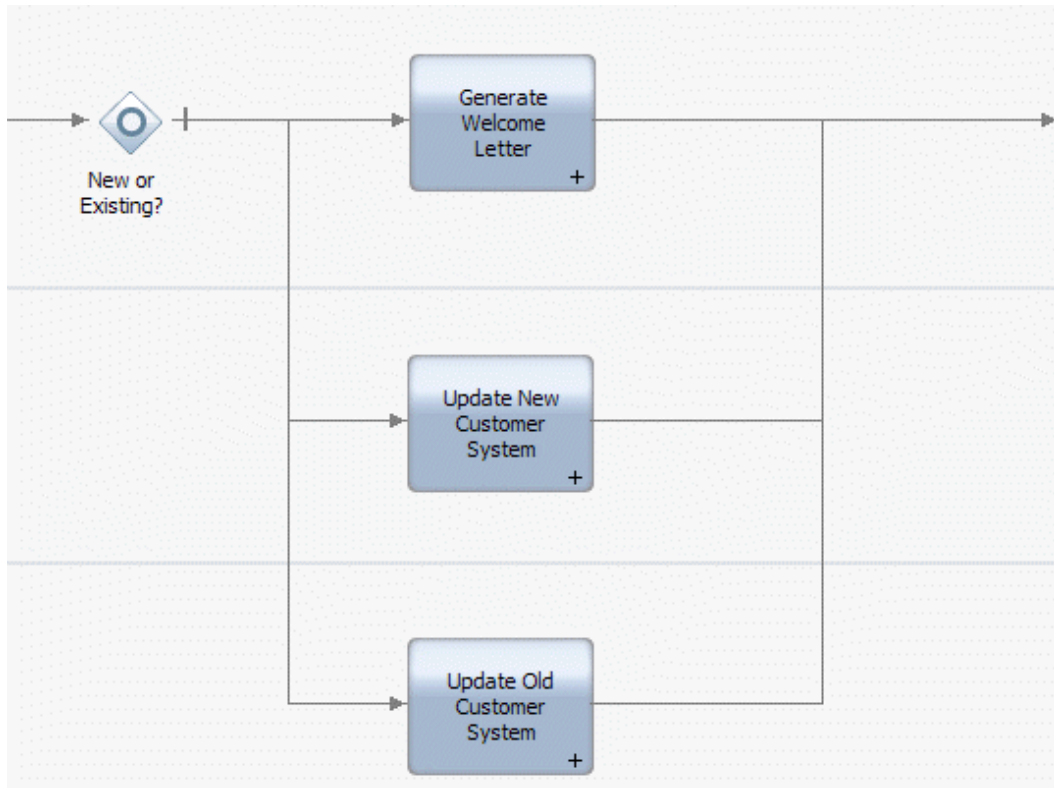
Sample conditional split

You should use a conditional split in your BPD when you need to split, or diverge, the process along more than one path and you want to follow *one or more* available paths based on conditions that you establish.

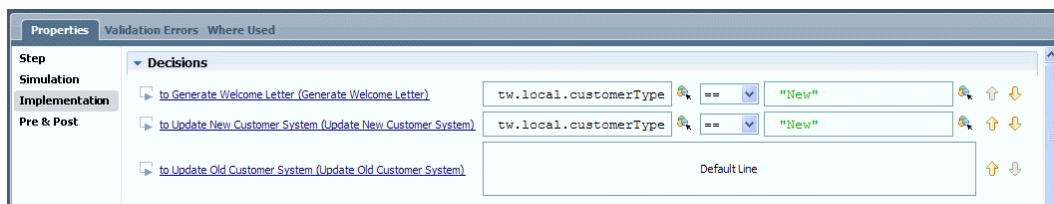


Conditional splits can follow a maximum of $n-1$ paths. So, if you model a conditional split with 3 paths, the process can only follow 2 of those paths.

For example, suppose that you want to model a process where the steps are different based on whether the customer type is new or existing. For new customers, you want two activities to be completed. For existing customers, only one activity is needed. You can use a conditional split for this type of process as shown in the following image.



With decision gateways, only one available path is followed from the gateway. With conditional splits like the one shown in the preceding image, one or more paths from the gateway can be followed. The conditional split gateway shown in the preceding example (named `New or Existing?`) determines the path(s) to follow based on the type of customer being processed. The conditions for this split are configured in the implementation properties for the gateway as shown in the following image:




The conditions control behavior as follows:

- If the value of the `tw.local.customerType` variable is "New", the path to the `Generate Welcome Letter` activity is followed.
- If the value of the `tw.local.customerType` variable is "New", the path to the `Update New Customer System` activity is also followed.
- If none of the preceding conditions evaluate to true, the path to the `Update Old Customer System` activity is followed.

Using this logic, you are able to execute two separate activities for new customers and a different activity when the customer is an existing one.

Lombardi naming conventions

When developing BPDs in Lombardi Authoring Environment, the following character limitations and other conventions are enforced:

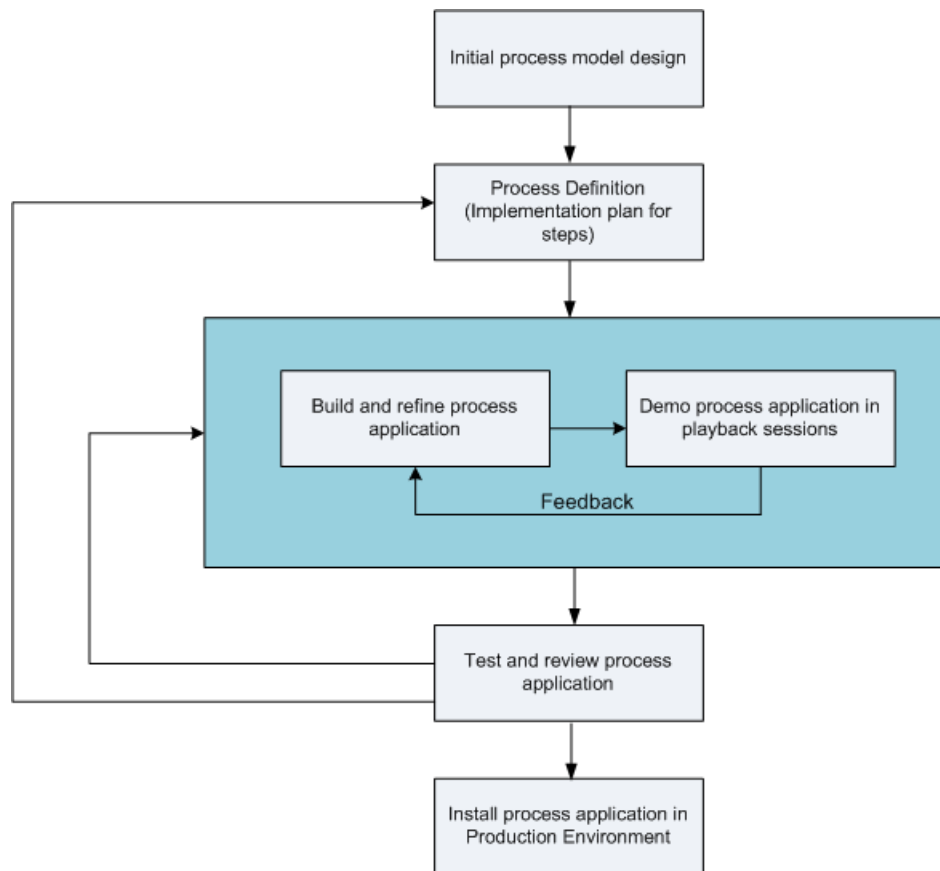
Object	Conventions
Business Process Definitions (BPDs)	<p>Maximum length of 54 alpha-numeric characters (ASCII 23 - 255) and the following special characters are allowed: , . _ - ; : !</p> <p> The following special characters are <i>not</i> allowed: " ' ` \ / ^</p>
Activities	Maximum 580 characters in length (anything over ASCII 32 is allowed)
Variable names	Should start with a lowercase letter, with subsequent words capitalized like so: myVar. Variable names cannot start with numbers or dashes and cannot include spaces. (Should follow the ECMA-262 standard; keywords are allowed.)
Tracked names (autotracked or in tracking groups)	Same conventions as variable names, with a limitation of 30 characters in length. (Cannot be SQL-92 keywords.)

Running and installing processes

The Inspector in Lombardi Authoring Environment enables you to run and debug your processes throughout their entire life cycle. The following sections provide an introduction to the types of tasks involved in running, testing, and then installing your Lombardi processes.

Overview

When developing processes in Lombardi, you should take full advantage of the iterative approach supported by the tools within the Authoring Environment. Processes evolve over time, initially from a development state, to testing, and then on to production. But, once in production, your processes will continue to evolve due to changing needs. Being prepared for the ongoing life cycle of your processes is important and will help you design effectively from the outset. The following figure illustrates an iterative approach to process development with Lombardi:



The Inspector in Lombardi Authoring Environment enables you to demonstrate your processes in playback sessions and it also enables you to run and debug your processes during development efforts. For these reasons, it is an important part of the iterative approach illustrated in the preceding figure. Additionally, the Inspector can help you resolve formal testing issues and then determine when to install processes in your production environment.

A typical Lombardi configuration includes three environments to support the development and eventual installation of your processes:

Environment	Description
Development	Build and refine your process applications in Lombardi Authoring Environment. Create your process models and implement the steps in those models using the Designer. Using the Inspector, demonstrate your development progress in playback sessions so that you can quickly evaluate and refine your prototype. Using the Process Center Console, install your process applications in Lombardi test and production environments.
Test	Using the Process Center Console, install your process applications on the Process Server in your test environment to implement formal quality assurance tests. You can use the Inspector to help verify and resolve issues.
Production	When all issues reported from formal testing are resolved, use the Process Center Console to install your process applications on the Process Server in your production environment. You can use the Inspector to investigate and resolve any issues reported in your production environment.

To learn more

See the following topics to learn more about running and installing Lombardi processes:

To learn how to...	See...
Manage process instances and debug processes. You can run and debug your processes on any the Process Center Server or connected Process Server in the Lombardi environment.	Running and debugging processes with the Inspector
Manage releases of your processes and install tested projects on your production server.	Releasing and installing processes

Running and debugging processes with the Inspector

The Inspector in Lombardi Authoring Environment is key to an iterative approach to process development. Using the Inspector, individual developers can run processes and services on the Process Center Server or remote runtime Process Servers. Plus, an entire development team can use the Inspector to demonstrate current process design and implementation in playback sessions. Playback sessions help capture important information from different stakeholders in a process, such as management, end users, and business analysts. Taking an iterative approach to process development ensures that your process applications meet the goals and needs of everyone involved.

The Inspector in Lombardi Authoring Environment includes several tools that enable you to complete tasks like the following in each of your configured environments:

Task	Description
Manage instances of processes	When you run a process, you can view all previously executed and currently running instances on the Lombardi servers in your environment. You can manage running instances by halting and then resuming them, for example. You can also manage previously executed instances by filtering or deleting specific records.
Step through and debug a process	For a selected instance, see the currently executing step and then move forward through the process, evaluating process execution step by step. A tree display of the process combined with indicators called tokens in the process diagram make it easy to understand where you are in the process. You also have the advantage of seeing the variables used in each step and their corresponding values (where applicable).

See the following topics to learn more about using the Inspector interface:

To learn how to...	See...
Manage current and previously running instances of your process on the server that you select.	Managing process instances
Step through process execution to ensure that your BPD works as expected.	Stepping through a process
Examine each underlying process or service in each step of your process execution to perform a more detailed inspection than simply stepping through your process provides.	Debugging a process
Easily find the source of errors generated when running your BPD and resolve them.	Resolving errors
Access and use each feature provided by the Inspector.	Inspector reference

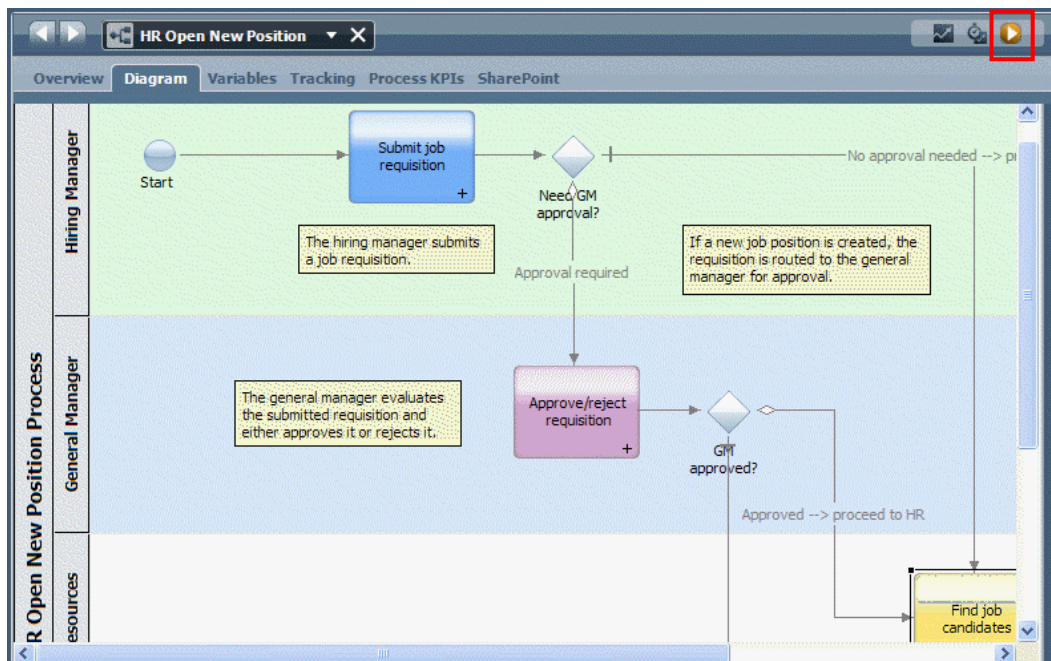
Managing process instances

The following procedure shows how to run a process and then manage running and completed instances of that process in the Inspector.



To learn more about the Inspector interface before you begin, see [Inspector reference](#).

1. Open a Business Process Definition (BPD) in the Designer in Lombardi Authoring Environment.
2. Click the Run icon as shown in the following image:



3. When Lombardi prompts you to change to the Inspector interface, click **Yes**.



Click the check box if you want Lombardi Authoring Environment to change interfaces without prompting for approval.

- In the Process Instances tab, you can see all currently active and completed process instances:

Instance Name	Snapshot	Process	Status	Due Date	Instance Id
Employee Requisition for ...	Tip	HR Open New Position	Active	Apr 1, 200...	60
Employee Requisition for ...	Tip	HR Open New Position	Completed	Apr 1, 200...	57
Employee Requisition for (...	Tip	HR Open New Position	Active	Apr 1, 200...	61
Employee Requisition for (...	Snap 1	HR Open New Position	Active	Apr 1, 200...	59
Employee Requisition for (...	Tip	HR Open New Position	Active	Apr 1, 200...	58

The highlighted instance in the preceding example is the process that you just started using the Run icon. Initially the Inspector shows the running and completed instances on the Process Center Server for all snapshots (versions) of the current BPD. For example, if a developer runs a process that he's working on in the Designer, the Process Instances tab shows all running and completed instances of the current BPD on the Process Center Server.

- (Optional) To view instances running on a different server or to view instances for a different version of the BPD, use the toolbar menus shown in the following image:

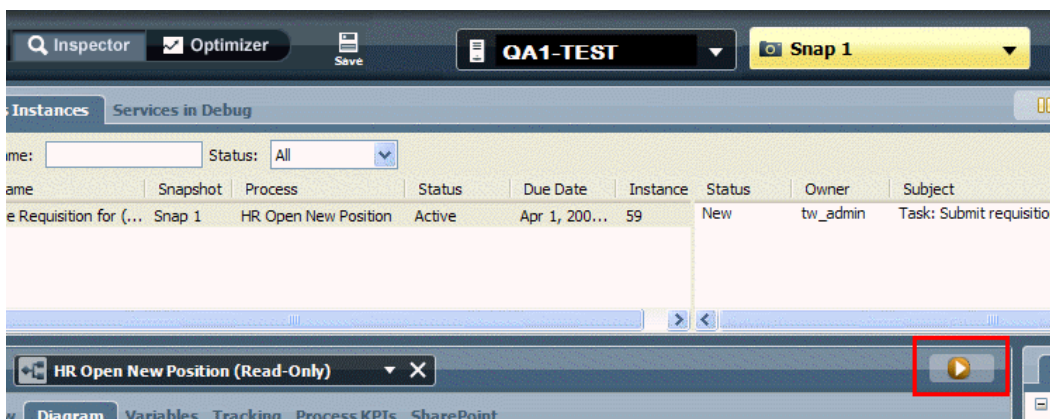


You can select a different server from the drop-down list, and you can also select from the drop-down list of snapshots.



Remote Process Servers must be connected to your Process Center to be available. See the *Lombardi Installation and Configuration Guide* appropriate for your environment to learn how to connect to remote Process Servers. To run a process on a different server using the Inspector, you must first install the process application snapshot that contains the process that you want to run as described in [Installing process applications: online Process Servers](#).

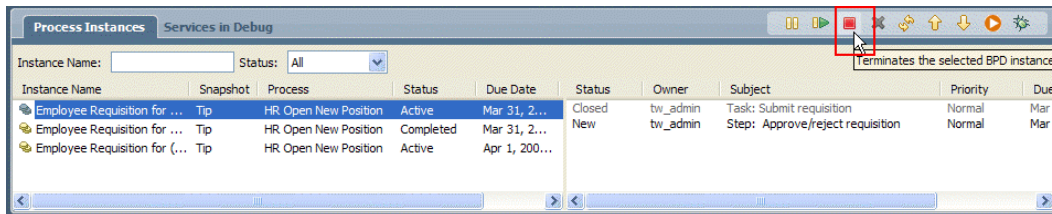
When you switch servers or snapshots, you can start the process by clicking the Run icon in the Inspector (which you can see at the bottom of the following image):





If you click the Run icon while **All versions** is selected from the list of snapshots, the Inspector runs the most recent snapshot of the BPD on the Process Center Server. For remote Process Servers, the snapshots available are limited to the snapshots that are installed on that server.

- To control instances, select an instance from the list and then click the toolbar option that you want. For example, if you want to stop an instance that you started earlier, click the instance and then click the icon to terminate the instance as shown in the following example:



You can also filter the list of instances shown by providing a name in the **Instance Name** field and using the **Status** drop-down menu. For complete information about the available options, see [Inspector reference](#).

- In the preceding image, you can see a new task in the right pane of the Process Instances tab. You can run new tasks to step through process execution as explained in the following procedure.

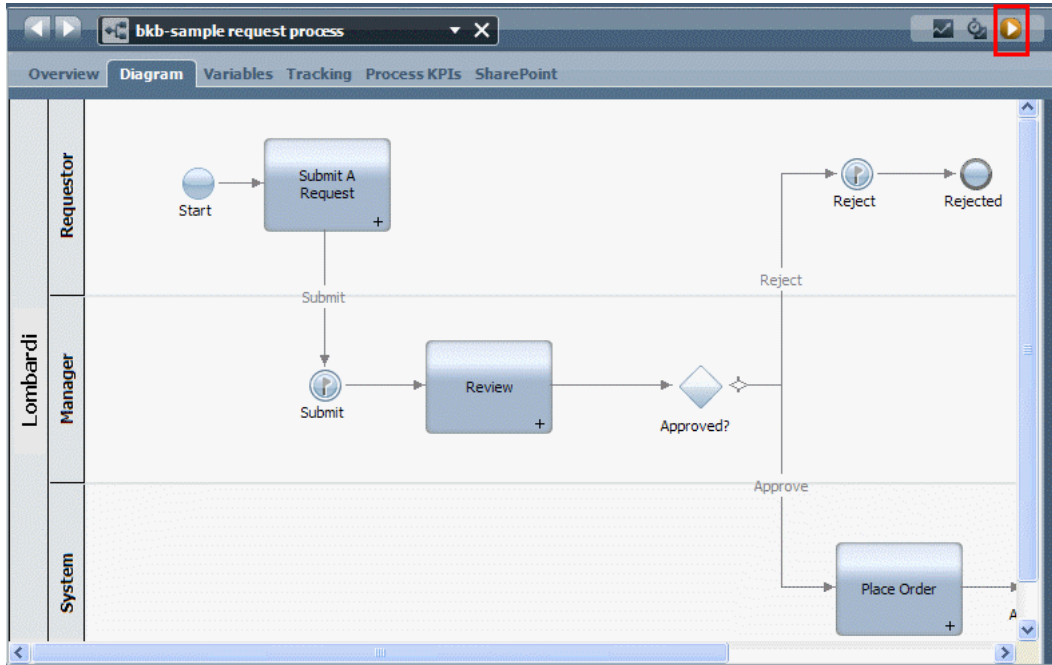
Stepping through a process

When you run a process, you can step through the execution to ensure that your BPD works as expected. You can use this functionality for team playbacks and to help debug your process.



To learn more about the Inspector interface before you begin, see [Inspector reference](#).

- Open a Business Process Definition (BPD) in the Designer in Lombardi Authoring Environment.
- Click the Run icon as shown in the following image:



3. When Lombardi prompts you to change to the Inspector interface, click **Yes**.



Click the check box if you want Lombardi Authoring Environment to change interfaces without prompting for approval.

4. (Optional) If you want to step through the current BPD on a different server or if you want to step through a different version of the current BPD, use the toolbar menus shown in the following image:

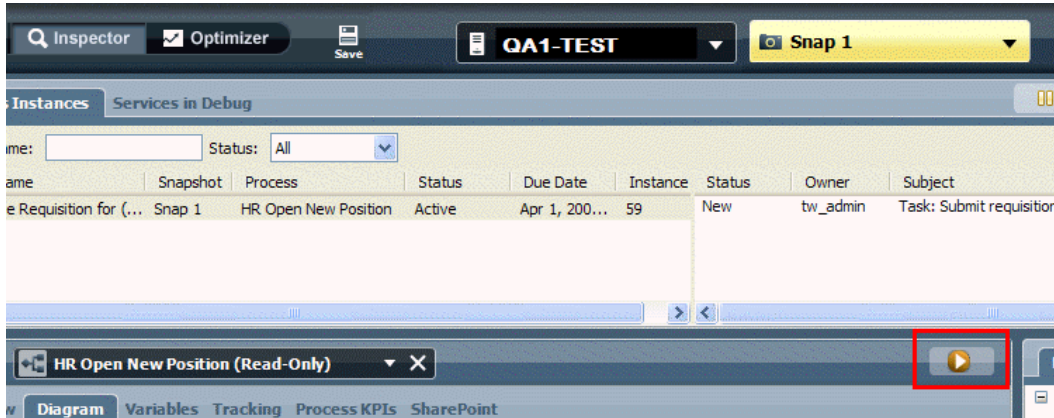


You can select a different server from the drop-down list, and you can also select from the list of snapshots.



Remote Process Servers must be connected to your Process Center to be available. See the *Lombardi Installation and Configuration Guide* appropriate for your environment to learn how to connect to remote Process Servers. To run a process on a different server using the Inspector, you must first install the process application snapshot that contains the process that you want to run as described in [Installing process applications: online Process Servers](#).

When you switch servers or snapshots, you need to start the process again by clicking the Run icon in the Inspector (which you can see at the bottom of the following image):

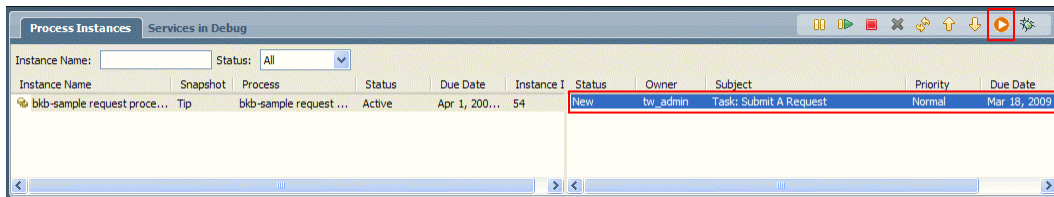


If you click the Run icon while **All versions** is selected from the list of snapshots, the Inspector runs the most recent snapshot of the BPD on the Process Center Server. For remote Process Servers, the snapshots available are limited to the snapshots that are installed on that server.

5. In the Process Instances tab, click the new or received task and then click the Run task icon as shown in the following image.

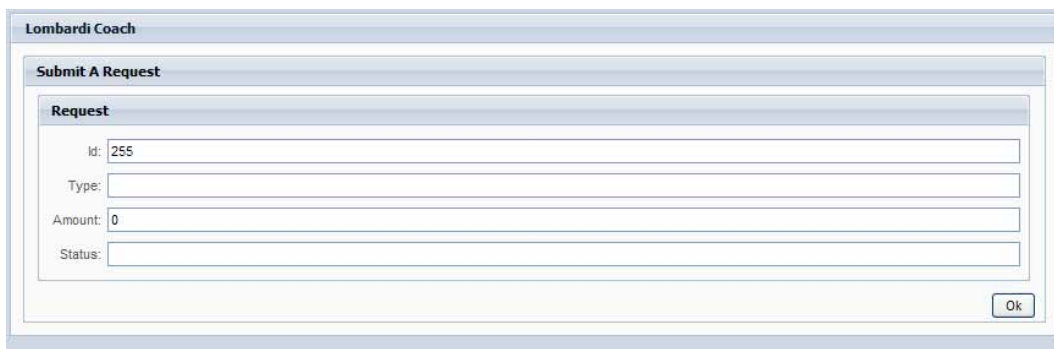


If a task is not displayed, click the process instance that you just started to see its current task.



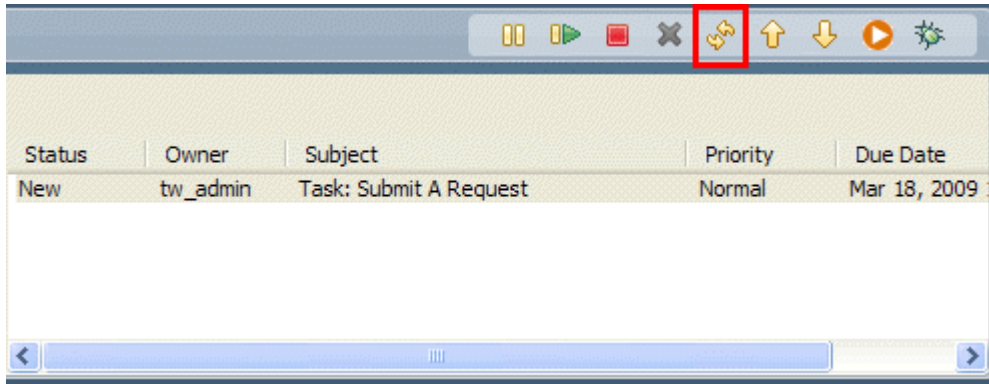
In some cases, you may need to select a user account or provide a password for a specific user account in order to run a task. This is controlled by lane assignments and routing for activities. See [Assigning participant groups to lanes](#) and [Routing activities](#) for more information.

6. In this sample BPD, the coach for the task called Submit a Request opens in a browser as shown in the following example:

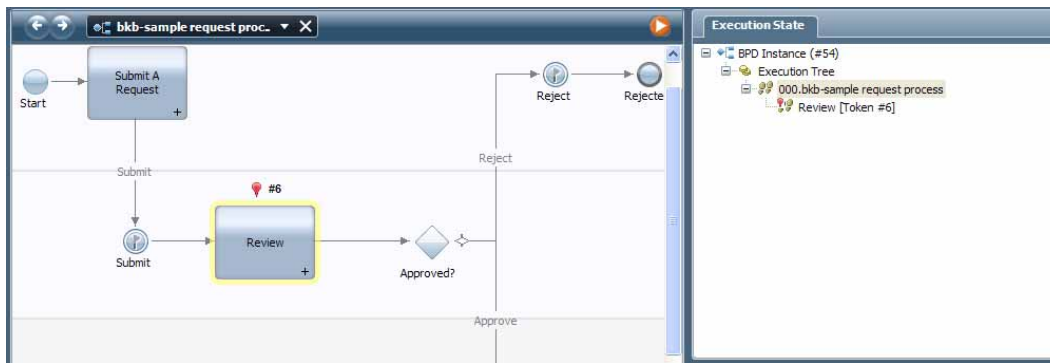


Filling in the fields and clicking the Ok button causes the BPD to move to the next step.

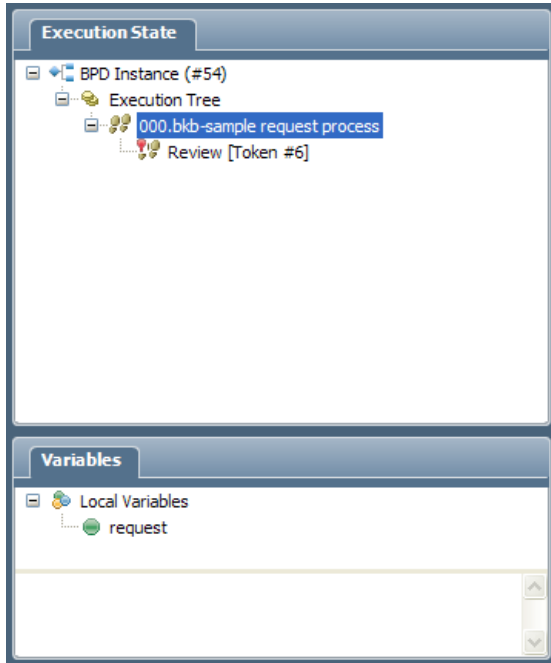
- Click the Refresh icon in the toolbar as shown in the following image:



The Inspector shows the progress by moving the token to the next step (the Review task) in both the BPD diagram and the tree view as shown in the following example:

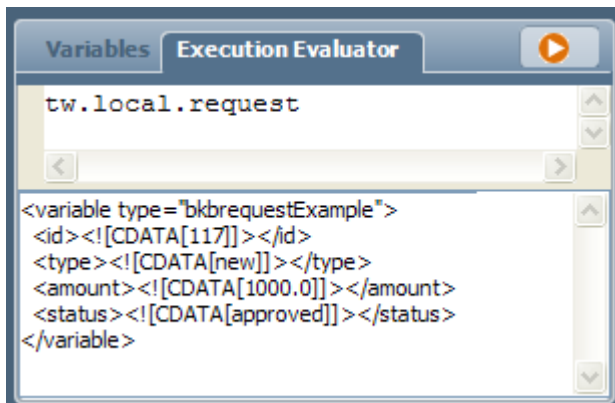


- To see the variables passed from step to step, click the process node in the tree view. In the following example, you can see that the Inspector shows the `request` variable in the Variables tab:



9. Right-click the `request` variable and select **Show in Execution Evaluator**.

The Inspector opens the Execution Evaluator tab and shows the values for the parameters within the `request` variable:

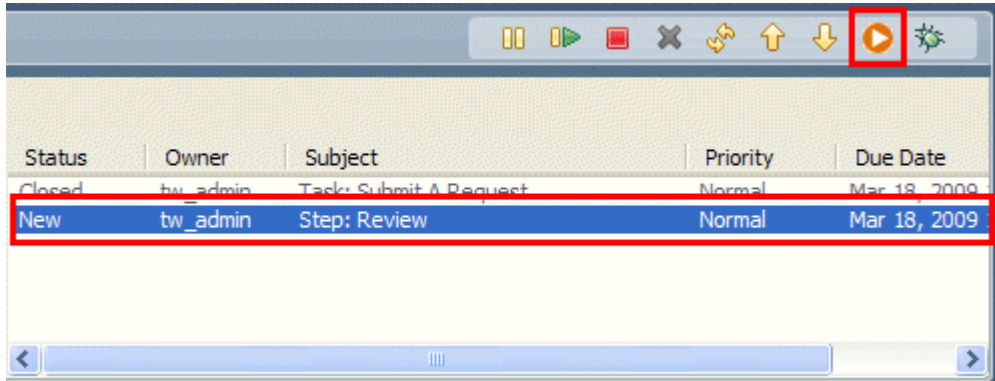


The Execution Evaluator enables you to inspect the variable values as they change through the flow of the BPD.



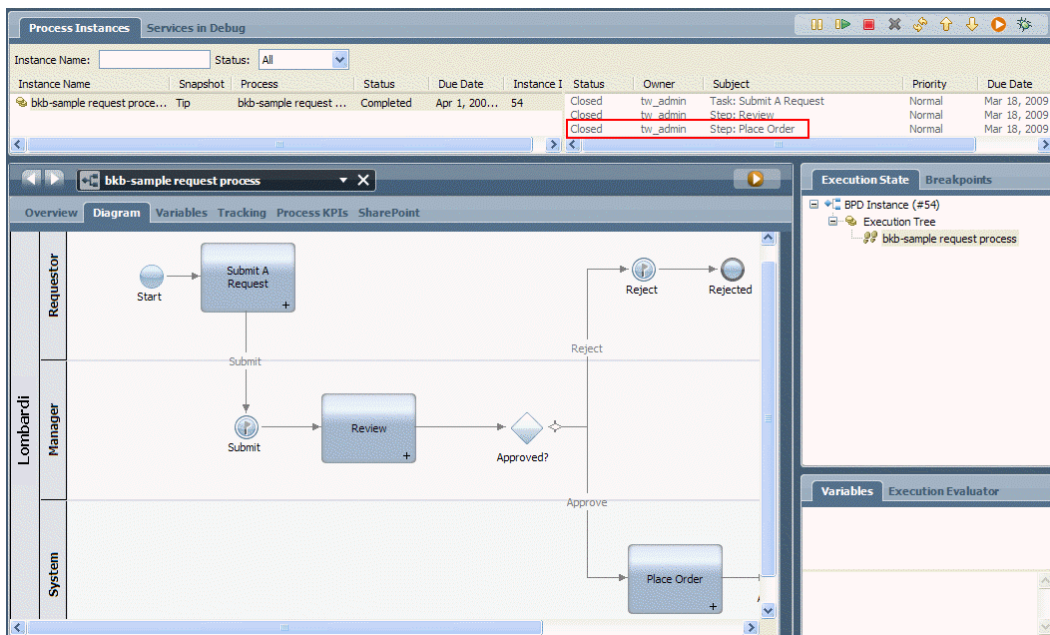
You can also manipulate variables in the Execution Evaluator using JavaScript expressions to validate your process implementation. To do so, enter the JavaScript expression and click the Run icon at the top of the Evaluator. The results are displayed in the bottom pane of the tab.

10. In the Process Instances tab, click the task for the Review step and then click the Run task icon as shown in the following image:



In this sample BPD, the coach for the Review task opens in a browser and clicking Ok for the submitted request causes the BPD to move to the next step.

11. Click the Refresh icon in the toolbar.
12. Now we can see that the BPD is complete because the final step, Place Order, has a status of Closed and there are no active tokens in the diagram or tree view:



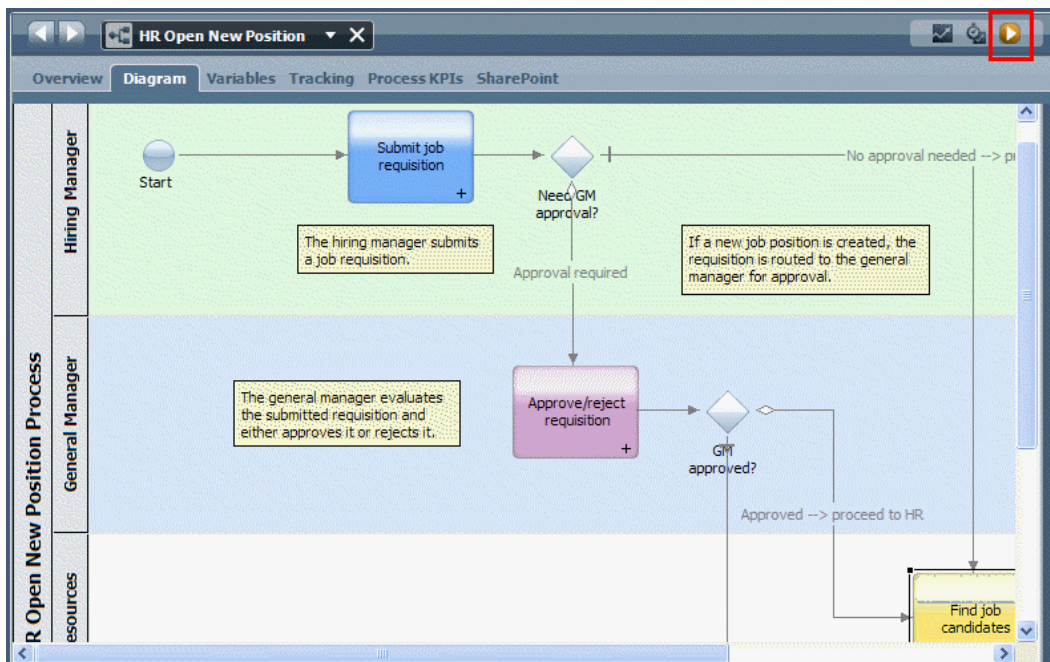
Debugging a process

The debugging feature in the Inspector enables you to examine each underlying process or service in each step of your process execution, providing more detailed inspection than simply stepping through your process provides. The Inspector executes a debugging session in a browser window. As you step through an underlying process or service in the debug session in your browser, the Inspector interface shows the same progress in its diagram view and tree view.



To learn more about the Inspector interface before you begin, see [Inspector reference](#).

1. Open a Business Process Definition (BPD) in the Designer in Lombardi Authoring Environment.
2. Click the Run icon as shown in the following image:



3. When Lombardi prompts you to change to the Inspector interface, click **Yes**.



Click the check box if you want Lombardi Authoring Environment to change interfaces without prompting for approval.

4. (Optional) If you want to debug the current BPD on a different server or if you want to debug a different version of the current BPD, use the toolbar menus shown in the following image:

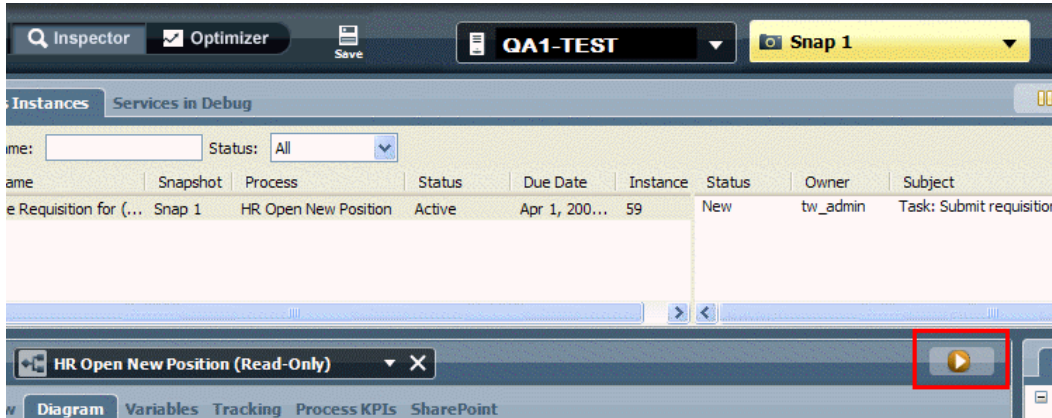


You can select a different server from the drop-down list, and you can also select from the list of snapshots.



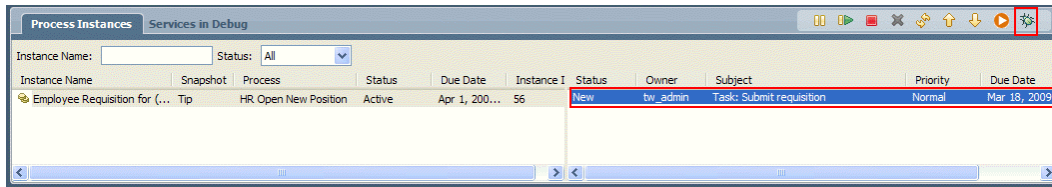
Remote Process Servers must be connected to your Process Center to be available. See the *Lombardi Installation and Configuration Guide* appropriate for your environment to learn how to connect to remote Process Servers. To run a process on a different server using the Inspector, you must first install the process application snapshot that contains the process that you want to run as described in [Installing process applications: online Process Servers](#).

When you switch servers or snapshots, you need to start the process again by clicking the Run icon in the Inspector (which you can see at the bottom of the following image):

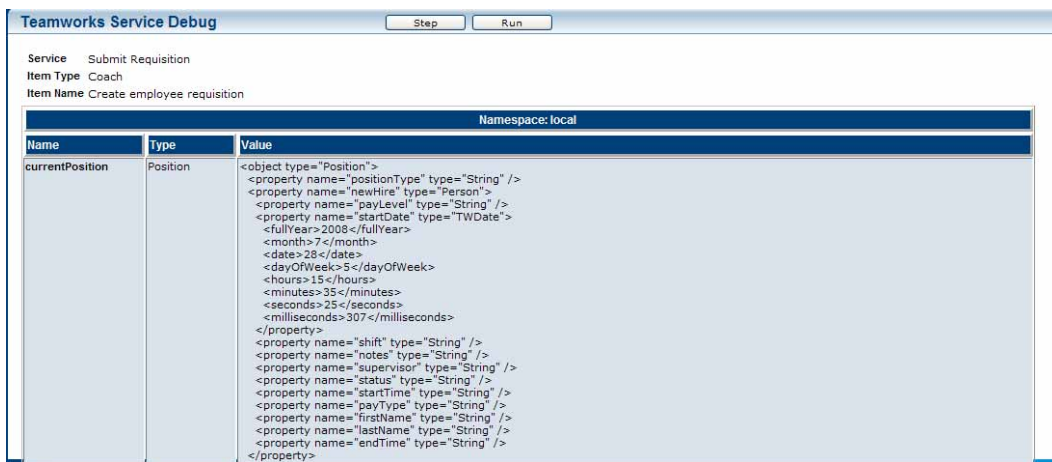


If you click the Run icon while **All versions** is selected from the list of snapshots, the Inspector runs the most recent snapshot of the BPD on the Process Center Server. For remote Process Servers, the snapshots available are limited to the snapshots that are installed on that server.

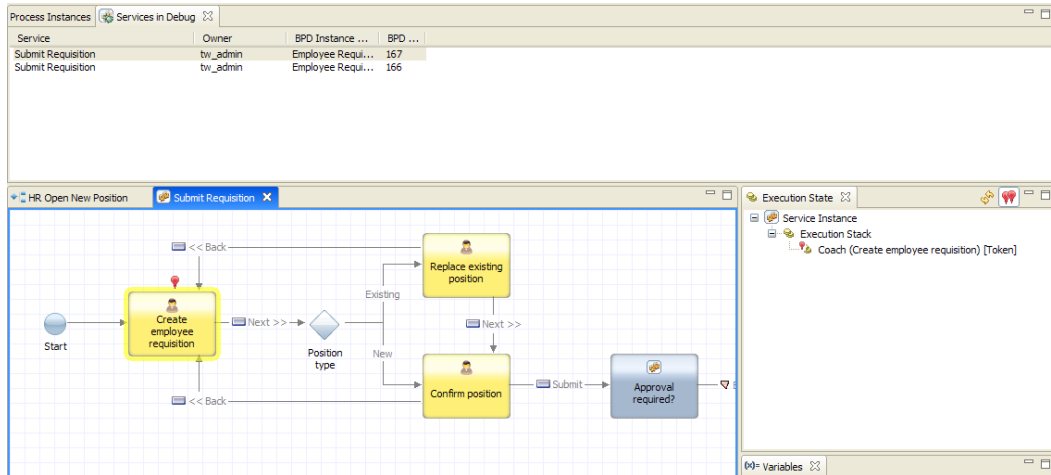
5. In the Process Instances tab, click the new task and then click the Debug task icon as shown in the following image:



The Inspector opens a debugging session in a browser window as shown in the following example:



At the same time, the Inspector opens the currently executing service in the **Services in Debug** tab and shows progress through the service, using tokens in the diagram and tree view. In the following example, you can see the token is on the same item (Create employee requisition) shown in the debugger session:



- Click the **Step** button in the debugger session in your browser to proceed. In this sample BPD, the coach for the current service opens in the browser window:

Filling in the fields and clicking the Next button causes the debugger to move to the next step.

- Click the **Step** button in the debugger session to continue debugging the current service.
- To continue through the rest of your BPD, click the Process Instances tab in the Inspector and then repeat the actions from Step 5.
- In the debugger session in your browser, you can see data that you enter into any displayed coaches as well as the values that cause the underlying logic in the services and BPD to proceed along the available paths. This insight can be extremely helpful when issues are identified and you need to find the point at which a process instance is not functioning as expected.



For BPDs or services that do not require user input, you can click the **Run** button in the debugger to execute all code and logic and then view the end values.

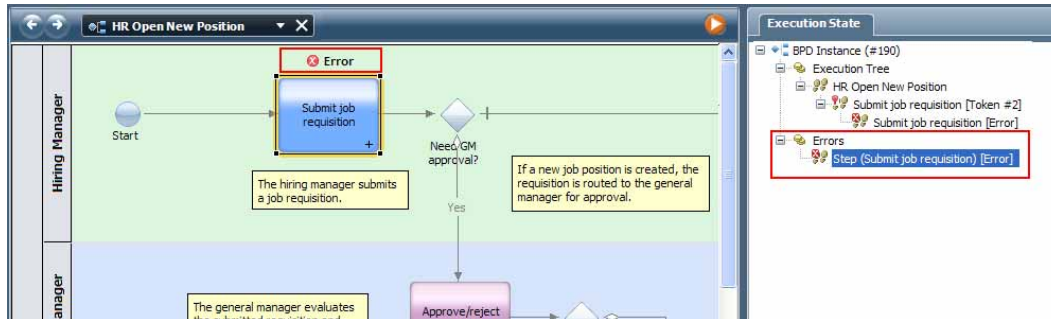
Resolving errors

When you run a process and an exception occurs in the instance, the Inspector clearly identifies the error in the diagram and tree view. The Inspector also:

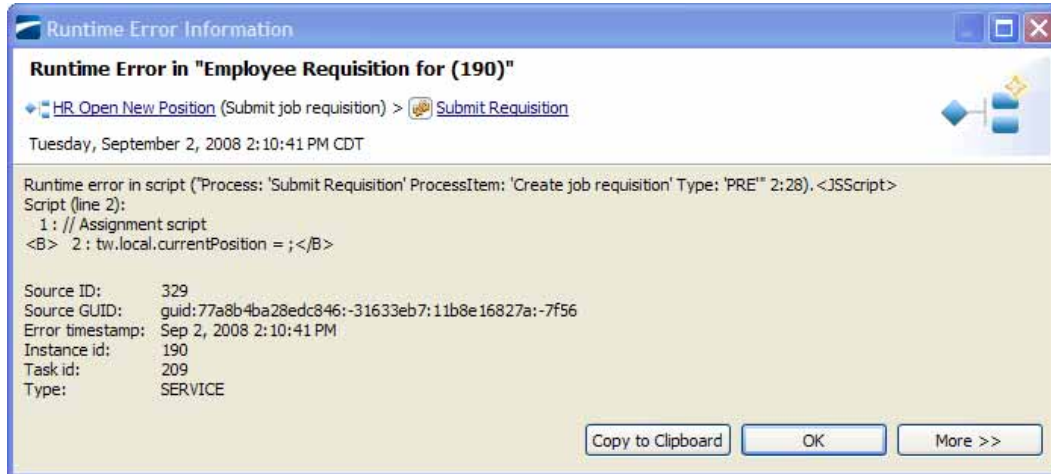
- Tells you exactly where the error happened
- Links directly to the source of the problem

The following example shows how the Inspector identifies an error in a running instance and how it helps you resolve the error:

1. When you run a BPD that does not execute properly, the Inspector displays the error as shown in the following image:

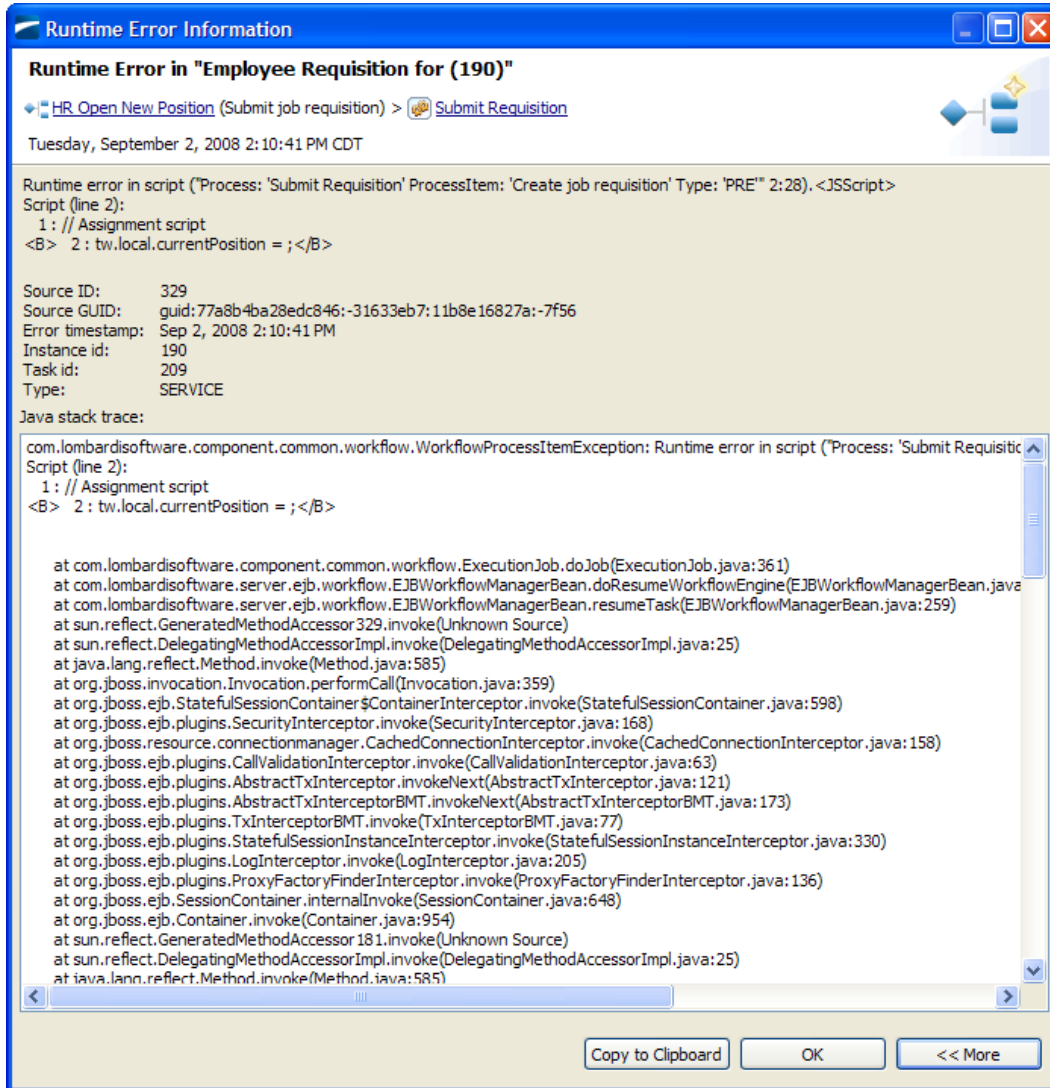


2. Click the error shown in the tree view to open the Runtime Error Information dialog:



In the preceding example, you can see the Runtime Error Information dialog tells you exactly where the error happened and it also provides a link to the service in which the error occurred so you can go directly to the source. It is within the Submit Requisition service that the variable assignment is missing for `tw.local.currentPosition`.

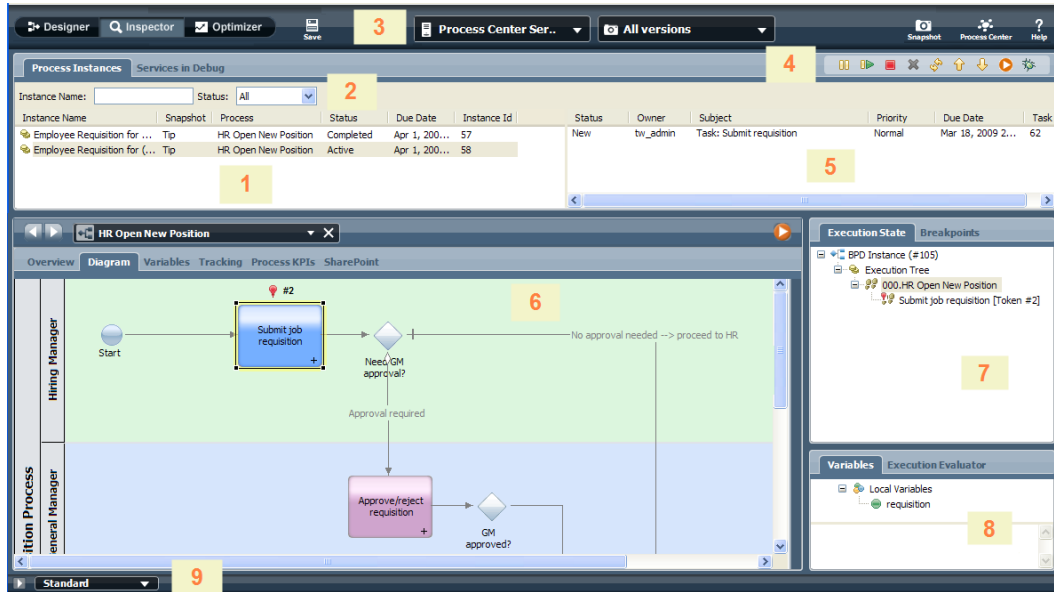
3. Click the **More** button to show additional details about the error, such as stack trace details as shown in the following image:



You can also use the **Copy to Clipboard** button if you want to paste the contents of the dialog to a text file or support ticket. The Inspector copies all information to the clipboard, including stack traces.

Inspector reference

The following image shows the Inspector interface and each major functional area:











The following table describes each numbered area in the preceding image of the Inspector interface in Lombardi Authoring Environment:

1	Shows the currently active and previously executed process instances on the Process Center Server or connected Process Server in the Process Instances tab. The highlighted instance is the currently selected instance, which means any action you perform or any data shown in the other areas of the Inspector apply to this instance. (The Services in Debug tab becomes active only if you select the Debug icon for a particular task.)
2	Use the Instance Name field and the Status drop-down menu to control the list of instances displayed in the Process Instances tab. For example, if you want to see only active process instances that include the word employee in their name, type <code>employee</code> in the Instance Name field and select Active from the Status drop-down menu.
3	<p>Use the drop-down lists to choose a different server on which to run the current BPD. You can also choose a different snapshot if you want to run a different version of the BPD. The Process Instances tab includes a Snapshot column so that you know which version of a process the currently displayed instances are running. For example, in the preceding image, the Snapshot column displays <code>Tip</code> to indicate that you are running the version of the process currently under development in the Designer.</p> <p>Remote Process Servers must be connected to your Process Center to be available. See the <i>Lombardi Runtime Environment Installation and Configuration Guide</i> to learn how to connect runtime Process Servers to the Process Center. To run a process on a different server using the Inspector, you must first install the process application snapshot that contains the process that you want to run as described in Installing process applications: online Process Servers.</p> <div data-bbox="467 1583 542 1654" style="float: left; margin-right: 10px;"> </div> <p>When you change servers or versions in the Inspector, you have to start the BPD again by choosing the run icon at the top-right of the BPD diagram.</p>
4	Use the icons in the toolbar to manage process instances, run tasks, or debug services. See Inspector toolbar options for more information.
5	Shows the current task for the selected process instance. In the preceding example, the current task is the first task in the BPD called Submit requisition. You can click on the task to select it and then use the toolbar icon to run the task so that you can step through the entire BPD.

6	Shows the BPD diagram for the selected instance and highlights the current task so you know where you are in the execution of the process for this instance. In the preceding example, the red token above and the yellow halo around the Submit job requisition task indicate the active step. If you want to view other information about the BPD for the selected instance, you can click the other available tabs such as Overview and Variables.
7	Shows a tree view of the execution progress for the selected instance. In the preceding example, you can see the first step in the instance (the start of the process) and you can see the active second step, indicated by the red token. The tree view continues to expand if you decide to run the task and step through the entire process in the Inspector.
8	Shows the variables used in the current step. In the preceding example, we can see that the requisition variable is used in the active step. You can select a variable, right-click, and then select Show In Execution Evaluator to view and manipulate the variable values.
9	Use the drop-down menu to change the Inspector interface display. When you open the Inspector interface, you see the Standard display which is described in the preceding rows. You can choose to show less information (Simple) or more (Edit & Debug).

Inspector toolbar options

When viewing instances of processes in the Process Instances tab of the Inspector, the following toolbar icons are available to help you manage those instances.

Toolbar icon	Function
	Pauses the selected process instance.
	Resumes a suspended process instance.
	Stops a running process instance.
	Removes any record of the selected process instance.
	Refreshes the current list of process instances based on the filter you have selected and the most recent data from the Process Server. When stepping through a process instance, you need to click Refresh after completing a step so that the diagram view and tree view properly display your progress.
	Pages through the BPD instances when more than 100 instances are displayed in the Process Instances tab.
	Runs the selected task. A Web browser opens to display coach(es), while the Inspector displays red tokens both in the BPD diagram and the tree view of the execution state to show the step that is executing in the active process instance.
	Opens a debug session in your default Web browser for the selected task. The debug feature enables you to step through each service in each stage of the process execution to view the business data that is passed in and out. As you step through a service in the debug session in your browser, the Inspector interface shows the same progress in the currently executing service in the diagram view and tree view.

Understanding tokens

In the Inspector interface, tokens indicate where the run-time instance is currently executing—both in the diagram of the BPD or service and in the tree view of the instance execution. The following general rules apply to tokens:

- If a token is on a step, that step is in an active state. Until the step has been completed, the process instance cannot proceed.

- If a token is on a conditional sequence line, the target step is enabled—it can receive the token—because the condition on the sequence line was met. Conditions are evaluated after the step has been completed but before the process enters the outgoing sequence line.
- A process instance can generate several tokens. For example, a split can cause two tokens to be generated. These tokens are merged into a single token when the process reaches a join gateway.

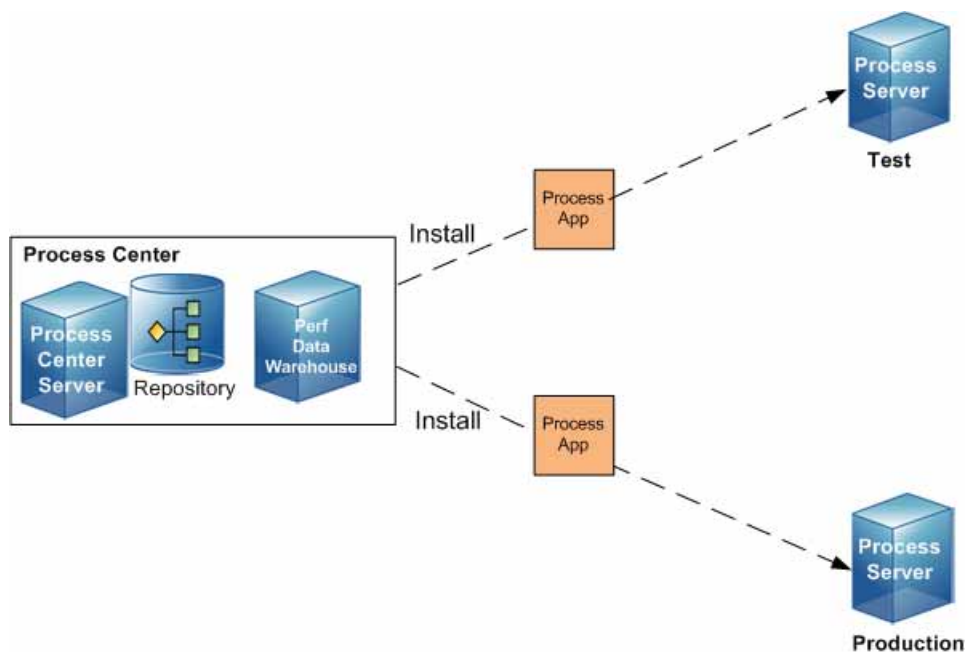
Releasing and installing processes

To ensure the process applications that you implement meet the quality standards of your organization, you should plan to invest time and resources to develop a release and installment strategy for those applications. When you have identified the goals and requirements for release and installment of new and updated process applications, you can use Lombardi to automate the processes required to approve and launch the programs. Doing so will ensure that your efforts to maintain and upgrade your automated processes lead to expected results.

You can install snapshots of process applications to connected Process Servers in your Lombardi configuration. When Lombardi Process Center performs an installation, it moves the library items (including toolkit dependencies) in the selected snapshot from the Process Center repository to the Process Server that you choose. Ordinarily, you have connections to one or more servers in your Lombardi environment as shown in the following figure:



To install to Process Servers in Production environments, you must have administrative access to the process application. For more information about required access for installation, see [Installing process applications: online Process Servers](#).



If you need to install to an offline server, you can create an installation package for a particular snapshot on the Process Center Server, transfer the package to the offline Process Server, and then run the installation package as described in [Installing process applications: offline Process Servers](#). The installation package installs all library items (including toolkit dependencies) from the selected snapshot to the offline Process Server.

Developing a release and installment strategy

In most cases, you should implement a formal review before a process application is installed in a test or production environment. For example, you may want to route a process to several different managers across different reporting structures in your organization. Only after each manager signs off on the new or updated process can it be installed in your production environment and rolled out to end users. You could easily create and implement the steps involved in such a review in Lombardi to ensure that all corporate guidelines have been satisfied and that you have the required signatures. The final step in the review could be notification to the IT team that the approved process application is ready for installation and roll out.

Lombardi also enables you to customize the installation of your process applications. When you create a process application, Lombardi automatically creates an installation service for that process application. You can add calls and scripts to the installation service to perform specific functions when a process application is installed on a server in another environment. For example, you may want to create database schemas or set environment variables to support running instances of your process.

Building installation services

When you create a process application, Lombardi automatically creates an installation service. You are not required to add components to the installation service to successfully install your process application, however, the service should be able to handle any advanced requirements in your target environment as outlined in this section.



To build the installation service, you must have write access to the process application that you want to install. See [Managing access to the Process Center repository](#) for more information about granting access to process applications.

Your installation service should handle the following types of configuration or other requirements on the target Process Server:

- Create or update database tables
- Update necessary environment variables
- Determine which snapshots are already installed
- Migrate individual process instances
- Create custom time schedules

For example, you can build an installation service to create tables on the target Process Server to hold data such as the options for drop-down menus that exist in your process. And, when you need to add to or change those menu options, you can alter the installation service so that those database updates are handled automatically during installation.

To add functionality to the installation service:

1. Open the process application that contains the installation service in the Designer view.

(See [Starting Lombardi Authoring Environment](#) to learn how to open process applications in the Designer view.)

2. Click the **Setup** category and double-click the Installation Service to open it.

- The diagram of the selected service opens so that you can add the service calls and scripts that you need for your particular installation.

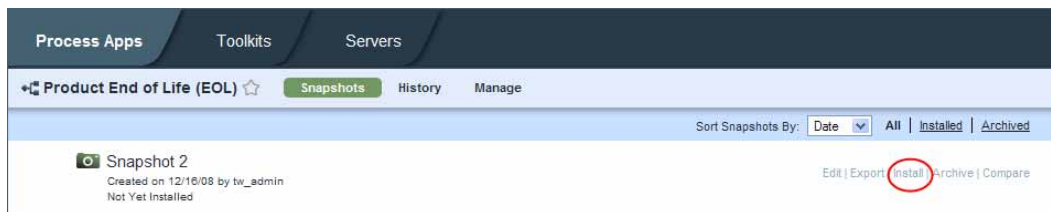
Installing process applications: online Process Servers

The following procedure describes how to install process applications on Process Servers that are connected to your Process Center Server. To learn how to install applications on Process Servers that are not connected to the Process Center Server, see [Installing process applications: offline Process Servers](#).

Before installing, be sure that you have:

- Started Lombardi Authoring Environment and opened the Process Center Console as explained in [Starting Lombardi Authoring Environment](#).
- Created a snapshot of the process application to be installed. See [Managing snapshots](#) to learn how to create snapshots.
- Access to the process application as follows: administrative access to install to Process Servers in Production environments; write access to install to any non-production Process Server; read access to install to Process Servers in Development environments. For more information, see [Managing access to process applications and toolkits](#).
- Connected to the server on which you intend to install the snapshot. See your *Lombardi Installation and Configuration Guide* to learn how to connect the Process Servers in your test and production environments to your Process Center.
- Reviewed the steps that Lombardi follows when performing an installation as described in [Troubleshooting installations](#).

- Select the **Process Apps** tab, and then click the process application that you want to install.
- Find the snapshot that you want to install and click the **Install** option for that snapshot as shown in the following image:



- Select the Process Server on which you want to install and click the **Install** button. (The Process Center Console displays all connected Process Servers.)

Lombardi checks to see if the target server is currently running instances of the BPDs included in the deployed snapshot. If running instances of the BPD(s) you are installing are detected on the target server, Lombardi issues a prompt to determine whether to migrate those running instances to the new snapshot. See [Migrating instances](#) to learn more about how to handle running process instances.

Any toolkits that the deployed process application depend upon (that are not already installed on the selected Process Server) are also installed, ensuring that all library items required to run the application are available.



If you experience problems with your installation, you can check the [Lombardi_home]\process-server\logs\tw-expimp.log file for progress messages and the [Lombardi_home]\process-server\logs\tw-error.log file for errors. See [Troubleshooting installations](#) for more information about where issues can occur.

4. If you need to adjust process application settings for the target environment, for example, you may need to set up environment variables or user access, see [Configuring installed snapshots](#) for more information.

Installing process applications: offline Process Servers

The following procedure describes how to install process applications on Process Servers that are *not* connected to your Process Center Server. To learn how to install applications on Process Servers that are connected to the Process Center Server, see [Installing process applications: online Process Servers](#).

Before installing, be sure that you have:

- Started Lombardi Authoring Environment and opened the Process Center Console as explained in [Starting Lombardi Authoring Environment](#).
- Created a snapshot of the process application to be installed. See [Managing snapshots](#) to learn how to create snapshots.
- Added an offline server as described in [Adding offline servers to the Process Center Console](#).
- Reviewed the steps that Lombardi follows when performing an installation as described in [Troubleshooting installations](#).

The following steps describe how to create an installation package that you can transfer to the offline server and then run from the command line.

1. Select the **Process Apps** tab, and then click the process application that you want to install.
2. Find the snapshot that you want to install and click the **Install** option for that snapshot as shown in the following image:



3. Select the offline Process Server for which you want to create an installation package and click the **Create installation package** button.



Creating an installation package requires the following type of access to the process application: administrative access to create an installation package for Process Servers in Production environments; write access to create an installation package for any non-production Process Server; read access to create an installation package for Process Servers in Development environments. For more information, see [Managing access to process applications and toolkits](#).

If you have already created installation packages for one or more snapshots of the current process application (for the currently selected Process Server), click the **Next** button.

Lombardi prompts you to determine how to handle running instances from those snapshots. Select the option that you want using the drop-down list. See [Migrating instances](#) for more information about each option.



If there are no running instances on the selected Process Server, Lombardi ignores the selected migration option when you install.

Click the **Create installation package** option.

Installation packages that you create are available on the Process Center Server as long the selected offline server exists. If you remove the offline server, the installation packages for that server are also deleted.

- Use the command-line utility `retrieveProcessAppPackage.cmd (.sh)` on the Process Center Server to retrieve the installation package. The command-line utility is available in the `[Lombardi_home]/tools/process-installer` directory.

At the command prompt, specify:

```
retrieveProcessAppPackage.cmd [process_app_acronym] [snapshot_name]
[offline_server_name] [install_package_name]
```

Where:

<code>process_app_acronym</code>	Is the acronym assigned to the process application for which you created the installation package.
<code>snapshot_name</code>	Is the name assigned to the snapshot for which you created the installation package. Put the name in quotation marks if it includes spaces.
<code>offline_server_name</code>	Is the name of the offline server that you selected in step 3. Put the name in quotation marks if it includes spaces.
<code>install_package_name</code>	Is the name that you want to assign to the file that contains the installation package. Put the name in quotation marks if it includes spaces.



On UNIX® systems, you need to put a backslash before each space in a name and put the name in quotation marks as shown in the following example: `retrieveProcessAppPackage.sh TESTDPL "Test\ Snapshot" "Dev\ Offline" test_package_file.zip`

- Transfer the installation package to the offline Process Server using ftp or a similar utility.
- Run the installation package on the offline Process Server using the `installProcessAppPackage.cmd (.sh)` utility. The command-line utility is available in the `[Lombardi_home]/tools/process-installer` directory.

At the command prompt, specify:

```
installProcessAppPackage.cmd [install_package_name]
```

Where `install_package_name` is the name that you assigned to the package in step 5.




If you want to customize your offline installation, see [Customizing process application installations on offline Process Servers](#).

If you experience problems with your installation, you can check the `[Lombardi_home]/tools/process-installer/logs/process-installer.log` file for errors. See [Troubleshooting installations](#) for more information about where issues can occur.

Migrating instances

When installing to a connected (online) server, if running instances of the BPD(s) you are installing are detected, Lombardi prompts you to determine which of the following actions to take. For offline servers, you can choose the option you want when creating an installation package.

Online migration option	Offline migration option	Description
Leave running instances on current version (snapshot)	Leave	The instances currently running continue to completion using the previously installed version (snapshot).
Migrate running instances to new version (snapshot)	Migrate	Currently running instances are migrated to the new version you are installing. Wherever the running instances are in the flow of the process, the new version is implemented for the next item or step.
Delete running instances of current version (snapshot)	Delete	The instances currently running are immediately stopped and do not continue to completion. All records of the running instances are removed from the Process Server.  This option is not available for Process Servers in production environments.



If you migrate currently running instances to a new version, problems can occur if the new version removes steps or other components from the BPD. When running instances have tokens on BPD or service-level components that have been removed in the new version, migration may fail. Tokens indicate where a run-time instance is currently executing and a token can be present on an activity, a conditional sequence line, a coach, a service call, and numerous other components. See [Understanding tokens](#) for more information.

To learn more about migrating instances, see [Data migration rules](#).

Completing post-installation tasks

Depending on your environment and the installed process application, there are several tasks that should be considered and possibly completed immediately after installation of your process application. The tasks to consider include the following:

Task	Description	See...
Set environment variables	In some cases, the correct value for a particular environment (such as test or production) may not be known during process design. In those cases, you need to provide the value after installing the process application in the new environment.	Configuring runtime environment variables

Task	Description	See...
Establish runtime participant groups	After a process application is installed on a Process Server in a new environment (such as test or production), you may need to add or remove users in the participant groups for that application. For example, users that exist in the test environment may not have been available in the development environment.	Configuring runtime participant groups
Control exposed processes and services	After a process application is installed on a Process Server in a new environment (such as test or production), you may need to disable a particular exposed process or service within that application.	Configuring exposed processes and services



The installation service for your process application can be customized to handle these types of tasks. See [Building installation services](#) for more information.

Troubleshooting installations

Lombardi completes the following steps in the order shown when performing an installation of a process application snapshot. Understanding the steps in this process is necessary to appropriately troubleshoot installation issues.



The following steps take place on the target Process Server. For example, if you are installing a process application snapshot in your production environment, the following steps take place on the Process Server in that environment.

The target Process Server...	Description
1. Installs the necessary library items and assets for the process application and referenced toolkits.	Lombardi installs only those referenced toolkits that are not already installed on the target server. Default values for environment variables and exposed process values (EPVs) are set and other design-time versioned assets (such as Portal searches) are created.
2. Sends tracking definitions to the Performance Data Warehouse.	The Process Server updates the Performance Data Warehouse with any new or changed tracking definitions.
3. Executes the installation service for each toolkit.	The installation service for each referenced toolkit must be executed before the installation service for the referring toolkit.
4. Executes the installation service for the process application.	The installation service for the process application is the final installation service executed.
5. Migrates data and process instances (if running instances of the BPDs you are installing are discovered).	The Process Server migrates data according to the rules outlined in Data migration rules . The specific actions of this step depend upon the migration option that you choose. The migration options are described in Migrating instances .
6. Sends an installation complete message to the Process Center. (Connected servers only)	The user who initiated the installation can see that the installation completed in the Process Center Console.

When you experience problems with your snapshot installation, you can check the [Lombardi_home] \process-server\logs\tw-expimp.log file for progress messages and the [Lombardi_home] \process-server\logs\tw-error.log file for errors. These log files should help diagnose the problem.

The following list describes the potential issues that can occur during the installation steps:

- If installation of library items and assets generates an exception (step 1), the installation fails at this point and no other steps are taken.
- Failures to send tracking definitions to the Performance Data Warehouse (step 2), do not cause the installation to fail. You can send definitions when the installation completes as described in [Configuring installed snapshots](#).
- If an installation service generates an exception (step 3 and 4), the installation fails at this point and no other steps are taken. You should build your installation services to capture exceptions and roll back any changes made before the exception is generated. If your installation services do not handle exceptions, you may need to manually roll back changes before attempting to re-install. For example, if all toolkit installation services complete (step 3) and then the installation service for the process application fails halfway through its execution (step 4), you may need to roll back changes resulting from partial completion of step 4, but the toolkit installations (step 3) will be complete and will not need to be executed again.
- Failures to migrate data or instances (step 5) do not cause the installation to fail. You can view the `tw-error.log` file and you can view validation errors as explained in [Viewing and correcting validation errors](#) to help rectify any issues.

Data migration rules

If running instances of the BPDs that you are installing are discovered on the target Process Server, the target server migrates data as described in this section. See [Migrating instances](#) to understand the migration options available when running instances are discovered.

Regardless of the migration option you choose, the Process Server copies environment variables from the installed snapshot(s). If environment variable values have been changed from the defaults, the values most recently set are the ones used. In the case where an installation service sets the values, those values are considered the most recent and are the values used.

If you choose to migrate running process instances to a new version (snapshot), the Process Server performs the following actions:

- Copies exposed process values (EPVs) from the installed snapshot(s). If EPVs have been changed from the defaults, the values most recently set are the values used. In the case where an installation service sets the values, those values are considered the most recent and are the values used.



EPVs in referenced toolkits are copied only if the referenced toolkit is not already installed on the target server.

- Copies participant groups that map to a list of users. The Process Server copies users from the snapshot with the most recent date.



Users are copied only if participant groups are empty. This ensures that users added by an installation service prior to migration are not overwritten.

- Moves the Default designation from the snapshot of the running instances to the newly installed snapshot. This action takes place only if the snapshot of the running instances has previously been designated the Default snapshot. (See [Configuring installed snapshots](#) for more information about the Default designation for snapshots.)

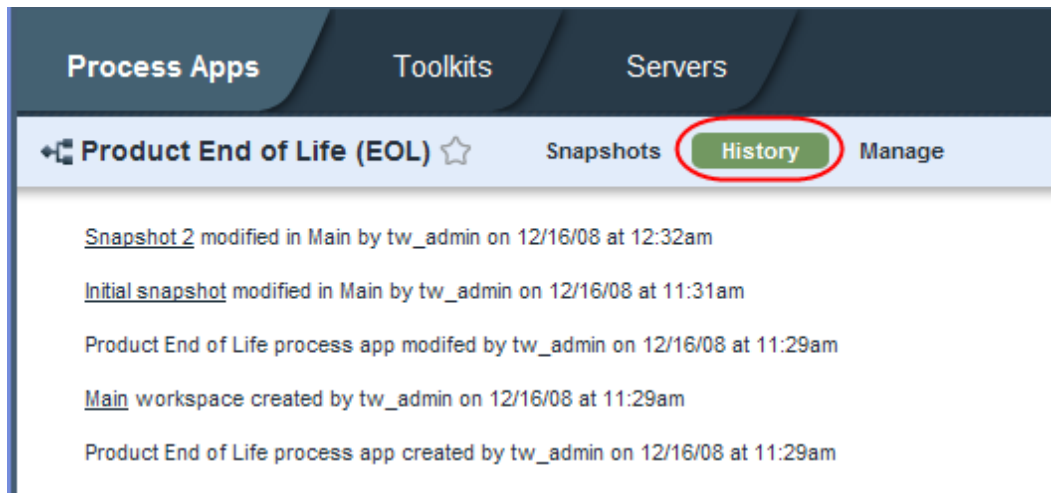
Viewing and correcting validation errors

During migration of data and instances, Lombardi performs validation to check for issues with the following:

- Variable mismatches
- Performance data mismatches
- Security changes
- Token management

When errors are discovered, the Process Server generates a message. If you are installing to a connected Process Server, you can view these messages in the Process Center Console as follows:

1. Start Lombardi Authoring Environment and open the Process Center Console as explained in [Starting Lombardi Authoring Environment](#).
2. Go to Process Apps, select a process application, and then click the History option as shown in the following image:



3. Scroll through the entries to find validation errors resulting from a snapshot installation.
4. Click the link in an entry to see the listed item or more detail.

Customizing process application installations on offline Process Servers

When installing process applications to offline Process Server in your Lombardi environment, you have the option of customizing those installations. You can choose from the following methods to customize installation of process applications on offline Process Servers:

Method	See...
Create a custom script to use with the <code>installProcessAppPackage.cmd (.sh)</code>	Creating a custom script
Override the default settings for <code>retrieveProcessAppPackage.cmd (.sh)</code> and <code>installProcessAppPackage.cmd (.sh)</code> at the command prompt or using the <code>process-installer.properties</code> file	Overriding default offline install settings

Method	See...
Perform each step of the installation separately, allowing for more robust customization between steps	Performing offline installation steps separately

Creating a custom script

The standard procedure for installing process applications to offline Process Servers is described in [Installing process applications: offline Process Servers](#). To create a custom script to use with the `installProcessAppPackage.cmd (.sh)` utility that you implement during the standard procedure:

1. On the offline Process Server where you want to customize the installation, open a command prompt and go to the `[Lombardi_home]/tools/process-installer` directory.
2. Open the `customProcessAppPreInstall.cmd (.sh)` file. For example, on Windows you can enter the following at a command prompt:

```
notepad customProcessAppPreInstall.cmd
```

3. Add commands that you want to run immediately before the process application snapshot is installed on the offline Process Server and then save your changes.
4. Run the `installProcessAppPackage.cmd (.sh)` utility as described in [Installing process applications: offline Process Servers](#). This utility automatically invokes `customProcessAppPreInstall.cmd (.sh)` before beginning the installation.


Overriding default offline install settings

You can also customize your offline installation by overriding the defaults for the `retrieveProcessAppPackage.cmd (.sh)` and `installProcessAppPackage.cmd (.sh)` utilities. You can override the defaults at the command line or by using the settings in the `[Lombardi_home]/tools/process-installer/process-installer.properties` file. For example, you can change the default host, port, user name, and password used for offline installation.

The `process-installer.properties` file includes the following settings:

Setting	Default	Description
<code>pcs.host</code>	Local host	Specify the name of the host on which to run the utility.
<code>pcs.port</code>	Port used by Lombardi server on local host	Specify the port used by the Lombardi server on the specified host.
<code>pcs.username</code>	<code>tw_admin</code>	The user account specified must be a member of the <code>tw_admins</code> security group in order to install to offline production servers. For other types of servers (staging and test), the specified user must be a member of the <code>tw_authors</code> security group.
<code>pcs.encrypted</code>	<code>tw_admin (encrypted password)</code>	The encrypted password for the specified user. See <i>Encrypting passwords</i> in Lombardi installation guides for instructions.

You can specify the following options at the command-line when running the `retrieveProcessAppPackage.cmd (.sh)` and `installProcessAppPackage.cmd (.sh)` utilities:

Option	Description	Default
-h [<i>host</i>], --host [<i>host</i>]	The host on which to run the offline installation utility.	Set in the <code>process-installer.properties</code> file.
-t [<i>port</i>], --port [<i>port</i>]	The port used by the Lombardi server on the specified host.	Set in the <code>process-installer.properties</code> file.
-u [<i>username</i>], --username [<i>username</i>]	 <p>The user account to run the offline installation utility.</p> <p>The user account specified must be a member of the <code>tw_admins</code> security group in order to install to offline production servers. For other types of servers (staging and test), the specified user must be a member of the <code>tw_authors</code> security group.</p>	Set in the <code>process-installer.properties</code> file.
-p [<i>password</i>], --password [<i>password</i>]	The password for running the offline installation utility.	Set in the <code>process-installer.properties</code> file.
--branch [<i>branch_name</i>]	The name of the workspace in which the snapshot to be installed resides.	No default exists.
-?, --help	Displays these command-line options and a description for each option.	No default exists.

Performing offline installation steps separately

To perform each step of process application installation separately, you can run the following commands individually or add them to a batch file or script, which enables you to perform customization tasks between installation steps. These commands are located in the `[Lombardi_home]/tools/process-installer` directory.

For each of the following commands, you must specify the name of the installation package. For example:

```
importProcessAppPackage.cmd [install_package_name]
```

Where `[install_package_name]` is the name that was assigned to the installation package using `retrieveProcessAppPackage.cmd` as described in [Installing process applications: offline Process Servers](#).

Command	Description
<code>importProcessAppPackage.cmd (.sh)</code>	Imports the installation package (zip file) so that all library items within it are available for customization or alteration via custom scripts or the process application's installation service.
<code>customProcessAppPreInstall.cmd (.sh)</code>	Runs any commands that you add. You should add commands that you want to run immediately before the process application snapshot is installed on the offline Process Server as instructed in Creating a custom script .
<code>executeProcessAppInstallationService.cmd (.sh)</code>	Runs the installation service for each referenced toolkit and then runs the installation service for the process application.
<code>migrateProcessAppGlobalData.cmd (.sh)</code>	Migrates data such as participant groups, EPVs, and environment variables to the new snapshot version that you are installing. The Process Server migrates data according to the rules outlined in Data migration rules .

Command	Description
migrateProcessAppInstances.cmd (.sh)	Migrates currently running process instances according to the migration option chosen when the installation package was created. Migrating instances describes the migration options. If other snapshot versions are not discovered, migration does not occur.

Configuring KPIs and SLAs

The data that Lombardi tracks and stores for key performance Indicators (KPIs) enable you to analyze process performance as well as create service level agreements (SLAs) as described in the following sections.

Using KPIs

Key performance indicators (KPIs) are measurements that Lombardi tracks at process run time, storing results that you can use to analyze process and task performance in the Optimizer. Lombardi includes the following types of KPIs:

KPIs	Description
Standard KPIs	Reside in the System Data Toolkit. By default, most of the standard KPIs are associated with each activity that you add to a BPD diagram. Click the KPIs option in the properties for an activity to see the associated KPIs. Each of these KPIs has default settings that you can change.
Custom KPIs	You can define custom KPIs and associate them with one or more activities in your BPDs.

When you run instances of a BPD, Lombardi tracks and stores data for configured KPIs in the Performance Data Warehouse. (To learn more about how data is tracked and stored, see [Tracking Lombardi performance data](#).) Lombardi uses stored KPI data when you run certain types of historical analyses in the Optimizer. (Not all historical analyses available in the Optimizer rely on data generated and stored due to KPIs. See [Simulating and optimizing processes](#) for more information.)

See the following topics for more information:

To learn how to...	See...
Develop custom metrics to measure the performance of activities	Creating custom KPIs
Configure the standard KPIs associated with activities by default as well as associate your custom KPIs with activities	Associating KPIs with activities



The standard KPI, Total Time (Clock), is associated with each BPD by default. To view the settings for this KPI, click the Process KPIs tab in the Designer. (You cannot alter the settings for this KPI.)

Using SLAs

You can create service level agreements (SLAs) based on standard and custom KPIs. SLAs enable you to establish a condition for one or more activities that triggers a consequence. For example, you can create an SLA that causes Lombardi to send an email notification when a particular activity takes longer than expected to execute.

When you run instances of your processes, SLA consequences do not trigger until the associated activity starts or completes. For example, if you configure an SLA to send an email notification when a particular activity takes longer than two days to execute, Lombardi does not send the notification at the time that the violation occurs. Lombardi sends the notification at the time that the activity is complete. So, if the activity takes three days to complete, Lombardi sends the notification at that time, informing users of the violation. SLAs enable you to easily report on violations and, for example, understand the trend in violations over time.

To enable end users to immediately react to time-based conditions for a single activity, use a timer event to capture the violation. The consequence of the violation can be any type of implementation that you want to develop as described in [Modeling timer events](#). If an SLA is based on something other than time, consider using exposed process values (EPVs) to model the SLA. See [Creating exposed process values \(EPVs\)](#) for more information. To provide immediate notification of violations, develop the appropriate implementation for your needs (such as a timer event for an escalation), and then also create an SLA so that you can track and report on historical trends.

See [Creating SLAs](#) to learn how to configure SLAs. SLAs enable in-depth performance analysis over time as described in the following table:

Analysis	Description
SLA Overview Scoreboard	View this report in Lombardi Process Portal to see the name, description, and current status of each configured SLA, as well as a trend chart of violations for all SLAs or a chosen SLA. See Using out of the box scoreboards for more information.
Custom SLA reports	Leverage SLA data stored in the Performance Data Warehouse to create custom reports using Lombardi or a third-party tool. To learn more about the data that is tracked and stored for SLAs, read about the SLA STATUS and the SLA THRESHOLD TRAVERSALS views in Performance Data Warehouse database architecture . See Reporting options for more information about the types of custom reports you can create.
Historical analysis in Optimizer view	When running scenarios, choose the SLA visualization mode to display results based on SLA violations. See Running simulations, historical analyses, and comparisons for more information.

Creating custom KPIs

To create a custom key performance indicator (KPI):

1. Start Lombardi Authoring Environment and open the appropriate process application or toolkit in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. Click the plus sign next to the **Rules** category and select **Key Performance Indicator** from the list.
3. In the New Key Performance Indicator dialog, type a descriptive name for the new KPI and click the **Finish** button.
4. Optionally provide a description for the KPI in the Documentation field.
5. In the Details section of the dialog, select the Unit for the KPI from the drop-down list.

For example, if your company assembles cell phones and you want to measure the number of phones in production, select **Count** from the drop-down list.

6. If you want the unit that you're tracking to be rolled up into a higher level KPI, click the **Select** button to choose the KPI that you want. (Lombardi displays the KPIs in the current process application and any KPIs in referenced toolkits, including the System Data toolkit.)

For example, select the standard KPI, **Resource Cost**, to roll the Count KPI from the previous step into a higher level KPI.

You can click the **New** button to create a new KPI. Click the X icon to remove a roll-up KPI that you choose.

7. In the Roll-up multiplier field, specify the value by which to multiply the unit that you are tracking before the data is collected in the roll-up KPI.

For example, to obtain an accurate Resource Cost for the previous step, enter the value of each cell phone in the Roll-up multiplier field. Rolling up to higher-level KPIs is useful when creating SLAs. The Resource Cost example in this procedure would enable you to create an SLA for a condition in which the cost of resources in production was either too high or too low.

8. Click the Save icon in the toolbar.

Associating KPIs with activities

Follow these steps to alter associated KPIs or associate a new KPI with an activity in a Business Process Definition (BPD):

1. Click to select the activity that you want in the BPD diagram.
2. Click the KPIs option in the properties.
3. In the Key Performance Indicator section, examine the list of KPIs currently associated with the selected activity. The KPI that you highlight in this list is the one to which the settings in the following steps apply.
4. To remove a KPI, click the KPI that you want and click the **Remove** button.
5. To add a KPI, click the **Add** button and select the KPI or KPIs that you want. (Lombardi displays the KPIs in the current process application and any KPIs in referenced toolkits, including the System Data toolkit.)
6. In the Assignment Settings area, clear the `Use KPI defaults` checkbox if you do not want to use the default assignments for the selected KPI.

Select the assignment type from the drop-down list. The assignment type establishes how the value for the KPI is determined.

For KPIs that measure time, the assignment type is Automatic and cannot be changed. Automatic assignment means that Lombardi automatically tracks and stores the values for these KPIs.

For other KPIs, you can choose from the following assignment types:

Value per Hour (clock)	Enables you to multiply the specified value times the total number of hours spent on the activity.
Value per Hour (calendar)	Enables you to multiply the specified value times the number of working hours spent on the activity.
Absolute Value	Enables you to specify a value for the KPI.
Custom JavaScript	Enables you to supply custom scripts to track the value for this KPI.
True after N traversals (for Rework KPI only)	Enables you to specify the number of times an activity must be performed before it is considered rework.

7. In the Threshold Settings area, clear the `Use KPI defaults` checkbox if you do not want to use the default threshold settings for the chosen KPI.

If you do not use the default thresholds, you can indicate the type of performance that you expect by providing minimum, expected, and maximum values in the appropriate fields.

Lombardi uses the specified thresholds as the range for producing heat maps and recommendations in the Optimizer. You can also use KPIs to specify conditions that trigger service level agreements (SLAs).

8. Click the Save icon in the toolbar to save your changes.

Creating SLAs

The Lombardi Authoring Environment enables you to create Service Level Agreements (SLAs) that you can use to analyze the performance of your business processes over time. See [Using SLAs](#) for more information about how to take advantage of SLAs.

1. Start Lombardi Authoring Environment and open the appropriate process application or toolkit in the Designer view as described in [Starting Lombardi Authoring Environment](#).
2. Click the plus sign next to the **Rules** category in the library and select **Service Level Agreement** from the list.
3. In the New Service Level Agreement dialog, type a descriptive name for the new SLA and click the **Finish** button.
4. Optionally provide a description for the SLA in the Documentation field.
5. In the Trigger section of the dialog, the default trigger statement is displayed:

Whenever the condition is violated.

You can click *Whenever* (displayed in blue font and underlined) to change the trigger for the SLA. For example, if you select **Violated % of the time over period**, the trigger statement changes to:

When the condition was violated *10% of the time* over the *last day*.

You can click *10% of the time* to set the percentage that you want (for example, 20%) and then click *last day* to set the time frame that you want (for example, last 2 days).

6. In the Condition section of the dialog, the default condition statement is displayed:

The Total Time (Clock) KPI for *<select activities>* is *greater than 1 day*.

Click *The* to choose from: Single value, Sum of values over time, or Average value over time.

Click *Total Time (Clock)* to choose the key process indicator (KPI) that you want to use.

Click *<select activities>* to choose the activities to which you want to apply this SLA. All activities are displayed under the BPD in which they reside.

Click *greater than* to choose from: greater than, less than, or equal to.

Click *1 day* to choose from: Threshold, % above threshold, % below threshold, Value above threshold, Value below threshold, or Value. Then set further parameters as necessary.

7. In the Consequence section of the dialog, click the checkbox next to the action that you want to take when the specified condition is violated.

If you choose the Send email option, click *<enter email address>* to provide the address or addresses of the recipients of the notification. Separate addresses with a comma.

If you choose the Initiate process option, click *<select process>* to choose the BPD that you want. (Lombardi displays the BPDs in the current process application and any BPDs in referenced toolkits.) The process that you run as a consequence of the violation must have the following input variables:

Input variable	Type	Description
violationRecord	SLAViolationRecord	Indicates which SLA was violated, to what degree, and when.
parameters	XMLElement	Reserved for future use.

8. Click the **Select** button next to *Expose to view* to choose the participant group whose members can view data for this SLA in the SLA Overview scoreboard in Lombardi Process Portal.
9. Click the Save icon in the toolbar.



When you run instances of your processes, SLA consequences do not trigger until the associated activity starts or completes. SLAs enable you to easily report on violations and, for example, understand the trend in violations over time. You should use other methods, such as timer events, to enable end users to immediately react to time-based conditions. See [Using SLAs](#) for more information.

Creating and configuring reports


Lombardi reporting options provide a powerful way to collect, publish, and consume process performance information. Specifically, Lombardi enables you to easily:

- Define the data that will help you analyze process performance
- Configure the product components to collect this data
- Create the reports that consume the collected process performance data

The following sections provide more information about the reporting options in Lombardi and also describe how Lombardi enables you to collect and analyze performance data.

Reporting options

Lombardi provides the following options for reports:

Report type	Description
Out of the box scoreboards	Enable you to analyze personal performance, team performance, business process performance, and SLA violations with minimal configuration. These scoreboards are available by default in Lombardi Process Portal. See Using out of the box scoreboards for more information.
Customized reports	Enable you to analyze business data specific to your processes. You can define the variables to track and create the customized reports to query your tracked data in the Designer in Lombardi Authoring Environment. Users can view the resulting report scoreboards in Lombardi Process Portal or any browser. The ad-hoc report wizard in the Designer provides a way to quickly configure and publish a report as shown in Creating a quick custom report . Or you can configure a more advanced report as shown in the sample in Creating a more advanced custom report .  Lombardi also enables you to create ad-hoc reports in the Process Portal. To learn more, see the online help for the Process Portal or the <i>Lombardi Process Portal User Guide</i> .
Reports generated in third-party tools	You can query the Lombardi Performance Data Warehouse and extract data using another tool such as Microsoft Access. For more information, see Creating a third-party report .



The reports covered in this document take advantage of the historical data stored in the Performance Data Warehouse database. Customized reports that require in-flight process data from the Process Server database are not covered in this document.

How reporting works in Lombardi

You interact with Lombardi as follows to create customized reports:

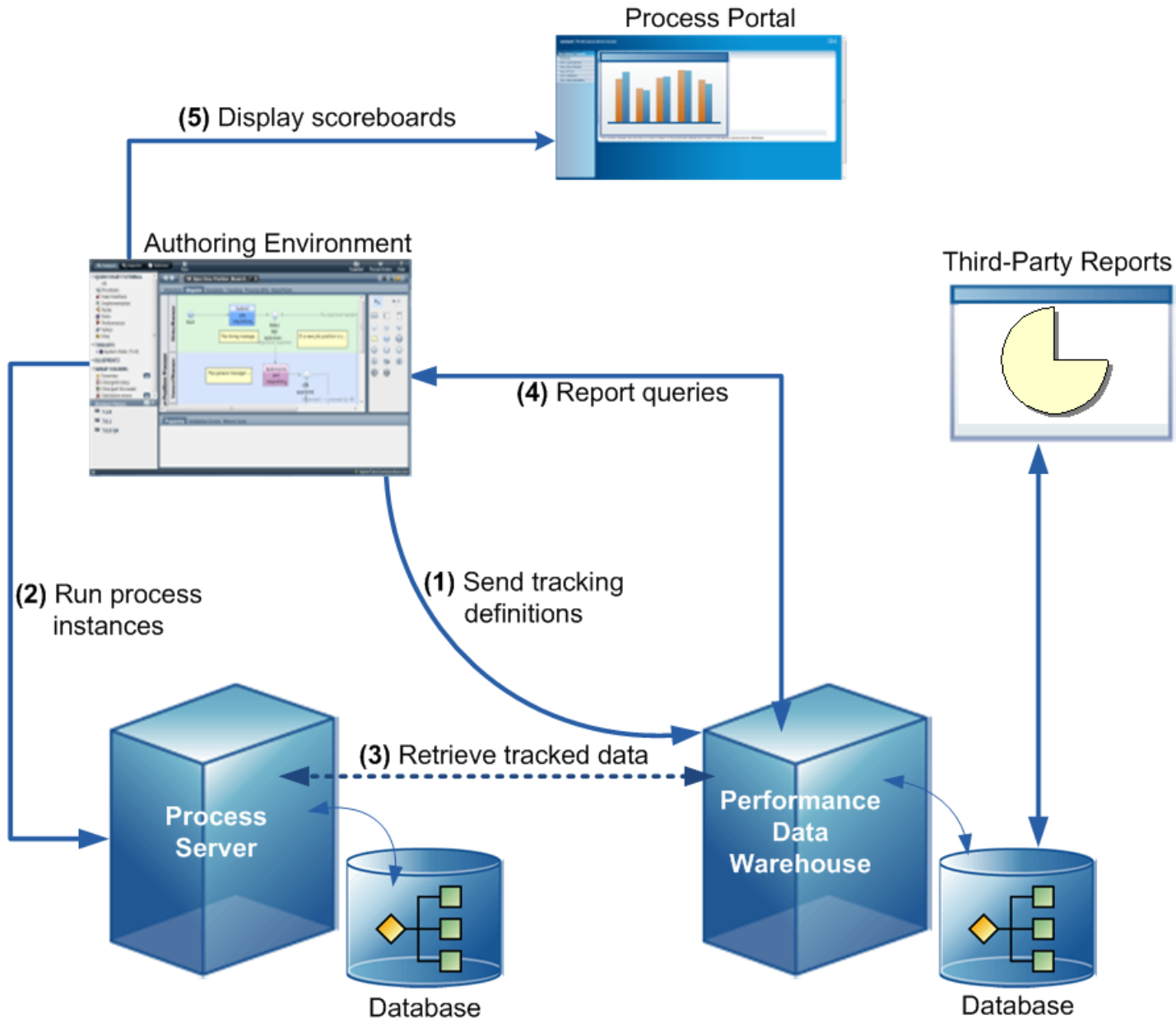
1. In the Designer in Lombardi Authoring Environment, define the variables that you want to track and then send tracking definitions to the Performance Data Warehouse.

The Performance Data Warehouse creates a database table to hold the tracked data.

2. Run instances of your processes on the Process Center Server or a Process Server in a runtime environment.
3. The Performance Data Warehouse retrieves tracked data for each variable from the Process Center Server or Process Server at regular intervals.
4. Create reports in the Designer that query the Performance Data Warehouse to retrieve the required data.
5. Reports that you define in the Designer display as scoreboards in Lombardi Process Portal or a customized portal.

You can also query the Performance Data Warehouse from third-party tools like Microsoft Access to generate reports.

The following image illustrates the preceding interaction:



How Lombardi transfers tracked data

After you send tracking definitions to the Performance Data Warehouse from Lombardi Authoring Environment and then start to run instances of your process, the Performance Data Warehouse retrieves tracked data from the Process Center Server or Process Server at regular intervals.

Lombardi generates and transfers tracked data as follows:

1. When a process participant accesses a task that is part of the process or a system involved in the process generates an event, the Process Server creates the tracked data. The tracked data can include run-time values for the fields in a Tracking Group, values for a Timing Interval, or variables whose values are autotracked.

2. The Process Server writes the tracked data to the Process Server database.
3. The Performance Data Warehouse polls the Process Server database at configurable intervals, checking for a batch of data that is ready to be transferred.
4. In a single transaction, the Performance Data Warehouse marks the data in the Process Server database as transferred (locking it to prevent any further updates), loads the data from the Process Server database to the Performance Data Warehouse database, and then deletes the transfer records from the Process Server database.

Determining which reporting option meets your needs

The following table describes the types of questions that Lombardi reports can help you answer:

Question	Data Tracking Configuration	Report
How many tasks are currently overdue?	None required	Out of the box scoreboards show overdue tasks by user, team, and process.
How many tasks are currently open for an activity?	None required	Out of the box scoreboards show on-track, overdue, and at-risk tasks by activity.
How many tasks are currently at risk for my team?	None required	Out of the box scoreboards show at-risk tasks by user, team, and process.
What is the average duration for each activity in a process?	Use autotracking for the process	Customized or third-party report
What is the total revenue per customer?	Define the business variables to track in the process	Customized or third-party report
Which geographical region generates the most sales leads?	Define the business variables to track in the process	Customized or third-party report
What is the trend of SLA violations by activity per quarter?	Configure KPIs and SLAs for the activities in question and use autotracking for the processes	Customized or third-party report
What is the average time it takes our organization to service deals for a specific customer?	Define the business variables to track in the process and add a timing interval to encompass the process steps involved in servicing deals	Customized or third-party report

The data-tracking options that are available with Lombardi are discussed in [Tracking Lombardi performance data](#).

Keep the following in mind when deciding which reporting option to use:

- The standard out of the box scoreboards provide answers quickly with minimal configuration. For configuration requirements, see [Using out of the box scoreboards](#).
- Custom reports that you can create using autotracked data and the ad-hoc wizard are quick and easy. For step-by-step instructions, see [Creating a quick custom report](#).
- More advanced custom reports that analyze data specific to a process or set of processes can be created by using tracking groups and by creating the required components in the Designer in Lombardi Authoring Environment. For step-by-step instructions, see [Creating a basic custom report](#) and [Creating a more advanced custom report](#).

Using out of the box scoreboards

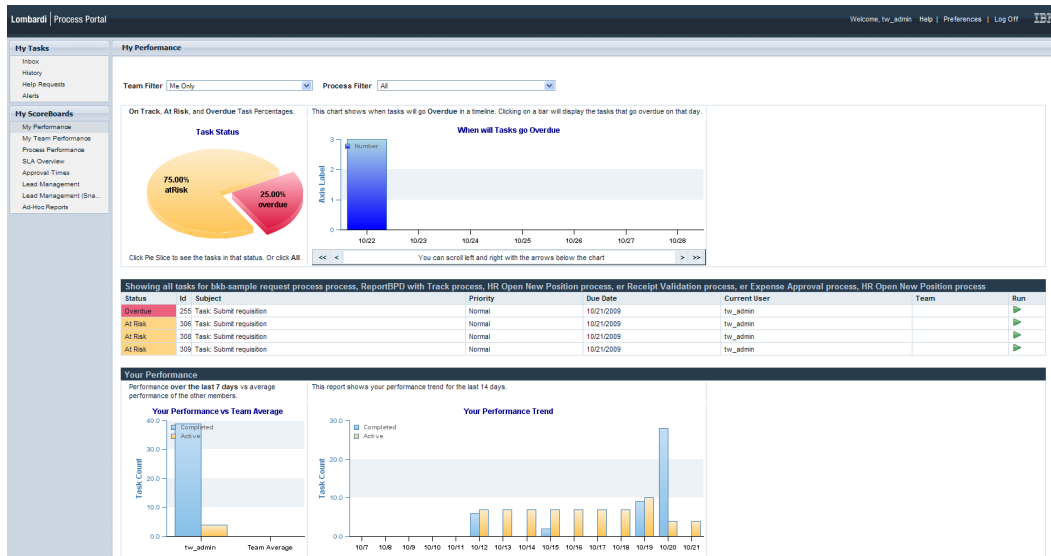
Lombardi includes several standard, out of the box scoreboards in the Process Portal that are designed to help you manage your business processes. The following sections provide more information about each scoreboard, including any configuration requirements to enable process participants to access these scoreboards from Lombardi Process Portal.

My Performance

The My Performance scoreboard shows the current status of your Lombardi tasks for either a specific Lombardi group, or all the groups of which you are a member. You can also choose a specific business process to analyze or view data for all the processes in which you participate. The report details include:

- A pie chart that shows the percentage of tasks on target, at risk, or overdue.
- A bar chart that shows when upcoming tasks will become overdue.
- A task list that shows the process status for each of your assigned tasks.
- A bar chart that compares your individual performance to the team/group average.
- A bar chart that shows the trend of your performance (completed versus active tasks).

The following image shows the My Performance scoreboard in Lombardi Process Portal:



You must have assigned Lombardi tasks due to in-flight or completed process instances in order to see report details in the My Performance scoreboard.

By default, this scoreboard can be accessed in Lombardi Process Portal by members of the All Users participant group. A Lombardi administrator can change the members of the All Users participant group to ensure that all users who need access in each configured Lombardi environment are given access. An administrator can start the Process Admin Console for the appropriate server (Process Server in runtime environment or Process Center Server in development environment), go to the Installed Apps area of the

Process Admin Console, select the current snapshot of the Process Portal process application, select the Role Bindings option, and then adjust members as described in [Configuring runtime participant groups](#).



The current snapshot of the Process Portal process application is named to match the release of Lombardi with which it was shipped. For example, the snapshot for Lombardi 7.0.0 release has a label of 7.0.0.

My Team Performance

The MyTeam Performance scoreboard shows the current status of the Lombardi tasks for Lombardi groups for which you are designated as a team manager. You can also choose a specific business process to analyze or view data for all the processes in which the currently selected team participates. The report details include:

- A pie chart that shows the percentage of tasks on target, at risk, or overdue.
- A bar chart that shows when upcoming tasks will become overdue.
- A task list that shows the status for each of the team's assigned tasks. The task list helps you understand the status of currently running processes and whether reallocating tasks to balance workloads is necessary.
- A bar chart that compares the performance of individual team members. The chart shows both current and completed tasks for each individual participating in the selected process(es). Use this chart to determine the best work allocations.

The following example describes how a manager can use the My Team Performance scoreboard to understand and balance workloads for team members:

Evan reports to Carl in IT and he also participates in an HR process that Sarah manages. Evan has five active tasks in the HR process and one active task in the IT process. Evan has previously completed two tasks in the HR process, but has not yet completed any tasks in the IT process.

In the My Performance scoreboard, Carl selects *IT team* from the **Team Filter** list and *All* from **Process Filter** and sees:

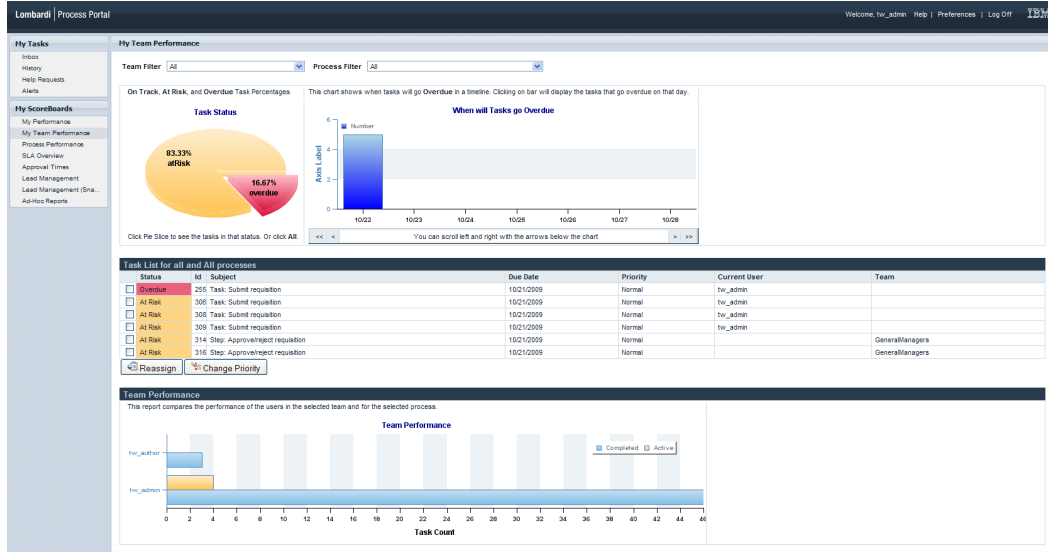
- One task in the Task List. Evan has only one active task as member of the selected IT team.
- Eight tasks in the Team Performance bar chart. This represents Evan's total current workload (six tasks), plus the tasks that Evan has already completed (two tasks), across all of the processes that Evan participates in (HR and IT).

Now, Carl selects the IT process from the **Process Filter** and sees:

- One task in the Task List. Evan has only one active task in the selected IT process.
- No data in the Team Performance bar chart. Evan has not yet completed any tasks in the IT process.

So, Carl sees that Evan is a shared resource with the bulk of his current workload in the HR process. Carl can now better determine if it makes sense to assign more work to Evan, given Evan's total current workload and past performance.

The following image shows the My Team Performance scoreboard in Lombardi Process Portal:



Process participants must have tasks assigned to them due to in-flight or completed process instances in order for team managers to see report details in the My Team Performance scoreboard.

Lombardi administrators can use the Process Admin Console to designate one security group as the team manager for another group. See *Managing group membership* in the *Lombardi Administration Guide* or *Process Admin Console online help* for step-by-step instructions. When you log in to Lombardi Process Portal as a member of a team manager group, you see task information for the members of the group that you manage.

By default, this scoreboard can be accessed in Lombardi Process Portal by members of the Managers participant group. A Lombardi administrator can change the members of the Managers participant group to ensure that all users who need access in each configured Lombardi environment are given access. An administrator can start the Process Admin Console for the appropriate server (Process Server in runtime environment or Process Center Server in development environment), go to the Installed Apps area of the Process Admin Console, select the current snapshot of the Process Portal process application, select the Role Bindings option, and then adjust members as described in [Configuring runtime participant groups](#).



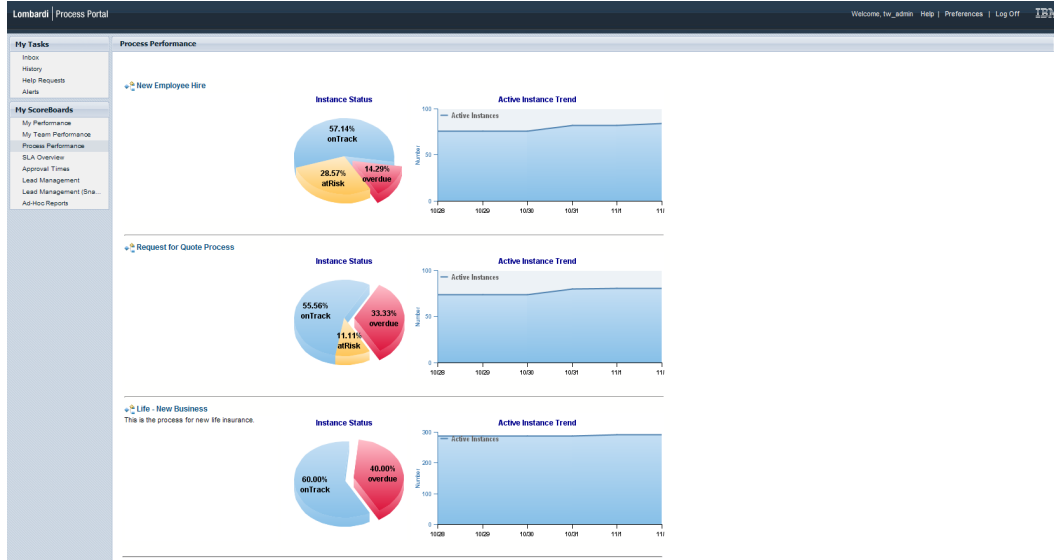
The current snapshot of the Process Portal process application is named to match the release of Lombardi with which it was shipped. For example, the snapshot for Lombardi 7.0.0 release has a label of 7.0.0.

Process Performance

The Process Performance scoreboard shows the current status of the active instances of particular processes in your organization. The specific processes that are included in the Process Performance scoreboard can be configured as explained in the following procedure. For each process, the performance data in the scoreboard includes:

- A pie chart that shows the percentage of tasks on target, at risk, or overdue for the currently active instances of the process.
- A trend chart of the number of active instances of the process over the last week.

The following image shows the Process Performance scoreboard in Lombardi Process Portal:



You can control which processes are included in the Process Performance scoreboard from the Designer in Lombardi Authoring Environment. Open each process (BPD) that you want to include and enable the **Expose performance metrics** setting in the Overview tab. For step-by-step instructions, see [Exposing BPDs](#).



The Business Process Definitions (BPDs) that you choose to include must have autotracking enabled and you must send tracking definitions to the Performance Data Warehouse. If not, performance data will not be displayed in the Process Performance scoreboard. To learn more about autotracking, see [Tracking Lombardi performance data](#).

By default, this scoreboard can be accessed in Lombardi Process Portal by members of the Process Owner participant group. A Lombardi administrator can change the members of the Process Owner participant group to ensure that all users who need access in each configured Lombardi environment are given access. An administrator can start the Process Admin Console for the appropriate server (Process Server in runtime environment or Process Center Server in development environment), go to the Installed Apps area of the Process Admin Console, select the current snapshot of the Process Portal process application, select the Role Bindings option, and then adjust members as described in [Configuring runtime participant groups](#).

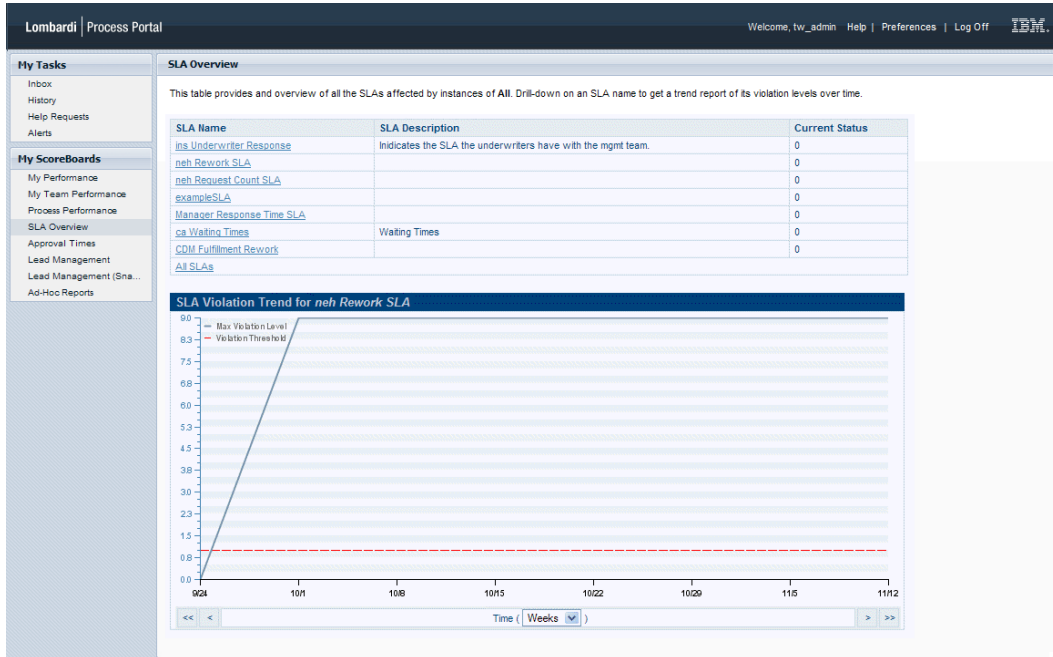


The current snapshot of the Process Portal process application is named to match the release of Lombardi with which it was shipped. For example, the snapshot for Lombardi 7.0.0 release has a label of 7.0.0.

SLA Overview

If SLAs have been defined, the SLA Overview scoreboard displays a table that lists the name, description, and current status of particular SLAs. The specific SLAs that are included in the SLA Overview scoreboard can be configured as explained in the following procedure. You can click any of the SLAs listed in the table or the **All SLAs** entry to display a trend chart of violations for the chosen SLA.

The following image shows the SLA Overview scoreboard in Lombardi Process Portal:



You can control which SLAs are included in the SLA Overview scoreboard from the Designer in Lombardi Authoring Environment. Open each SLA that you want to include and enable the `Expose` setting. For step-by-step instructions, see [Creating SLAs](#).

By default, this scoreboard can be accessed in Lombardi Process Portal by members of the Process Owner participant group. A Lombardi administrator can change the members of the Process Owner participant group to ensure that all users who need access in each configured Lombardi environment are given access. An administrator can start the Process Admin Console for the appropriate server (Process Server in runtime environment or Process Center Server in development environment), go to the Installed Apps area of the Process Admin Console, select the current snapshot of the Process Portal process application, select the Role Bindings option, and then adjust members as described in [Configuring runtime participant groups](#).



The current snapshot of the Process Portal process application is named to match the release of Lombardi with which it was shipped. For example, the snapshot for Lombardi 7.0.0 release has a label of 7.0.0.

Tracking Lombardi performance data

To create customized and third-party reports in Lombardi, you need to identify the data to track and push that data to the Performance Data Warehouse. The following sections describe the data tracking options in Lombardi.

Tracking options

To track data, you must use autotracking or create tracking groups. You can also take advantage of both tracking methods in a single BPD. The following table describes the two tracking methods:

Tracking Method	Description
Autotracking	Use if you want to capture data to quickly configure and publish reports using the ad-hoc wizard. You can also use autotracking if you want to capture data that automatically includes tracking points at the entry and exit of each item in a BPD (such as activities,

Tracking Method	Description
	services, and gateways). For example, if you know that you want to compare the duration for each activity in a BPD, autotracking enables you to do so. When you enable autotracking for a BPD, you also track the Key Performance Indicators (KPIs) associated with your BPD. And, when you track KPIs, you can use that data as a condition to trigger Service Level Agreements (SLAs). The About autotracking section that follows provides more information.
Tracking groups	Use if you want to explicitly control your tracked data and tracking points for more advanced custom reports. For example, you can group the variables that you want to track by type, strategically place tracking points in your BPD, and track variables across multiple BPDs. With tracking groups, your tracking points can also span multiple BPDs. The About tracking groups section that follows provides more information. The About timing intervals section that follows describes tracking points and using a timing interval to measure the duration between the points.

About autotracking

To learn how to use autotracking, see [Creating a quick custom report](#).

When you use autotracking, the following Key Performance Indicators (KPIs) are tracked:

- Custom KPIs associated with your BPD
- Custom KPIs associated with the activities in your BPD
- The TotalTime KPI associated with each BPD by default

KPIs act as conditions for Service Level Agreements (SLAs), which you can use to trigger a particular consequence such as an email notification. To learn how to create custom KPIs and SLAs, see [Configuring KPIs and SLAs](#). You can analyze the performance of your SLAs using the SLA Overview out of the box scoreboard or by creating custom reports. To create custom SLA reports, use the SLA STATUS view and the SLA THRESHOLD TRAVERSALS view in the Performance Data Warehouse database. See [Performance Data Warehouse database architecture](#) for more information about these views.

About tracking groups

When you want to create more advanced custom reports, you can track process data in Lombardi Authoring Environment by creating tracking groups. Tracking groups provide the following advantages:

- You can group similar data together for analysis via reports. For example, you can track employee data, account data, or shipment status information in independent tracking groups.
- You can track process variables across multiple BPDs and process applications. For example, if your organization includes several locations and each location has a similar, but unique, onboarding process for new employees, you can create a tracking group to capture the business data for all of these processes. See [Tracking data across processes and process applications](#) for more information.
- Tracking points for a timing interval can span multiple BPDs. For example, if you want to measure the duration between steps that start in one process and end in a nested process, you can include the start tracking point in the main process and the end tracking point in the nested process.

To learn how to create tracking groups, see [Creating a more advanced custom report](#).

About timing intervals

If you want to create reports to analyze the amount of time that elapses between certain steps in your process, you can add tracking points to your BPD and then create a timing interval to capture the duration between a starting point and an ending point. When using timing intervals, you should autotrack the process variables that you want to capture and then create a tracking group to hold the timing interval data as explained in [Creating a basic custom report](#).

Sending tracking definitions

You must send tracking definitions to the Performance Data Warehouse as described in the following table. If you fail to send tracking definitions, the Performance Data Warehouse does not track performance data as expected.

Server	Description	How to send tracking definitions
Process Center Server	When you use autotracking, manually create or edit tracking groups, or perform any other task in the Designer in Lombardi Authoring Environment to capture performance data, you must send these tracking requirements to the Performance Data Warehouse if you plan to run your processes on the Process Center Server to test data tracking and reports.	Choose File > Send Definitions to Performance Data Warehouse from the Authoring Environment main menu.
Process Servers in runtime environments	When you install snapshots of process applications on Process Servers in runtime environments, all tracking definitions are automatically sent to the Performance Data Warehouse in the selected runtime environment. This ensures that your data is tracked as expected when instances of your processes are executed in the runtime environment.	No need to send tracking definitions unless a problem occurs during snapshot installation. If a problem does occur, you can select the Send Tracking Definitions option for the snapshot as described in Configuring installed snapshots .



Definitions must be sent to the Performance Data Warehouse when you make any changes to your tracking requirements in the Designer in Lombardi Authoring Environment. So, when developing on the Process Center Server, be sure to send definitions when you makes changes. For process applications installed in runtime environments, snapshot any changes and install the new snapshot to ensure that the data you want to collect is available in the runtime environment.

When you send tracking definitions, either directly or as part of a snapshot installation, the Performance Data Warehouse establishes the structure in its database to hold the data that is generated by the Process Server when you run instances of your processes. In Lombardi, these tracking requirements are called *definitions* because they establish the database schema in the Performance Data Warehouse to accommodate the tracked data generated by the Process Server. [Creating a basic custom report](#) describes how to verify that the required database structure has been established in the Performance Data Warehouse.

Supported data types

Data types that Lombardi tracks include the following:

Type of tracking	Supported data types
Autotracking	String, Integer, Decimal, Boolean, and Date
Tracking groups	String, Number, and Date

When tracking data, be aware of the following:

- Variables for which the **Is List** option is enabled cannot be tracked.
- Complex types cannot be mapped directly; their fields must be mapped individually.
- Variables of type ANY, Map, Record, XMLDocument, XMLElement, and XMLNodeList cannot be tracked.

Naming tracking groups

When naming tracking groups and tracked fields, be aware of the following restrictions:

- Do not use SQL-92 reserved words. Several sources available on the internet provide a complete list of SQL-92 reserved words.
- Do not use any of the names used for the views and fields in the Performance Data Warehouse database schema. ([Performance Data Warehouse database architecture](#) provides details about the schema.)

Tracking data across processes and process applications

To track data from multiple processes (BPDs) that reside in the same process application, create a tracking group and implement it for as many BPDs as you like, mapping the tracked fields to the appropriate variables for each BPD. To learn how to create and implement tracking groups, see [Creating a more advanced custom report](#).

If you want to capture data from multiple processes (BPDs) that reside in different process applications, you can do so by using the same tracking group in each process application. For example, you can create a tracking group in a toolkit, and then create a dependency on that toolkit in each process application where you want to use the tracking group. From each process application, you can implement the tracking group one or more times, mapping the tracked fields to variables within each application. When you send tracking definitions and then run instances of the BPDs, the data is captured in a single tracking group view as described in [Tracking Group views](#). The data that Lombardi captures enables you to analyze the tracked data in any way you choose. For example, you can analyze the tracked fields as a whole or you can compare the data from each process application or from each process.

Working with versioned data

All data tracked by Lombardi includes snapshot (version) information that enables you to create reports to compare versions of your processes if you have that requirement. See [Performance Data Warehouse database architecture](#) to learn about the views that you can query and the version data that they provide.

When tracking data, keep the following in mind:

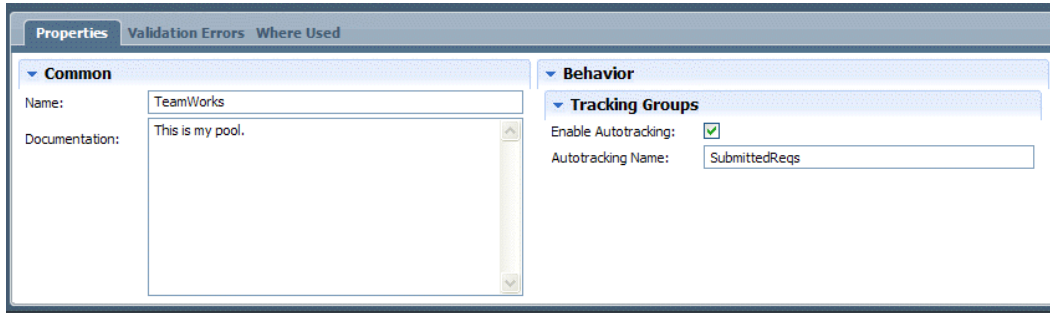
- Timing intervals work across snapshots (versions). For example, a process that starts in one version (1.0) might be migrated to a new version (2.0) before reaching the end of a timing interval. In such a case, the data for the timing interval is captured in the Performance Data Warehouse as expected with the version change noted.
- Data types of variables that are being stored (autotracked or part of a tracking group) can change between versions. If data types do change, a new column is created in the corresponding view as described in [Tracking Group views](#).

Creating a quick custom report

The following sections explain how to quickly configure Lombardi to track data and generate a custom report using the ad-hoc wizard.

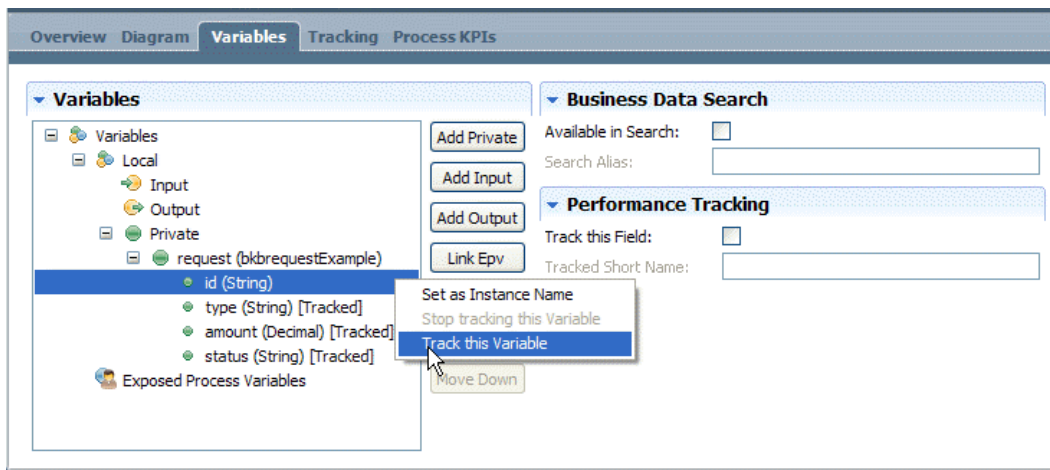
Configuring autotracking

1. To use autotracking, open the BPD diagram in the Designer in Lombardi Authoring Environment, click the Pool, and make sure the Enable Autotracking check box in the Properties tab is enabled (autotracking is enabled by default). For this example, change the default autotracking name to `SubmittedReqs` as shown in the following image:

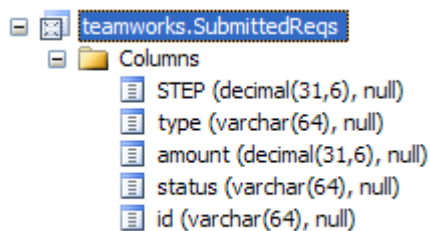


When autotracking is enabled for a BPD, data for any nested processes of that BPD will also be tracked. Sub-processes and services inherit the setting of the parent process.

2. For this example, you want to analyze process data according to particular business variable values, go to the Variables tab for your BPD, right-click each variable that you want to track, and select **Track this Variable** as shown in the following example:



3. Save the BPD and then send these newly defined tracking requirements to the Performance Data Warehouse by selecting **File > Send Definitions to Performance Data Warehouse**. Then you can go to the Performance Data Warehouse database in your development (Process Center) environment to verify that a `SubmittedReqs` view has been created that includes a column for each variable that you tracked:



When you run instances of the process, Lombardi stores the tracked data for each variable in the appropriate column. Each row in a Tracking Group view represents a discrete unit of tracked data.

To use the ad-hoc wizard to quickly create a report that uses this data, see the following example.



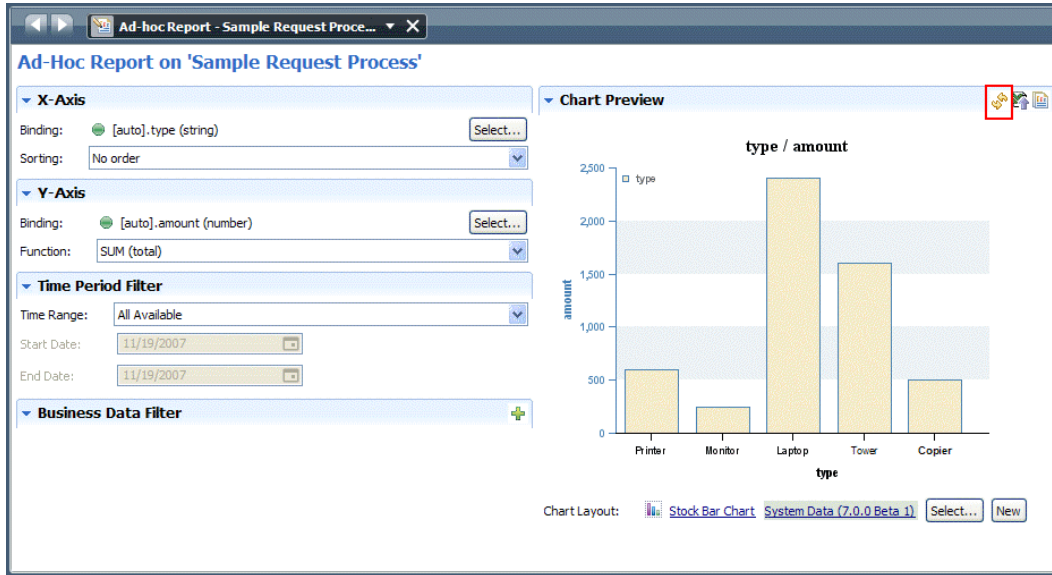
When you autotrack variables, values are tracked for each ordered step in a BPD. If a process author changes the order of the steps in the BPD, or adds one or more steps, custom report queries against the tracked variables will not produce expected results if those queries rely on the order of the steps in the process.

Creating a quick report using the ad-hoc wizard

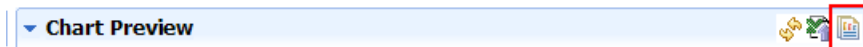
The ad-hoc wizard in the Designer in Lombardi Authoring Environment enables you to quickly generate and publish reports. Using autotracked data with the wizard, you can easily create custom reports to analyze various aspects of your business.

Let's say we want a chart that shows the sum amount for each type of request tracked in the `SubmittedReqs` autotracked data discussed in the preceding section. The following steps outline how to use the ad-hoc wizard to generate such a report:

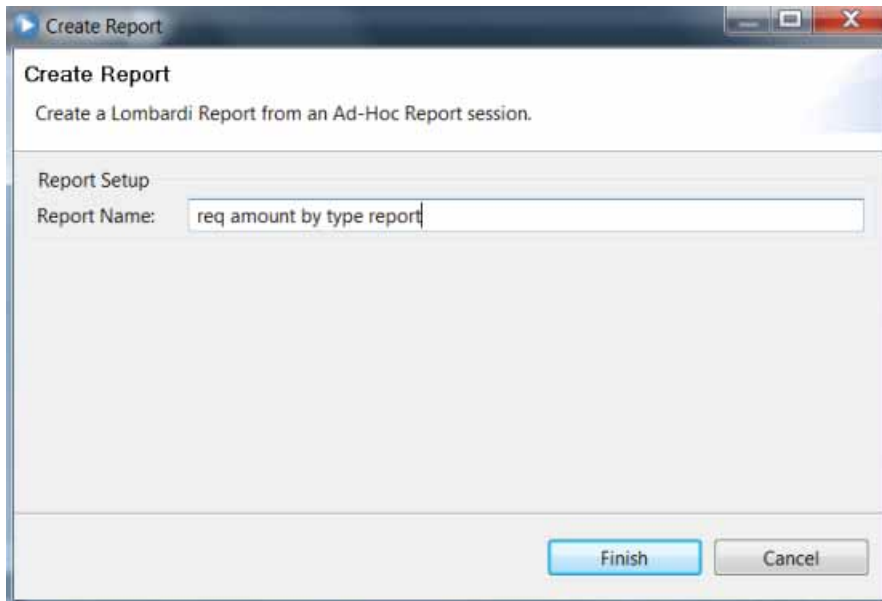
1. In the Designer in Lombardi Authoring Environment, open the diagram of the BPD.
2. Select **File > Ad-Hoc Report Analysis** from the main menu.
3. Choose the settings that you want for the report from the wizard. For example, in the following image, we've selected the `type` variable for the X-axis binding and the `amount` variable for the Y-axis binding with the function of SUM (total) since we want to show the sum amount for each type of request. We leave the default chart layout (Stock Bar Chart), and then click the Refresh icon in the upper right to preview our chart directly within the wizard:



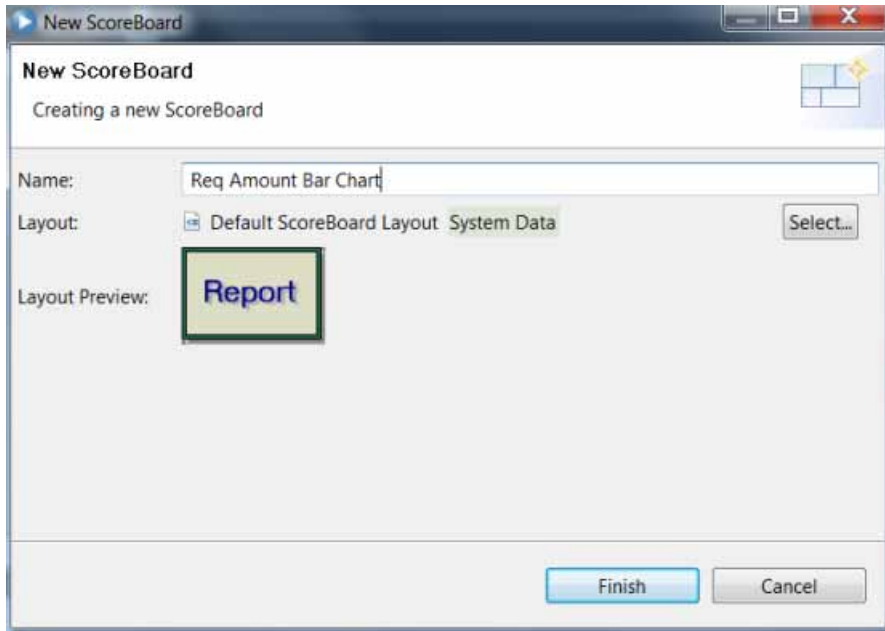
- Satisfied with the chart's appearance and data, we click the Create Report icon in the upper right as shown in the following image:



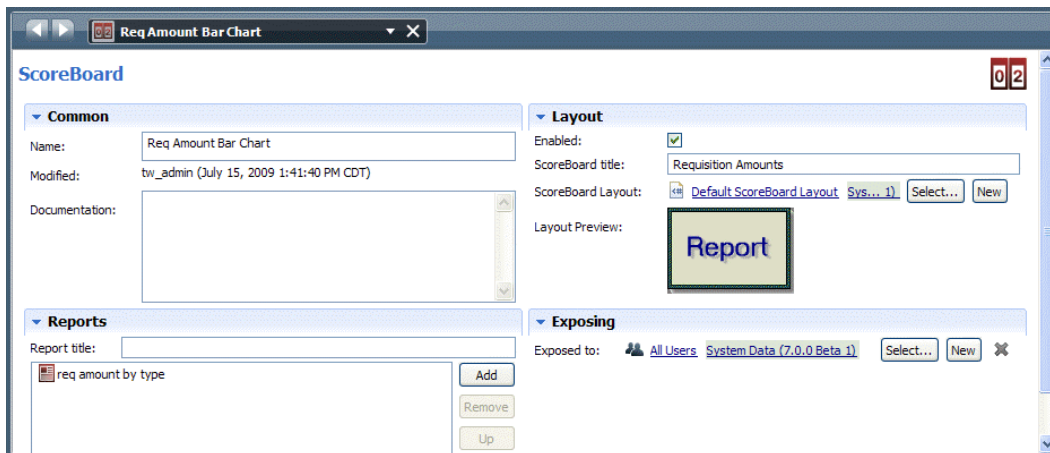
- In the Create Report dialog, we type in a name for the report and click **Finish**.



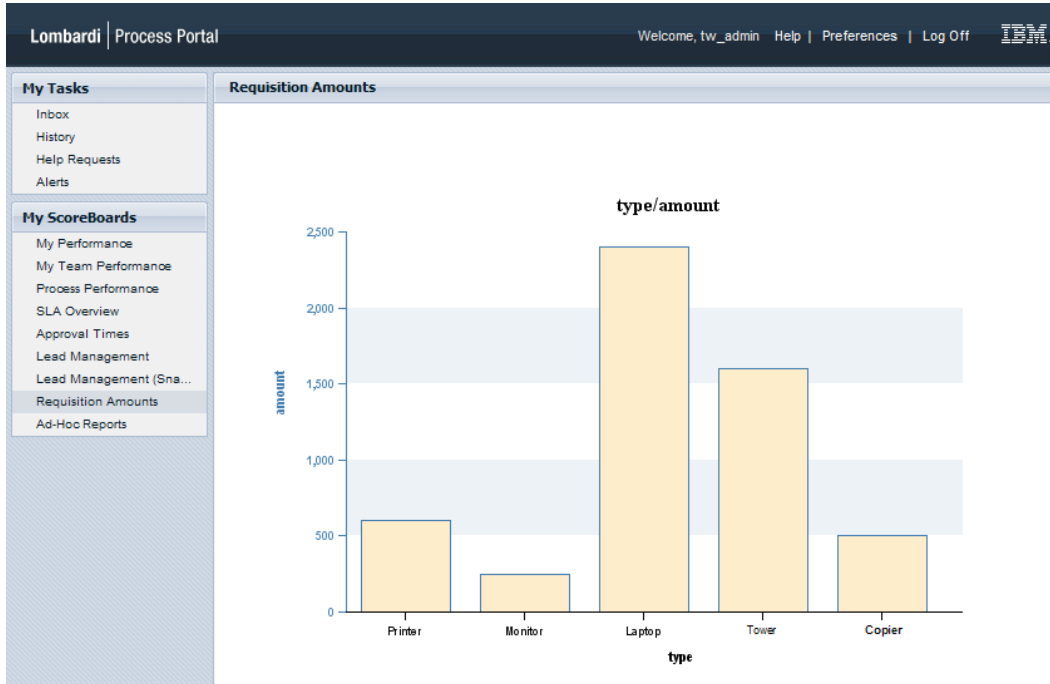
- To create a scoreboard so that we can display the report in Lombardi Process Portal, we click the plus sign next to the Performance category in the library and select **Scoreboard** from the list of components.
- In the New Scoreboard dialog, we name the scoreboard Req Amount Bar Chart, leave the default layout setting (Default Scoreboard Layout), and click **Finish** as shown in the following image:



- In the scoreboard we've created, under the **Reports** heading, we click the **Add** button, find our report, and click it to add it to the scoreboard. Then, under the **Layout** heading, we click the **Enabled** checkbox and provide a title for the scoreboard. We also click the **Select** button next to the **Exposed to** option and pick the participant group whose members can view this scoreboard in Lombardi Process Portal as shown in the following image:



- Now when we log in to Lombardi Process Portal as a member of the participant group to whom the scoreboard is exposed, we can see the title of the Scoreboard (**Requisition Amounts**) in My Scoreboards on the left of the portal. When we click **Requisition Amounts**, Lombardi Process Portal displays the report:



10. To control access to the scoreboard in different environments:

Environment	Required configuration
Development (on Process Center Server)	In the Designer in Lombardi Authoring Environment, create a new participant group or adjust the members of the existing participant group to whom the scoreboard is exposed. See Creating a participant group .
Runtime (Process Server in test, staging, or production environment)	After installing a process application snapshot that includes the scoreboard, adjust the members of the participant group to whom the scoreboard is exposed as described in Configuring runtime participant groups .

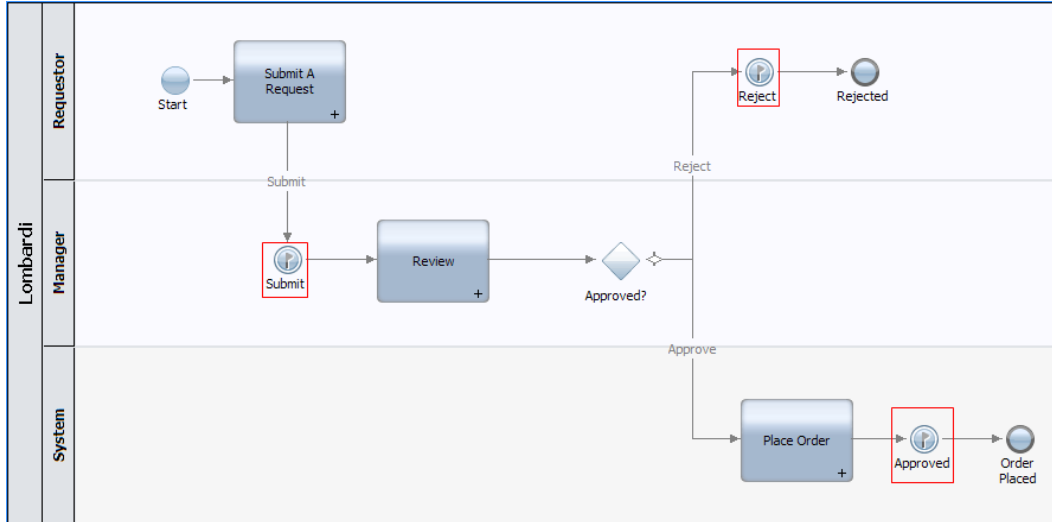
Creating a basic custom report

The following sample shows how to use the ad-hoc wizard to leverage data captured in a timing interval so that you can compare the duration of a set of tasks for different process variables.

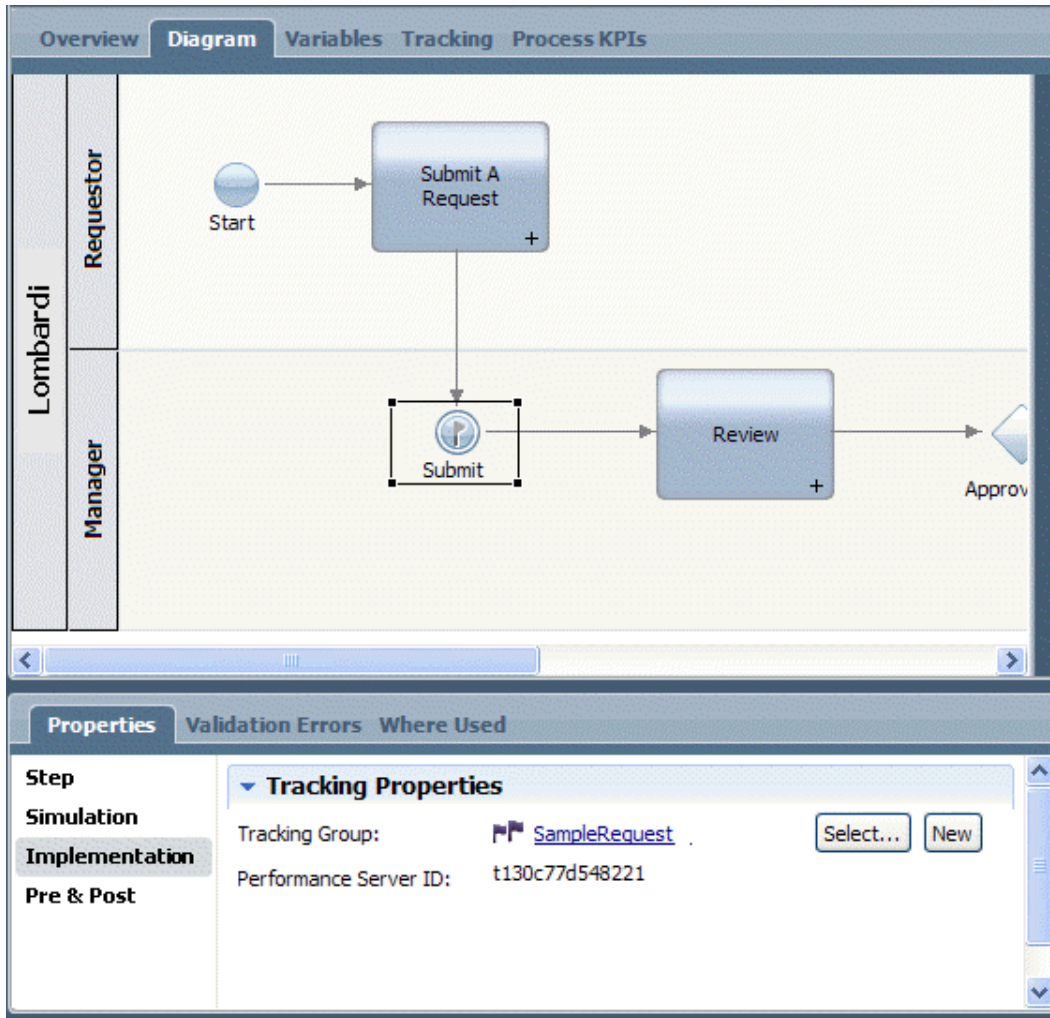
Creating a timing interval

Let's say we have a BPD in which orders are submitted for approval and we want to compare the duration of the approval process for orders from different departments.

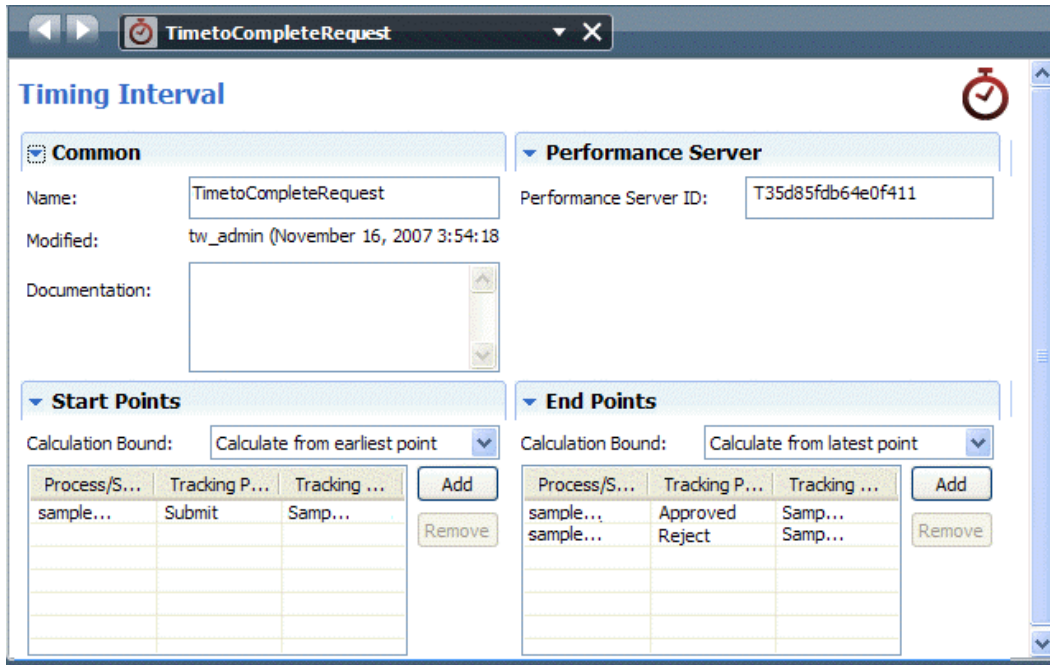
1. For this sample process, we're already autotracking the order ID, type, amount, department, and status.
2. To be able to compare the duration of the approval process for these orders, we go to the BPD diagram and add the Submit, Reject, and Approved tracking points shown in the following image:



3. We need to create a tracking group to hold the timing interval data. In the library in the Designer, click the plus sign next to the Performance category and select **Tracking Group** from the list of components. For this sample, we name the tracking group `sampleRequest` and click **Finish**.
4. Then, for each tracking point, select the icon in the BPD diagram, go to the Implementation option in the properties, and select the `sampleRequest` tracking group as shown in the following image:



- Now we need to create the timing interval. In the library in the Designer, click the plus sign next to the Performance category and select **Timing Interval** from the list of components. After we name the interval (TimeToCompleteRequest) and click **Finish**, we add the start and end tracking points as shown in the following image. Notice that we select the Submit tracking point for the start of the interval with Calculate from earliest point and the Approved and Reject tracking points for the end of the interval with Calculate from latest point:

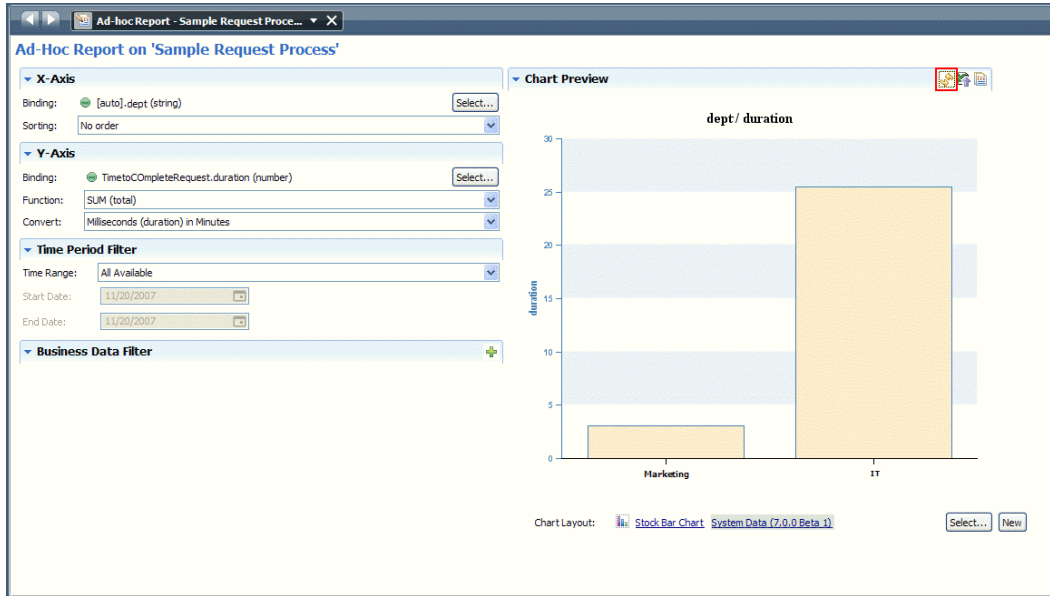


To use this timing interval to compare the duration of orders using the ad-hoc report wizard, see following report sample.

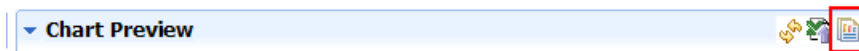
Creating a basic report that uses the timing interval

Now let's use the ad-hoc wizard to take advantage of the timing interval discussed in the preceding section. Because we've configured a timing interval, we can compare the duration of the review process for orders from different departments as outlined in the following steps:

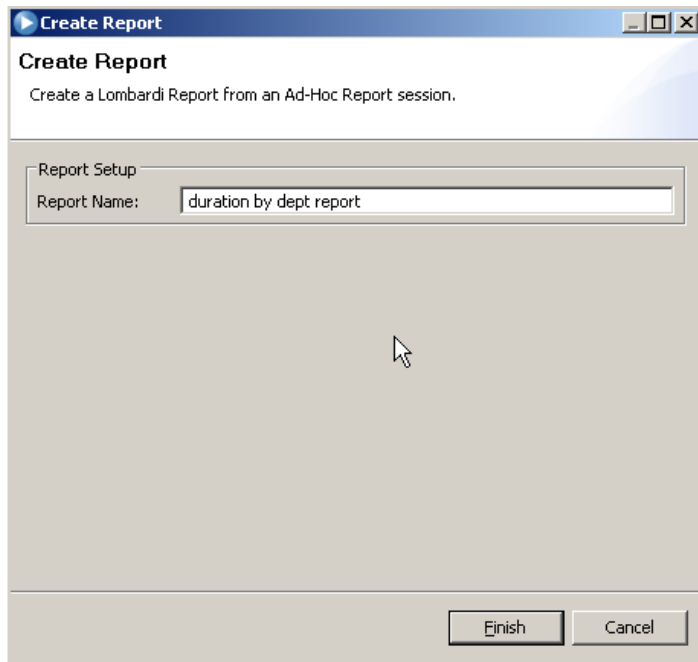
1. In the Designer in Lombardi Authoring Environment, open the diagram of the BPD.
2. Select **File > Ad-Hoc Report Analysis** from the main menu.
3. Choose the settings that you want for the report from the wizard. For example, in the following image, we've selected the dept variable for the X-axis binding and the TimetoCompleteRequest timing interval for the Y-axis binding with the function of SUM (total) since we want to show the total duration. We choose Milliseconds in Minutes for the Convert field since we want to see duration in minutes. We leave the default chart layout (Stock Bar Chart), and then click the Refresh icon in the upper right to preview our chart directly within the wizard:



- Satisfied with the chart's appearance and data, we click the Create Report icon in the upper right as shown in the following image:

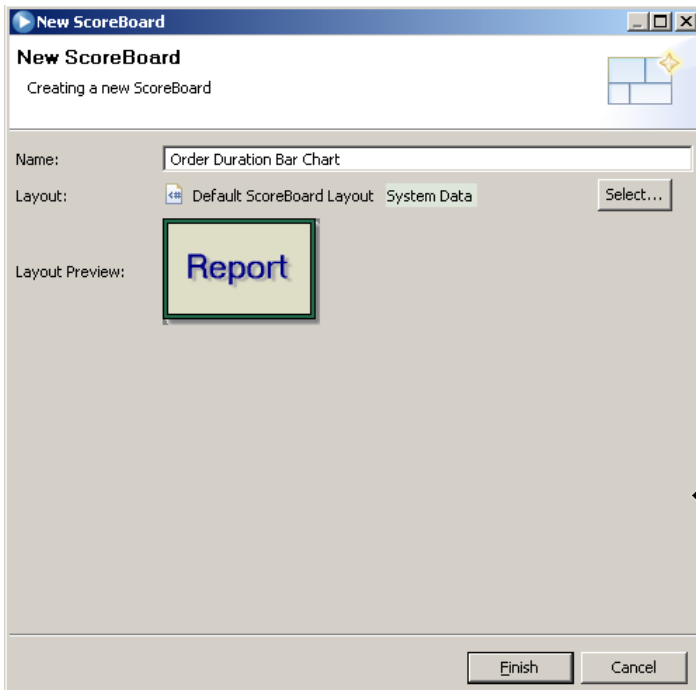


- In the Create Report dialog, we type in a name for the report and click **Finish**.

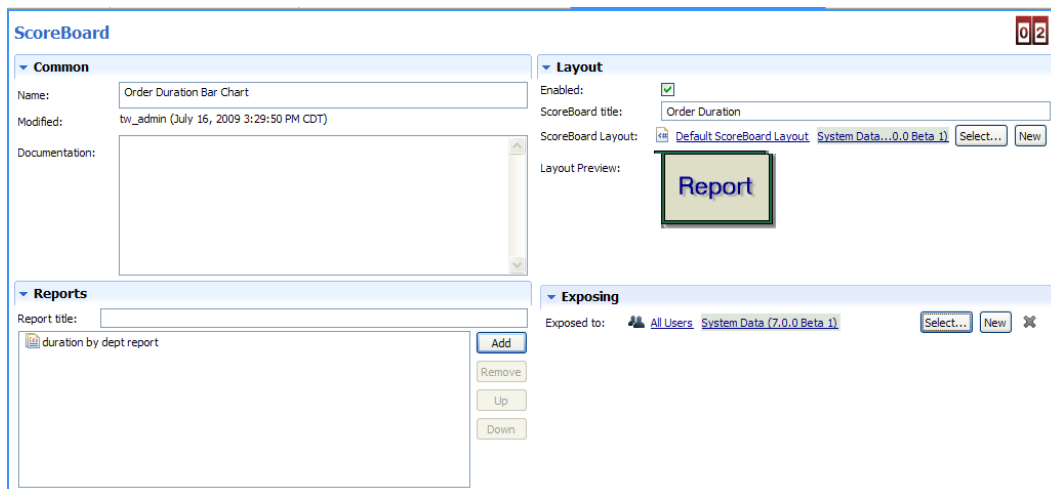


- To create a scoreboard so that we can display the report in Lombardi Proses Portal, click the plus sign next to the Performance category in the library and select **Scoreboard** from the list of components.

- In the New Scoreboard dialog, name the scoreboard **Order Duration Bar Chart**, leave the default layout setting (Default Scoreboard Layout), and click **Finish** as shown in the following image:



- In the scoreboard we've created, under the **Reports** heading, we click the **Add** button, find our report, and click it to add it to the scoreboard. Then, under the **Layout** heading, we click the **Enabled** checkbox and provide a title for the scoreboard. We also click the **Select** button next to the **Exposed to** option and pick the participant group whose members can view this scoreboard in Lombardi Process Portal as shown in the following image:



- Now when we access Lombardi Process Portal, we can see the title of the Scoreboard (**Order Duration**) in My Scoreboards on the left of the portal. When we click **Order Duration**, Lombardi Process Portal displays the report:



10. To control access to the scoreboard in different environments:

Environment	Required configuration
Development (on Process Center Server)	In the Designer in Lombardi Authoring Environment, create a new participant group or adjust the members of the existing participant group to whom the scoreboard is exposed. See Creating a participant group .
Runtime (Process Server in test, staging, or production environment)	After installing a process application snapshot that includes the scoreboard, adjust the members of the participant group to whom the scoreboard is exposed as described in Configuring runtime participant groups .

Creating a more advanced custom report

The following example explains how to create a tracking group and then use the data captured for that group to create a more advanced query and report, building each required component in the Designer in Lombardi Authoring Environment.



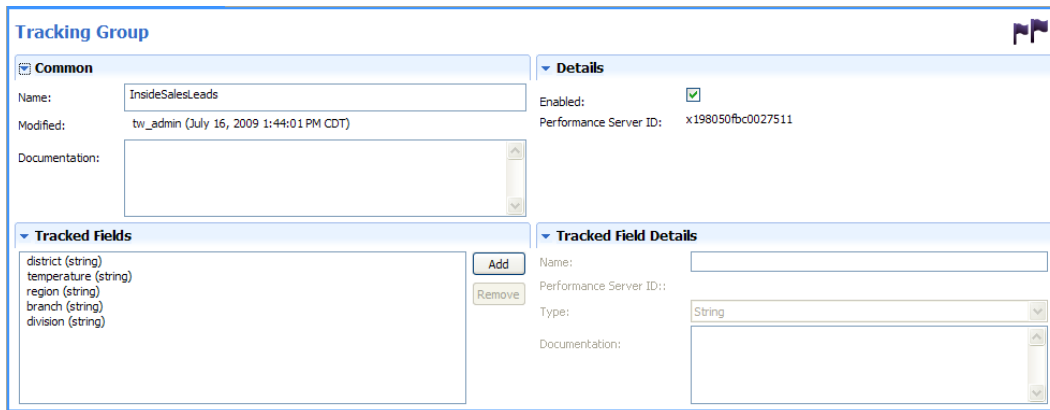
When querying the Performance Data Warehouse, you should query the views as described in [Performance Data Warehouse database architecture](#).

Creating a tracking group

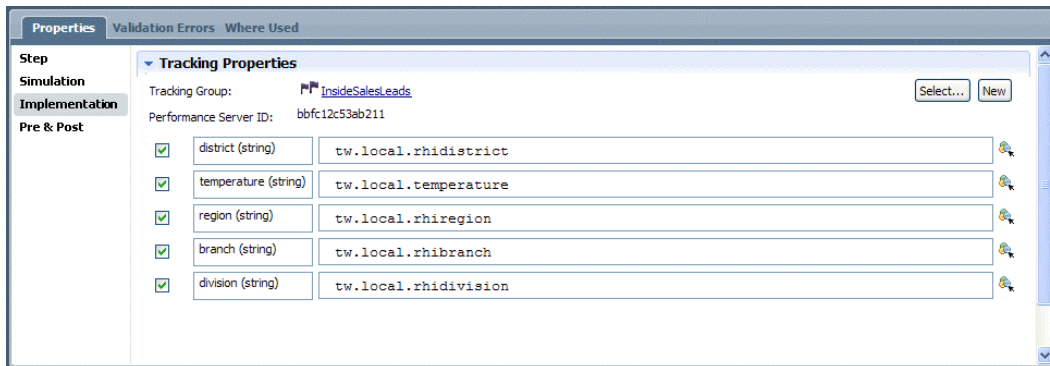
Let's say that you have developed a Business Process Definition (BPD) in Lombardi Authoring Environment that collects data for incoming leads. This process enables your Inside Sales team to quickly capture details about each lead that they identify. Process participants downstream will use the collected data in a nested process to qualify and assign leads. Because the tracked data needs to span multiple BPDs, you decide to create a tracking group to capture the variable values.

The data collected from the Inside Sales team is represented by several variables in the BPD. A variable exists for each field of data that the Inside Sales team provides, including: district, region, branch, division, and temperature (temperature is used to indicate whether the lead is hot or cold). Your entire Sales team realizes early on that they want to be able to capture information from this business process and report on it. For example, they will need to know how many leads came in for each district, region, and branch.

1. To be able to report on this data for this process and the related nested processes, you create a Tracking Group in the library called `InsideSalesLeads` and then add a field to the group for each process variable that you want to track:

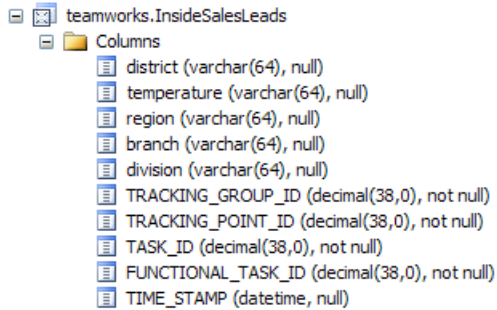


2. Add a Tracking Event to your BPD diagram, select it, and in the Implementation tab, add the process variables to associate with each tracked field as shown in the following image:



3. Save the BPD and then send these newly defined tracking requirements to the Performance Data Warehouse by selecting **File > Send Definitions to Performance Data Warehouse**.

When you view the Performance Data Warehouse database, you can see that sending the definitions caused the Performance Data Warehouse to create a view for your tracking group. The view name matches the name of the tracking group, with the schema name of the database (`teamworks`) prepended. When you access the view created for the tracking group, you can see that a column exists for each variable that you included (district, temperature, region, branch, and division) along with the standard system-generated columns:



When you run instances of the process, the tracked data for each variable is stored in the appropriate column. Each row in a Tracking Group table or view represents a discrete unit of tracked data. When instances of the process run and some leads have actually been entered by the team, we can see the data in the `InsideSalesLeads` tracking group view in the Performance Data Warehouse:

	district	temperature	region	branch
▶	NorthEast	Hot	New York	NY Uptown
	SouthEast	Hot	Florida	Miami
	Central	Cold	Texas	Pleasanton
	West	Hot	California	San Jose
	Baha	Cold	Halisco	Los Gatos
	NorthEast	Cold	New York	Long Island
	Central	Cold	Mountain	Montreal
	SouthEast	Cold	Florida	Miami
	West	Hot	Quebec	Montreal
	West	Cold	California	Santa Cruz
	Central	Hot	Mountain	Montreal

You are able to verify that Lombardi is properly capturing the data by checking the Performance Data Warehouse. To build a custom report that takes advantage of this data, see the following sample.

Steps to create a more advanced custom report

To create a report by building each component in the Designer in Lombardi Authoring Environment, the following are the basic steps to follow:

1. Create an Integration service that contains a query.
2. Create a new report and: a) specify a data source for the report (the Integration service from the preceding step); b) add a chart to the report and bind the chart to the data source; c) create a report page, make it the default page for the report, and then save the report.
3. Create a scoreboard, add the report to the scoreboard, and then enable the scoreboard.
4. Grant access to the end users who want to view this scoreboard by setting the `Expose to` option for the scoreboard.

The following sections cover the preceding steps in more detail and also include some optional steps like adding a filter to the report. Let's use the `InsideSalesLeads` tracking group discussed in the preceding section to create a more advanced query and report. We've already created the business process to collect data from the Inside Sales team and we've also created a field in the `InsideSalesLeads` tracking group for each process variable that we want to track. Using this data, we'll create a report that compares the leads by region. We'll use a stacked bar chart to illustrate the break-down of hot and cold leads per region.

Creating an Integration service that contains a query

1. To create the Integration service for our bar chart in the Designer, we click the plus sign next to the Implementation category in the library and select **Integration Service** from the list of components.
2. In the New Integration Service dialog, type a name in the appropriate field (for this sample, the name is `Bar Chart Example`) and then click **Finish**.
3. Click the Variables tab and add the following private variables for this service: `sql` of type String, `returnType` of type String, and `dataSourceName` of type String. (To learn more about variables, see [Managing and mapping variables](#).)
4. Click the `returnType` variable to select it, enable the `Has Default` checkbox, and type `"XMLElement"` in the text box.
5. Click the `dataSourceName` variable to select it, enable the `Has Default` checkbox, and type `"jdbc/PerformanceDB"` in the text box. This directs the Integration service to the Performance Data Warehouse database.
6. Add an output variable named `results` of type `XMLElement`.
7. Click the Diagram tab and drag a Server Scriptlet component from the palette to the service diagram.
8. While the Server Scriptlet component is still selected, click the **Step** option in the properties and type `SQL Query` in the Name text box.
9. Click the Implementation option in the properties, click the Select button next to the **Binding** option, and choose the `sql` variable created in an earlier step.
10. Enter the following query in the text box. Notice that we're performing a Union of SQL statements to retrieve counts of Hot and Cold leads per region:

```

SELECT
  district, 'Hot' as name, count(*) as var
FROM
  InsideSalesLeads
WHERE
  temperature = 'Hot'
GROUP BY
  district
UNION
SELECT
  district, 'Cold' as name, count(*) as var
FROM
  InsideSalesLeads
WHERE
  temperature = 'Cold'
GROUP BY
  district

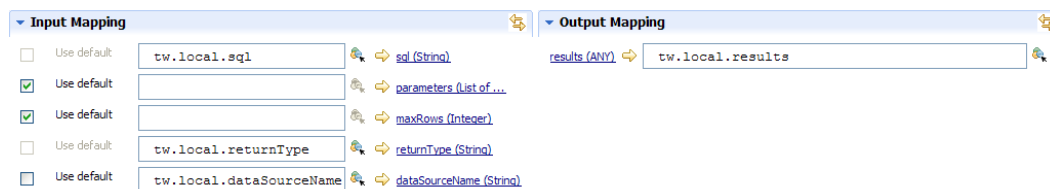
```

11. Drag the SQL Execute Statement Integration service from the System Data toolkit to the service diagram so that it follows the Server Scriptlet component added in the preceding steps.

12. While the SQL Execute Statement Integration service is still selected, click the **Step** option in the properties and type `Execute SQL` in the Name text box.
13. Click the **Data Mapping** option in the properties.
14. For Input Mappings, click the variable selector icon next to `sql (String)` and choose the local `sql` variable created in an earlier step.

Do the same for `returnType (String)` and `dataSourceName (String)`, choosing the corresponding local variables created in an earlier step.

15. For Output Mappings, click the variable selector icon next to `results (Any)` and choose the local output `results` variable created in an earlier step as shown in the following image:



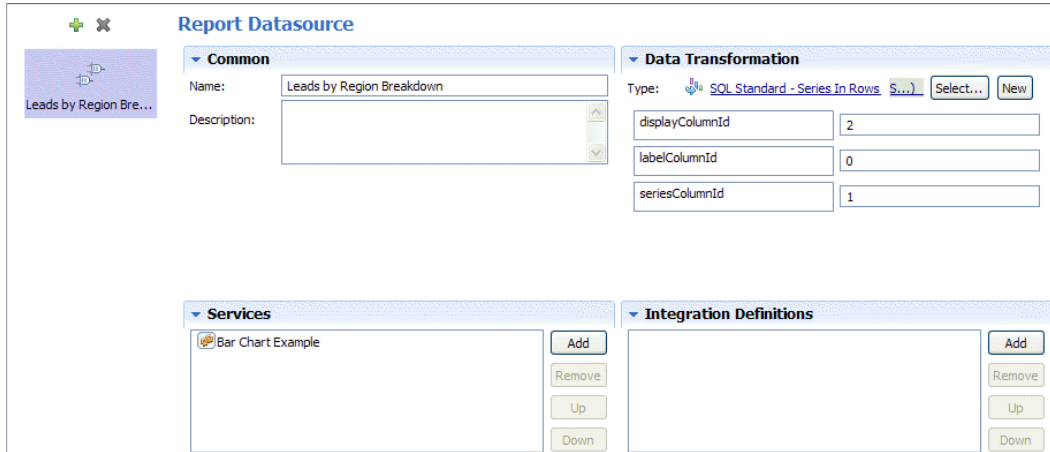
16. Use the Sequence Flow tool to connect the Start component, SQL Query server scriptlet, Execute SQL Integration service, and End component.
17. Save the Integration service so that you can specify it as the data source for the report created in the following section.



You can test the Integration service to ensure that it returns the data that you want by clicking the Debug icon in the upper right corner of the Designer. Click the **Step** button in the debugger until the query results are displayed.

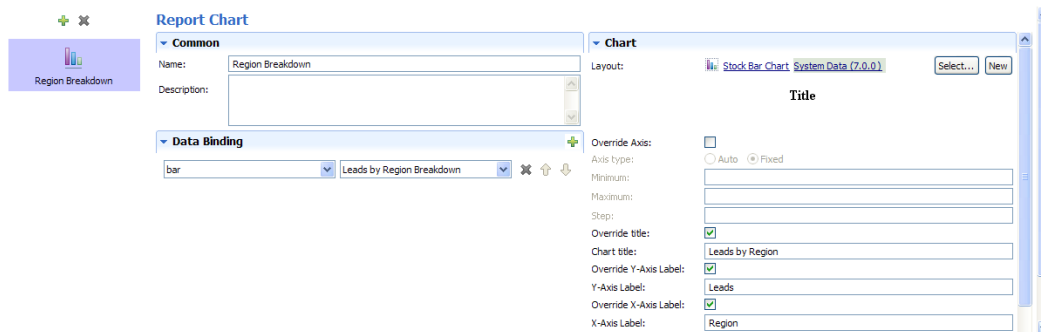
Creating the report

1. To create the report itself, we click the plus sign next to the Performance category in the library and select **Report** from the list of components. In the New Report dialog, we enter a name (`Sample Report - Bar Chart`) in the appropriate field and click **Finish**.
2. In the report that we've created, we click the Datasources tab, click the green plus sign in the upper left corner of the dialog, click the **Add** button under Services, find the `Bar Chart Example` Integration service, click it so that it is added to the report, and then type a name for the datasource in the **Name** field as illustrated in the following image:



As you can see in the preceding image, we select the `SQL Standard - Series in Rows` Data Transformation for our bar chart because we want to display stacked bars that include both hot and cold leads for each region. (A Data Transformation is a script that executes the associated Integration service and adds series to our charts.) To control how the data is displayed, we enter the values shown for `displayColumnId`, `labelColumnId`, and `seriesColumnId`. Later, when we test our report page using the Playback option, we'll show how changing these values changes our report.

- To add a chart to the report, we click the **Charts** tab and then click the green plus sign in the upper left corner of the dialog. We provide a name for the chart under the **Common** heading, and then click the **Select** button next to the **Layout** field under the **Chart** heading to select **Stock Bar Chart** from the **System > Chart Types** folder. To bind data to the chart, we click the green plus sign next to the **Data Binding** heading, select **bar** from the first drop-down menu and **Leads by Region Breakdown** from the second menu as shown in the following image:



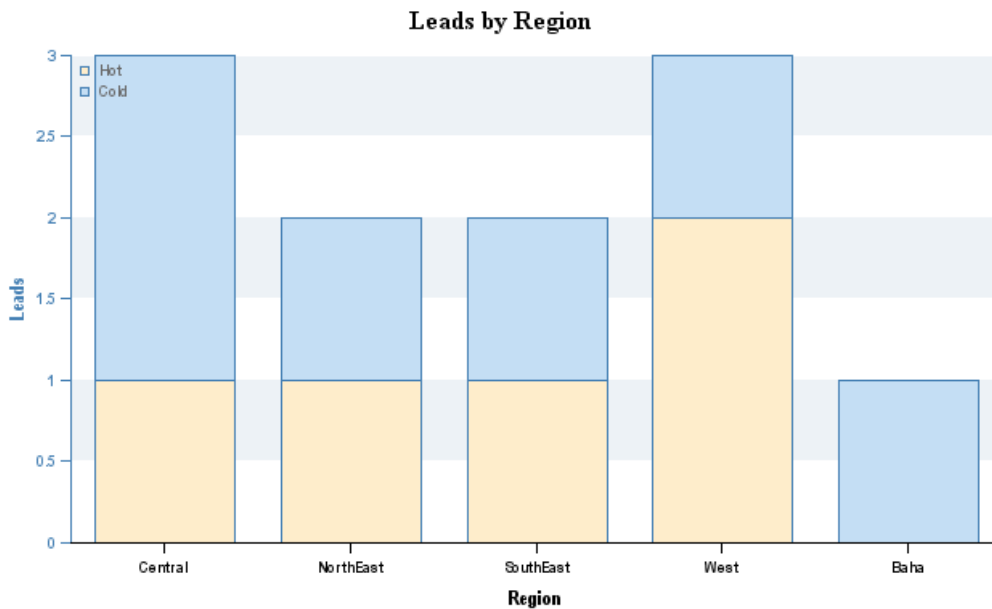
We enable the **Override title** checkbox to override the default chart title and provide our own title of `Leads by Region`. Notice that the same is true for the Y-Axis label and X-Axis label.

- To create a page for the report, we click the **Pages** tab, click the green plus sign in the upper left corner of the dialog, and name the page `Main`. Under **Page Details**, we use the drop-down menu to select the chart to display and under the **Behavior** heading, we click the **Default Page** checkbox and the **Simple Layout** checkbox as shown in the following image:

Report Page

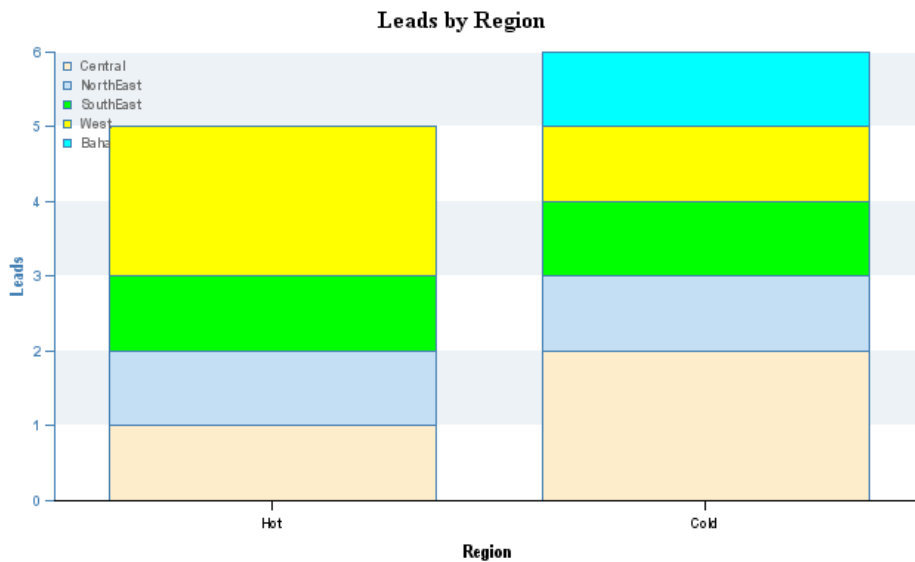
Common Name: <input type="text" value="Main"/> Description: <input type="text"/>		Behavior Default page: <input checked="" type="checkbox"/> Default Width: <input type="text" value="640"/> Default Height: <input type="text" value="400"/> Simple Layout: <input checked="" type="checkbox"/> Playback this Report Page.	
Page Details Chart to display: <input type="text" value="Region Breakdown"/>			
Display data table with chart: <input type="checkbox"/>			

- Click the Save icon on the toolbar to save the report. To test our page, we click **Playback**. Lombardi displays the report page in our browser:



But, what if we want to display the data differently? Suppose we want two bars in our chart: one to show all hot leads and one to show all cold leads? We can alter the final chart by changing the values that we provide for `labelColumnId` and `seriesColumnId` in the Datasources tab of the report. For example, if we change those values as shown in the following image:

When we click **Playback** on the Pages tab, the chart displays as follows:



To understand why changing the column ID values has such an effect, let's take a look at the results of the query in our Integration service. To test and view the results of the query, go to the Integration service and click Debug icon in the upper right corner. Click the **Step** button in the debugger until the query results are displayed as shown in the following image:

results	XMLElement	<pre> <resultSet recordCount="9" columnCount="3"> <record> <column name="district">Central</column> <column name="name">Hot</column> <column name="var">1</column> </record> <record> <column name="district">NorthEast</column> <column name="name">Hot</column> <column name="var">1</column> </record> <record> <column name="district">SouthEast</column> <column name="name">Hot</column> <column name="var">1</column> </record> <record> <column name="district">West</column> <column name="name">Hot</column> <column name="var">2</column> </record> <record> <column name="district">Baha</column> <column name="name">Cold</column> <column name="var">1</column> </record> <record> <column name="district">Central</column> <column name="name">Cold</column> <column name="var">2</column> </record> <record> <column name="district">NorthEast</column> <column name="name">Cold</column> <column name="var">1</column> </record> <record> <column name="district">SouthEast</column> <column name="name">Cold</column> <column name="var">1</column> </record> <record> <column name="district">West</column> <column name="name">Cold</column> <column name="var">1</column> </record> </resultSet> </pre>
----------------	------------	--

There are three columns of returned data as shown in the following example:

[Column 0]	[Column 1]	[Column 2]
Central	Hot	1
NorthEast	Hot	1
SouthEast	Hot	1

So, if we set:

labelColumnId to 0	We get a separate bar for each district, which are the values stored in Column 0.
--------------------	---

seriesColumnId to 1 labelColumnId	The series displayed in each bar is Hot and Cold, which are the values stored in Column 1.
-----------------------------------	--

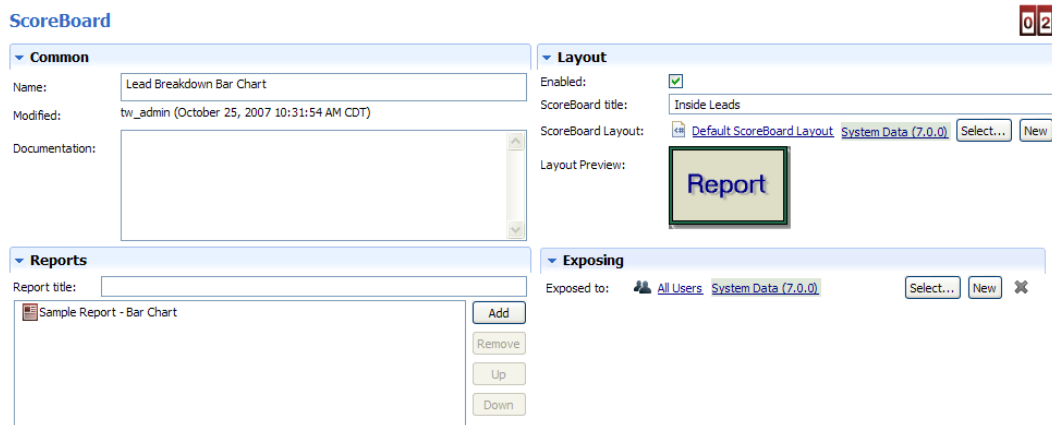
And, if we set:

labelColumnId to 1	We get a separate bar for Hot and Cold, which are the values stored in Column 1.
seriesColumnId to 0	The series displayed in each bar is each district, which are the values stored in Column 0.

Creating the scoreboard

- To display this report in Lombardi Process Portal, we need to create a scoreboard and add our report to that scoreboard. Click the plus sign next to the Performance category in the library and select **Scoreboard** from the list of components. In the New Scoreboard dialog, we name the scoreboard **Lead Breakdown Bar Chart** and leave the default layout setting (Default Scoreboard Layout).

In the scoreboard we've created, under the **Reports** heading, we click the **Add** button, find our report, and click it to add it to the scoreboard. Then, under the **Layout** heading, click the **Enabled** check box and provide a title for the scoreboard. Also click the **Select** button next to the **Exposed to** option and pick the participant group whose members can view this scoreboard in Lombardi Process Portal as shown in the following image:



- Now when we access Lombardi Process Portal, we can see the title of the scoreboard (**Inside Leads**) in My Scoreboards on the left of the portal. When we click **Inside Leads**, Lombardi Process Portal displays the report:



- To control access to the scoreboard in different environments:

Environment	Required configuration
Development (on Process Center Server)	In the Designer in Lombardi Authoring Environment, create a new participant group or adjust the members of the existing participant group to whom the scoreboard is exposed. See Creating a participant group .
Runtime (Process Server in test, staging, or production environment)	After installing a process application snapshot that includes the scoreboard, adjust the members of the participant group to whom the scoreboard is exposed as described in Configuring runtime participant groups .

Adding a filter

Optionally, we can add a filter to our report. For example, let's say that we want to filter our report results by a certain number of days. The following sample shows how to add a filter to a report.



Before you begin, consider taking a snapshot of the process application that contains the Integration service and other items created in the preceding sections since we will be altering some of those items in this section. With snapshots, you can view and run each version you save which ensures that you can access these completed samples when you need them.

- Go to the Lombardi library and open the `Bar Chart Example` Integration service.
- Click the `Variables` tab and add an Input variable named `filters` of type `Map`.

Click the `Has Default` checkbox to enable it and type the following in the text box to initialize the variable: `new tw.object.Map()`

- To add a filter, let's change the SQL statement in the Integration service so that it includes a where clause that limits the hot leads shown to those with a timestamp within the timeframe (number of days) that we specify.

To do so, first we add javascript to define the `leadDays` variable as shown in the following sample:

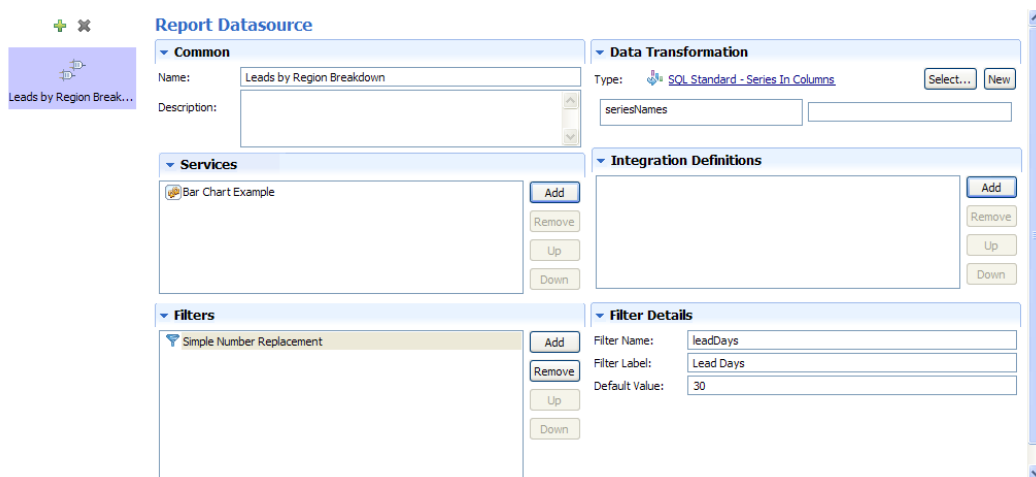
```

<#
var leadDays = tw.local.filters.get("leadDays");
if(leadDays == null) {
leadDays = "30";
}
#>

SELECT
    district, count(*)
FROM
    InsideSalesLeads
WHERE
    temperature = 'Hot'
    and time_stamp_days > (GETDATE() - <#= leadDays #>)
GROUP BY
    district
    
```

The and statement that we added to the where clause limits the results to those entries with time_stamp_days within a particular value as specified in the leadDays variable. The default value of leadDays is 30. In the following steps, we'll specify leadDays as the name of our filter.

4. Open the Sample Report - Bar Chart report and go to the Datasources tab.
5. Change the Data Transformation to SQL Standard - Series in Columns to work with the new query.
6. Under the **Filters** heading, click **Add** and select the Simple Number Replacement filter from the System Data toolkit.
7. Under the **Filter Details** heading, enter leadDays for the Filter Name, Lead Days for the Filter Label, and 30 for the default value as shown in the following image:



8. Click the Save icon on the toolbar.
9. Go to the Overview tab for the report and click **Playback**. Lombardi displays the report page in a browser window:



Notice the Lead Days filter in the upper left. Currently the report is showing the hot leads by region entered within the last 30 days.

- Change the value in the Lead Days filter to 7 and click **Refresh**.



Now the report shows the hot leads entered within the last 7 days. With the filter, users can configure the report to analyze the incoming leads for the specific timeframes that they want.

Creating a third-party report

The following sample shows how to use a third-party tool such as Microsoft Access to create a report.



When querying the Performance Data Warehouse database, you should query the views as described in [Performance Data Warehouse database architecture](#).

Using autotracked data

You can simply enable autotracking, send definitions to the Performance Data Warehouse, run instances of your process, and Lombardi captures data that you can use to run a report like the following sample that compares the duration of each activity in a BPD. That's because autotracking captures entry and exit points for each process component in a BPD.

Using a third-party tool

Because Lombardi stores all performance-related data in the Performance Data Warehouse, you can use third-party tools to query the database and generate reports. The following sample shows how to use Microsoft Access to generate a report from Lombardi data.

In the following sample, our Lombardi Performance Data Warehouse uses a Microsoft SQL Server 2005 database, which we establish a connection to from Microsoft Access 2007. Using MS Access, we'll create a report that shows the average duration for all activities in a particular BPD.

When autotracking is enabled, Lombardi adds a timing interval for each activity in an autotracked BPD. So, if you have autotracking enabled for your BPDs, you can take advantage of the tracked timing intervals using queries and reports.

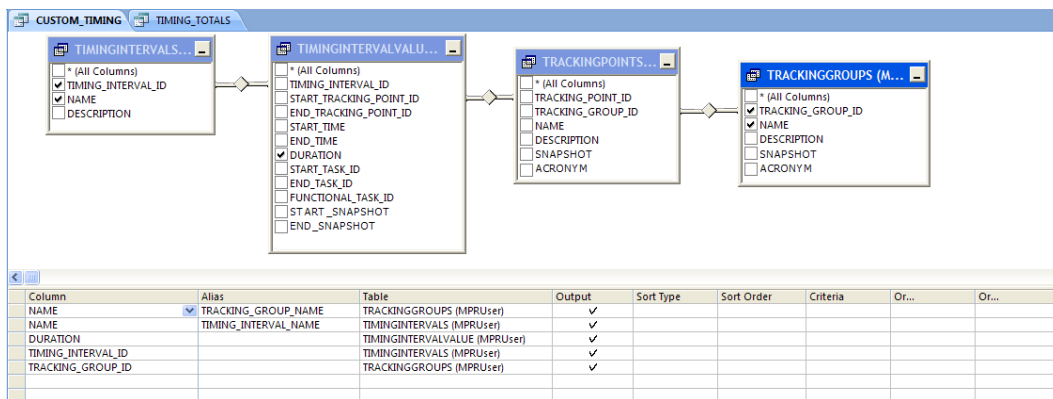
To start, we create a query in MS Access called CUSTOM_TIMING to return the following values:

- Tracking Group Name and ID
- Timing Interval Name, ID, and Duration

From the following views in the Performance Data Warehouse:

- TIMINGINTERVALS
- TIMINGINTERVALVALUE
- TRACKINGPOINTS
- TRACKINGGROUPS

We create the CUSTOM_TIMING query in the MS Access Design view as shown in the following image:



Notice the columns that are used for SQL joins in the preceding Design view:

- TIMING_INTERVAL_ID in TIMING INTERVALS view to TIMING_INTERVAL_ID in TIMINGINTERVALVALUES view
- START_TRACKING_POINT_ID in TIMINGINTERVALVALUES view to TRACKING_POINT_ID in TRACKINGPOINTS view
- TRACKING_GROUP_ID in TRACKINGPOINTS view to TRACKING_GROUP_ID in TRACKINGGROUPS view

The CUSTOM_TIMING query returns the following rows of data as shown in the MS Access Datasheet view:

TRACKING_GROUP_NAME	TIMING_INTERVAL_NAME	DURATION	TIMING_INTERVAL_ID	TRACKING_GROUP_ID
MPRProposalAssemblyAndReview	Hotel Buyer Response (MPR Proposal Assembly and Review)	3028543	1828	51
MPRProposalAssemblyAndReview	Air Buyer Response (MPR Proposal Assembly and Review)	3559448313	1831	51
MPRProposalAssemblyAndReview	Hotel Buyer Response (MPR Proposal Assembly and Review)	1782857	1828	51
MPRProposalAssemblyAndReview	Support Services Response (MPR Proposal Assembly and Review)	3559450297	1835	51
MPRProposalAssemblyAndReview	Hotel Buyer Response (MPR Proposal Assembly and Review)	248115237	1828	51
MPRProposalAssemblyAndReview	Creative Response (MPR Proposal Assembly and Review)	3559449124	1834	51
MPRProposalAssemblyAndReview	Hotel Buyer Response (MPR Proposal Assembly and Review)	1292403893	1828	51
MPRProposalAssemblyAndReview	Engage Support Groups (MPR Proposal Assembly and Review)	75685437	1818	51
MPRProposalAssemblyAndReview	Hotel Buyer Response (MPR Proposal Assembly and Review)	36564	1828	51
MPRProposalAssemblyAndReview	Untitled7 (MPR Proposal Assembly and Review)	0	1812	51
MPRProposalAssemblyAndReview	Develop Budget (MPR Proposal Assembly and Review)	3635133250	1817	51
MPRProposalAssemblyAndReview	Hotel Buyer Response (MPR Proposal Assembly and Review)	28857	1828	51

The returned columns contain the following data:

- TRACKING_GROUP_NAME: The name assigned to the autotracking group in Lombardi Authoring Environment.
- TIMING_INTERVAL_NAME: Lombardi names each timing interval in an autotracking group as follows: *activity_name (bpd_name)*.
- DURATION: The amount of time that it took the timing interval to complete in milliseconds.
- TIMING_INTERVAL_ID: The ID assigned to the timing interval in the Performance Data Warehouse by Lombardi.
- TRACKING_GROUP_ID: The ID assigned to the tracking group in the Performance Data Warehouse by Lombardi.

Now we create a second query called TIMING_TOTALS to perform the following functions on the results of the first query:

- Group the data by timing interval name and filter the results to limit them to the BPD named *MPR Program Planning*; this is possible because Lombardi names the timing intervals: *activity_name (bpd_name)*
- Add up the total duration times (seconds, minutes, and hours)
- Average the total duration times (seconds, minutes, and hours)

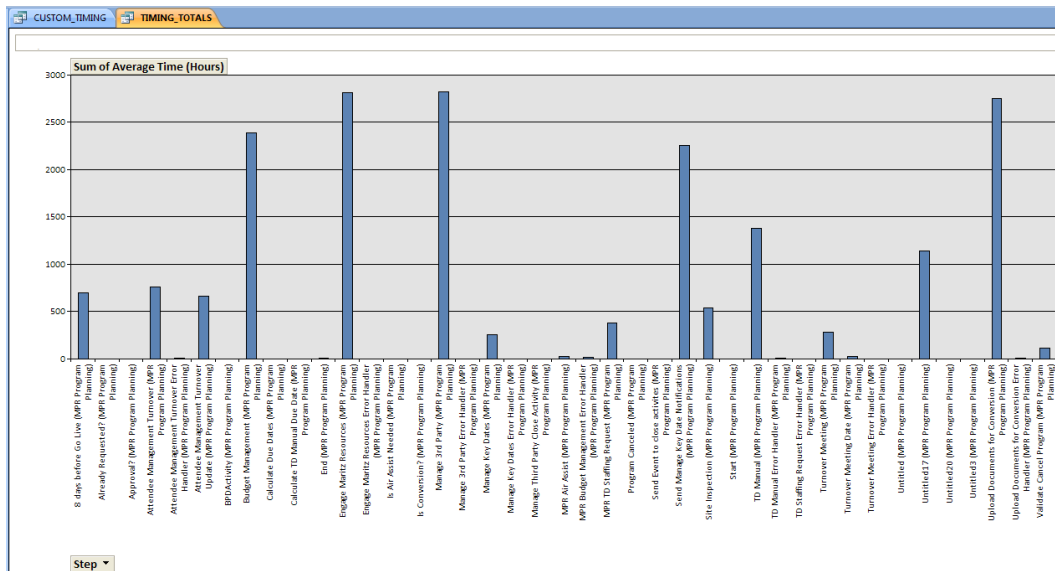
We create the TIMING_TOTALS query in the MS Access Design view as shown in the following image:

Column	Alias	Table	Output	Sort Type	Sort Order	Group By	Criteria
TIMING_INTERVAL_NAME	Step	CUSTOM_TIMING	✓			Group By	LIKE N% (MPR Program Planning)
DURATION / 1000	[Total Time (Seconds)]		✓			Sum	
DURATION / 60000	[Total Time (Minutes)]		✓			Sum	
DURATION / 3600000	[Total Time (Hours)]		✓			Sum	
DURATION / 1000	[Average Time (Seconds)]		✓			Avg	
DURATION / 60000	[Average Time (Minutes)]		✓			Avg	
DURATION / 3600000	[Average Time (Hours)]		✓			Avg	

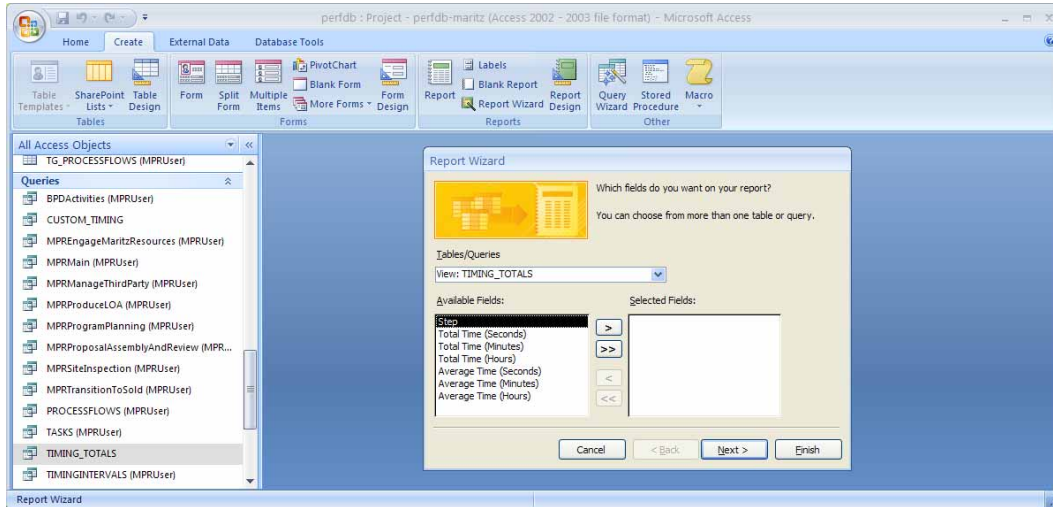
TheTIMING_TOTALS query returns the following rows of data as shown in the MS Access Datasheet view:

Step	Total Time (Seconds)	Total Time (Minutes)	Total Time (Hours)	Average Time (Seconds)	Average Time (Minutes)	Average Time (Hour)
8 days before Go Live (MPR Program Planning)	835500603.741000	13925010.062240	232083.500893	2516568.083557	41942.801392	699.046689
Already Requested? (MPR Program Planning)	1.310000	.021795	.000311	.002467	.000041	.000000
Approval? (MPR Program Planning)	.013000	.000216	.000003	.000333	.000005	.000000
Attendee Management Turnover (MPR Program Planning)	1082435323.847000	18040588.730647	300676.478664	2754288.355844	45904.805930	765.080098
Attendee Management Turnover Error Handler (MPR Program Planning)	2374978.800000	39582.979977	659.716298	34419.982608	573.666376	9.561105
Attendee Management Turnover Update (MPR Program Planning)	298023482.900000	4967058.048387	82784.300750	2384187.863248	39736.464387	662.274406
BPDActivity (MPR Program Planning)	13.186000	.219605	.003424	.022617	.000376	.000005
Budget Management (MPR Program Planning)	2846254809.014000	47437580.150127	790626.335684	8598957.126930	143315.952115	2388.599201
Calculate Due Dates (MPR Program Planning)	833854.866000	13897.580899	231.626098	1532.821444	25.547023	425.783
Calculate TD Manual Due Date (MPR Program Planning)	1.536000	.025559	.000367	.004219	.000070	.000001
End (MPR Program Planning)	145432076.061000	2423867.934086	40397.798539	44652.157218	744.202620	12.403376
Engage Maritz Resources (MPR Program Planning)	2560536148.238000	42675602.470546	711260.041053	10120696.238094	168678.270634	2811.304510

With MS Access, we can switch to the PivotChart view to see these query results in a bar chart as shown in the following image:



We can also use the Report wizard in MS Access to create a report suitable for printing and distribution. For example, if we click theTIMING_TOTALS query and click Report Wizard in the Create tab, the wizard guides us through report creation as shown in the following image:



With the report wizard, we can preview and alter the final report as required:

TIMING_TOTALS

Step	Total Time (Hours)	Average Time (Hours)
8 days before Go Live	232083.500893	699.046689
Already Requested?	.000311	.000000
Approval?	.000003	.000000
Attendee Management Turnover	300676.478664	765.080098
Attendee Management Turnover Error H	659.716298	9.561105
Attendee Management Turnover Updat	82784.300750	662.274406
BPDActivity	.003424	.000005
Budget Management	790626.335684	2388.599201
Calculate Due Dates	231.626098	.425783
Calculate TD Manual Due Date	.000367	.000001

As the preceding sample illustrates, you can use the third-party tools available in your environment to query the Performance Data Warehouse to analyze and report on your Lombardi business processes. If you create tracking groups to track the variables specific to your processes and you enable autotracking, the data available in the Performance Data Warehouse can provide informative analysis of all your process initiatives.

Performance Data Warehouse database architecture

Using the Tracking Group tables, the Performance Data Warehouse creates views to make the tracked data externally available for both Lombardi and third-party reports. When you query the Performance Data Warehouse database, you should query the Tracking Group and other views outlined in the following sections.

Tracking Group views

Tracking Group views contain the same data as the Tracking Group tables, which includes a column for each tracked variable and several columns to capture timing and other important task information. The Performance Data Warehouse:

- Creates a view for each Tracking Group defined in the Authoring Environment.
- Gives each view the same name as the corresponding Tracking Group in the Authoring Environment. (When you use one tracking group in multiple process applications, all data is stored in a single view, with an ID to distinguish each implementation.)

The Tracking Group tables in the Performance Data Warehouse database differ from the views in that they contain all tracked data, including data for those variables that you may have stopped tracking during the lifetime of your process. Only the views are updated when you make such changes to your tracking requirements.

The following general rules apply to how Performance Data Warehouse manages changes to an existing Tracking Group:

- When a Tracking Group is renamed (in Lombardi Authoring Environment), the Tracking Group view is renamed, but the name of the Tracking Group table does not change.
- When a Tracking Group is deleted, the view is deleted and the Tracking Group in the table is marked as deleted, but the table does not change.
- When a new field is added to an existing Tracking Group, a field of the same name is added to the view whose name matches the name of the Tracking Group. Additionally, a new column with a unique name that is similar (but not necessarily identical) to the name of the field is added to the table.
- When a field is deleted from a Tracking Group (in Lombardi Authoring Environment), the field remains in both the table and the view. If a snapshot that contains particular fields in a Tracking Group is archived using the Performance Data Warehouse command-line tool (`perfsvrtool`), those fields are removed from the view and marked as deleted. If the snapshot is restored, those fields are added back to the view.
- When a field is renamed, both the old name and the new name are present in the view and the table.
- When the type of a field (string, number, or date/time) is changed, a new column with a unique name that is similar (but not necessarily identical) to the name of the field in the Tracking Group is added to the table, and the view is updated to use the new column. The formerly used column does not change. You must manually migrate the data for the modified field to the new column. This is also true for type changes from one version (snapshot) to another.



When you send tracking definitions from Lombardi Authoring Environment or as part of snapshot installation on a runtime server, Performance Data Warehouse performs a comparison between the updated definitions and the definitions in its tables, and performs the appropriate update action for each change. Unchanged data is not re-sent.

Tracking Group views include the following columns:

Column	Description
<i>business data columns</i>	One column for each variable in the corresponding Tracking Group. Each column has the same name as the corresponding tracked field.

Column	Description
TRACKING_GROUP_ID	The primary key of the Tracking Group to which this entry corresponds. This column links to the TRACKINGGROUPS view.
TRACKING_POINT_ID	The primary key of the Tracking Point to which this entry corresponds. This column links to the TRACKINGPOINTS view.
TASK_ID	The primary key in the Performance Data Warehouse of the task to which this entry corresponds. This column links to the TASKS view.
FUNCTIONAL_TASK_ID	The primary key in the Performance Data Warehouse of the BPD instance to which this entry corresponds. This column links to the TASKS view.
SNAPSHOT	The snapshot (version) of the process application or toolkit to which this entry corresponds. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit to which this entry corresponds.
TIME_STAMP	The date and time at which the corresponding tracking point was traversed.
TIME_STAMP_DAYS	TIME_STAMP truncated to days.
TIME_STAMP_WEEKS	TIME_STAMP truncated to weeks.
TIME_STAMP_MONTHS	TIME_STAMP truncated to months.
TIME_STAMP_QUARTERS	TIME_STAMP truncated to quarters.
TIME_STAMP_YEARS	TIME_STAMP truncated to years.

SNAPSHOTS view

This view contains one row for each snapshot defined in the Process Center Console.

The SNAPSHOTS view includes the following columns:

Column	Description
SNAPSHOT_ID	The primary key of this snapshot in the Performance Data Warehouse.
NAME	The name assigned to this snapshot in the Process Center Console.
DESCRIPTION	The description given to this snapshot, if one was provided, in the Process Center Console.
ACRONYM	The acronym designated for the process application or toolkit that contains this snapshot.

TASKS view

This view contains one row for each task that is executed and an additional row for each BPD instance that is started.

The TASKS view includes the following columns:

Column	Description
TASK_ID	The primary key of this task in the Performance Data Warehouse.
FUNCTIONAL_TASK_ID	The primary key in the Performance Data Warehouse of the BPD to which this entry corresponds. In the case where this row corresponds to the beginning of a BPD instance, it will equal TASK_ID.
CREATION_TIME	The date and time that the task was created.
START_TIME	The date and time that the task was started. If CREATION_TIME and START_TIME are not equal, the lag indicates that the task was not immediately started by the user.

Column	Description
END_TIME	The date and time that the task was finished.
SYSTEM_USER_ID	The primary key of the user in the Process Data Warehouse that executed this task.
USERNAME	The name of the user who executed this task.
BPDNAME	The name of the BPD that triggered this task.
STARTING_PROCESS_ID	The primary key in the Process Data Warehouse of the BPD instance that created this task.
ACTIVITY_NAME	The name of the Activity that triggered this task.
SYSTEM_TASK_ID	The primary key in the Process Data Warehouse of the task. This entry includes a trailing a if the task is an activity.
SYSTEM_FUNCTIONAL_TASK_ID	The primary key in the Process Data Warehouse of the BPD instance.
SNAPSHOT	The snapshot (version) of the process application or toolkit that triggered this task. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit that triggered this task.

TRACKINGGROUPS view

This view contains one row for each tracking group defined in Lombardi.

The TRACKINGGROUPS view includes the following columns:

Column	Description
TRACKING_GROUP_ID	The primary key of the tracking group.
NAME	The name assigned to the tracking group in the Authoring Environment.
DESCRIPTION	The description of the tracking group if one was provided in the Authoring Environment.
SNAPSHOT	The snapshot (version) of the process application or toolkit that contains the Tracking Group. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit that contains the Tracking Group.

TIMINGINTERVALS view

This view contains one row for every timing interval defined in the Authoring Environment. When autotracking is enabled, Lombardi adds a timing interval for every activity in an autotracked BPD.

The TIMINGINTERVALS view includes the following columns:

Column	Description
TIMING_INTERVAL_ID	The primary key of this timing interval.
NAME	The name assigned to the timing interval in the Authoring Environment. For autotracking timing intervals, this name corresponds to the name of the activity.
DESCRIPTION	The description of the timing interval (if one was provided in the Authoring Environment).

TIMINGINTERVALVALUE view

A row is recorded in this view each time a timing interval is traversed. When autotracking is enabled, a row is recorded each time an autotracked task is executed.

The TIMINGINTERVALVALUE view includes the following columns:

Column	Description
TIMING_INTERVAL_ID	The primary key of the timing interval. This column links to the TIMINGINTERVALS view.
START_TRACKING_POINT_ID	The primary key of the tracking point that marks the beginning of the timing interval. This column links to the TRACKINGPOINTS view.
END_TRACKING_POINT_ID	The primary key of the tracking point that marks the end of the timing interval. This column links to the TRACKINGPOINTS view.
START_TIME	The date and time that the timing interval started.
END_TIME	The date and time that the timing interval ended.
DURATION	The duration of the timing interval in milliseconds.
START_TASK_ID	The primary key in the Performance Data Warehouse of the task where this timing interval began. This column links to the TASKS view.
END_TASK_ID	The primary key in the Performance Data Warehouse of the task where this timing interval ended. This column links to the TASKS view.
FUNCTIONAL_TASK_ID	The primary key in the Performance Data Warehouse of the BPD instance that started this timing interval. This column links to the TASKS view.
START_SNAPSHOT	The snapshot (version) of the tracking point that marks the beginning of the timing interval. If no snapshots exist, a Null value is stored in this column.
END_SNAPSHOT	The snapshot (version) of the tracking point that marks the end of the timing interval. If no snapshots exist, a Null value is stored in this column.

TRACKEDFIELDS view

This view contains one row for every tracked field defined in the Authoring Environment. The TRACKEDFIELDS view includes the following columns:

Column	Description
TRACKED_FIELD_ID	The primary key of the tracked field.
TRACKING_GROUP_ID	The primary key of the tracking group to which this tracked field belongs. This column links to the TRACKINGGROUPS view.
NAME	The name assigned to the tracked field in the Authoring Environment. When autotracking is enabled, this entry corresponds to the name of the associated activity.
DESCRIPTION	The description of the tracked field (if one was provided in the Authoring Environment).
FIELD_TYPE	The type of the tracked field as follows: 0 = string, 1 = number, 2 = date.
SNAPSHOT	The snapshot (version) of the process application or toolkit that contains the tracked field. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit that contains the tracked field.

TRACKEDFIELDUSE view

A row is recorded in this view when a tracked field from a tracking group is used by a specific tracking point. The TRACKEDFIELDUSE view includes the following columns:

Column	Description
TRACKED_FIELD_USE_ID	The unique identifier (primary key) for this use of the tracked field.

Column	Description
TRACKING_GROUP_ID	The primary key of the tracking group to which this tracked field belongs. This column links to the TRACKINGGROUPS view.
TRACKING_POINT_ID	The primary key of the tracking point that uses this tracked field. This column links to the TRACKINGPOINTS view.
TRACKED_FIELD_ID	The primary key of the tracked field. This column links to the TRACKEDFIELDS view.
SNAPSHOT	The snapshot (version) of the tracking group to which this tracked field belongs. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit that contains the tracking group to which this tracked field belongs.

TRACKINGPOINTS view

This view contains one row for every tracking point defined in the Authoring Environment. When autotracking is enabled, two tracking points are created for each autotracked activity in order to mark the beginning and end of a task.

The TRACKINGPOINTS view includes the following columns:

Column	Description
TRACKING_POINT_ID	The primary key of the tracking point.
TRACKING_GROUP_ID	The primary key of the tracking group to which this tracking point belongs. This column links to the TRACKINGGROUPS view.
NAME	The name assigned to the tracking point in the Authoring Environment. When autotracking is enabled, this entry corresponds to the name of the associated activity.
DESCRIPTION	The description of the tracking point (if one was provided in the Authoring Environment).
SNAPSHOT	The snapshot (version) of the process application or toolkit that contains the tracking point.
ACRONYM	The acronym of the process application or toolkit that contains the tracking point.

TRACKINGPOINTVALUE view

A row is recorded in this view each time a tracking point is traversed. When autotracking is enabled, there will be two rows for every task that is executed, because autotracking defines start and end tracking points for each autotracked activity.

The TRACKINGPOINTVALUE view includes the following columns:

Column	Description
TRACKING_POINT_ID	The primary key of the tracking point. This column links to the TRACKINGPOINTS view.
TASK_ID	The primary key in the Performance Data Warehouse of the task that corresponds to this tracking point. This column links to the TASKS view.
FUNCTIONAL_TASK_ID	The primary key in the Performance Data Warehouse of the BPD instance that corresponds to this tracking point. This column links to the TASKS view.
TIME_STAMP	The date and time at which the tracking point was traversed.

Column	Description
SNAPSHOT	The snapshot (version) of the task that corresponds to this tracking point. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit that contains the task that corresponds to this tracking point.
TIME_STAMP_DAYS	TIME_STAMP truncated to days.
TIME_STAMP_WEEKS	TIME_STAMP truncated to weeks.
TIME_STAMP_MONTHS	TIME_STAMP truncated to months.
TIME_STAMP_QUARTERS	TIME_STAMP truncated to quarters.
TIME_STAMP_YEARS	TIME_STAMP truncated to years.

PROCESSFLOWS view

The PROCESSFLOWS view is an implicit tracking group view that captures the following values each time that a line is traversed in a BPD. This view is used primarily for the flow-traversal visualization modes in the Optimizer.

The PROCESSFLOWS view includes the following columns:

Column	Description
BPD_ID	The unique identifier for this BPD.
BPD_INSTANCE_ID	The instance ID of the BPD that is traversing the line.
LOCAL_SNAPSHOT_ID	The primary key of the snapshot of the BPD that is traversing the line. This column links to the SNAPSHOTS view.
SEQUENCE_FLOW_ID	The external unique ID of this line in the BPD.
SOURCE_EUID	The external unique ID of the POST tracking group of the preceding step in the BPD, which is used to match up this flow-traversal with the flow object that it came from.
STEP_NUMBER	The current STEP_NUMBER of this BPD instance. STEP_NUMBER is incremented when traversing each flow object's PRE and POST tracking points. Since STEP_NUMBER is not incremented when traversing lines, there should be a tracking point value (in the tracking group specific to the BPD) with a matching STEP_NUMBER and BPD_INSTANCE_ID, whose external unique ID matches SOURCE_EUID.
TRACKING_GROUP_ID	The primary key of the tracking group specific to the BPD. This column links to the TRACKINGGROUPS view.
TRACKING_POINT_ID	The primary key of the tracking point for this flow-traversal. This column links to the TRACKINGPOINTS view.
TASK_ID	The primary key of the task that corresponds to this traversal. This column links to the TASKS view.
FUNCTIONAL_TASK_ID	The primary key in the Performance Data Warehouse of the BPD instance that corresponds to this traversal. This column links to the TASKS view.
TIME_STAMP	The date and time of this traversal.
SNAPSHOT	The snapshot (version) of the task that corresponds to this traversal. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit that contains the task that corresponds to this traversal.
TIME_STAMP_DAYS	TIME_STAMP truncated to days.
TIME_STAMP_WEEKS	TIME_STAMP truncated to weeks.

Column	Description
TIME_STAMP_MONTHS	TIME_STAMP truncated to months.
TIME_STAMP_QUARTERS	TIME_STAMP truncated to quarters.
TIME_STAMP_YEARS	TIME_STAMP truncated to years.

SLASTATUS view

The SLASTATUS view is an implicit tracking group view that tracks the status of each SLA as a result of an SLA roll-up. SLA roll-ups occur periodically to calculate the current state of SLAs based on information in the SLATHRESHOLDTRAVERSALS view (described in the following section). Roll-ups are necessary because many SLA conditions are based on ranges such as *the last N days* or *in the last 2 weeks*. Lombardi tracks a new value any time any of the following values change.

The SLASTATUS view includes the following columns:

Column	Description
INTERIM	This is either Υ or \aleph . An SLA status value is interim if it comes from the middle of a condition other than a range. A range is something like <i>the last 5 occurrences</i> or <i>in the last 3 weeks</i> . A non-range condition is something like <i>today</i> . If you have an SLA based on the sum of some metric <i>today</i> , then you want to be able to see the interim values during the day, however when plotting a graph you'd really want to see only the sum at the end of the day. SLAs with non-range conditions always track a non-interim value at the end of the window.
SLA_VALUE	The current value of the SLA. This value is only meaningful for SLAs with conditions like <i>SUM(X) over today</i> , in which case the value will hold that sum. In most other cases, this column holds the METRIC_VALUE as of the last traversal.
VIOLATION_LEVEL	The violation level of the SLA as follows: a value between 0.0 and 1.0 = unviolated; a value greater than or equal to 1.0 = violated. Wherever possible, the violation level is set to a value above 1.0 to indicate how violated the SLA is. For example, if an SLA measures whether 3 of the last 10 conditions were violated, and 6 of the last 10 conditions were violated, then the violation level would be 2.0.
TRACKING_GROUP_ID	The primary key of the Tracking Group to which this entry corresponds. This column links to the TRACKINGGROUPS view.
TRACKING_POINT_ID	The primary key of the Tracking Point to which this entry corresponds. This column links to the TRACKINGPOINTS view.
TASK_ID	The primary key of the task to which this entry corresponds. This column links to the TASKS view.
FUNCTIONAL_TASK_ID	The primary key of the BPD instance to which this entry corresponds. This column links to the TASKS view.
TIME_STAMP	The date and time at which the corresponding tracking point was traversed.
SNAPSHOT	The snapshot (version) of the process application or toolkit to which this entry corresponds. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit to which this entry corresponds.
TIME_STAMP_DAYS	TIME_STAMP truncated to days.
TIME_STAMP_WEEKS	TIME_STAMP truncated to weeks.
TIME_STAMP_MONTHS	TIME_STAMP truncated to months.
TIME_STAMP_QUARTERS	TIME_STAMP truncated to quarters.
TIME_STAMP_YEARS	TIME_STAMP truncated to years.

SLATHRESHOLDTRAVERSALS view

The SLATHRESHOLDTRAVERSALS view is an implicit tracking group view that tracks a value whenever an activity with an attached SLA completes. An activity has an attached SLA if there is an SLA with a condition dependent on a metric of that activity.

The SLATHRESHOLDTRAVERSALS view includes the following columns:

Column	Description
METRIC_GUID	The GUID of the metric referenced in the SLA.
METRIC_VALUE	The value of the metric referenced in the SLA.
TARGET_VALUE	The value that the metric is compared to in the SLA.
TRACKING_POINT_GUID	The GUID of the tracking point corresponding to where this traversal occurred, which is the end autotracking point for the activity.
VIOLATION_LEVEL	An indication of how much of a violation this particular occurrence is. This value is only relevant and, thus, populated for <i>greater than</i> SLA conditions, and is basically METRIC_VALUE/TARGET_VALUE.
TRACKING_GROUP_ID	The primary key of the Tracking Group to which this entry corresponds. This column links to the TRACKINGGROUPS view.
TRACKING_POINT_ID	The primary key of the Tracking Point to which this entry corresponds. This column links to the TRACKINGPOINTS view.
TASK_ID	The primary key of the task to which this entry corresponds. This column links to the TASKS view.
FUNCTIONAL_TASK_ID	The primary key of the BPD instance to which this entry corresponds. This column links to the TASKS view.
TIME_STAMP	The date and time at which the corresponding tracking point was traversed.
SNAPSHOT	The snapshot (version) of the process application or toolkit to which this entry corresponds. If no snapshots exist, a Null value is stored in this column.
ACRONYM	The acronym of the process application or toolkit to which this entry corresponds.
TIME_STAMP_DAYS	TIME_STAMP truncated to days.
TIME_STAMP_WEEKS	TIME_STAMP truncated to weeks.
TIME_STAMP_MONTHS	TIME_STAMP truncated to months.
TIME_STAMP_QUARTERS	TIME_STAMP truncated to quarters.
TIME_STAMP_YEARS	TIME_STAMP truncated to years.

Simulating and optimizing processes

Lombardi Optimizer enables you to:

- Simulate your processes while you're developing them to understand how well those process models might perform.

Lombardi Optimizer runs simulations using estimates that you provide for staffing levels, activity execution times, and so on. Simulating your processes during development enables you to test and refine process designs before implementation.

- Analyze your processes after they're up and running using historical data stored in the Performance Data Warehouse.

For each process, with autotracking enabled, you can measure actual execution, wait, and other times. You can also track the values of specific business data (process variables) as they move through each step in a process. Running historical analyses using Lombardi Optimizer enables you to measure and then improve the efficiency of your processes.

The Optimizer is a tool designed to help you understand and refine the process models that you develop in Lombardi. It does not analyze the hardware or other systems involved in running your processes. For example, if you suspect that issues identified by Lombardi Optimizer might be due to the performance of hardware or other systems, you can use tools from those vendors to investigate further.

The Optimizer provides a variety of analysis scenarios, ranging from simple simulations to validate your overall process modeling strategy to advanced what-if comparative analyses.

Lombardi Optimizer enables you to...	Benefit
Simulate process performance	Understand process design issues that could affect performance before process implementation
Identify bottlenecks and other issues	Optimize processes already in production
Compare actual process performance to simulations	Analyze how well your processes are doing compared to the goals that you set
Compare simulations to historical performance data	Analyze what would happen if you made specific changes to your processes
Simultaneously analyze multiple processes from a single or multiple process applications	<ul style="list-style-type: none">• Identify resources that are over or under-utilized across processes and applications• Compare performance from month to month or quarter to quarter for specific sets of processes• Experiment with the performance of multiple processes by simulating the addition of resources to one or more participant groups and finding the best results across processes and workloads

To take full advantage of the Lombardi Optimizer, you need to complete the tasks described in [Configuration requirements for simulation](#) and [Configuration requirements for optimization](#).

Configuration requirements for simulation

If you are developing process models and want to perform simulations using Lombardi Optimizer, you must complete the following tasks in the order shown.



You can quickly run a simulation for a single process using default simulation values. To do so, open the process in Lombardi Authoring Environment and select **Playback > Simulate (Single) Process** from the main menu.

Task	Description	See
Set up simulation profiles	For each item in a process model, provide estimates for task duration, probabilities for gateways, and other values on which to base your simulations. Lombardi provides a default simulation profile that you can use or you can create one or more new profiles. The advantage of profiles is that they let you specify and save different estimates for specific situations that you know might occur in your environment.	Setting up simulation profiles
Set Participant Group simulation properties	For each Participant Group, provide estimated capacity, availability, efficiency, and cost per hour. Lombardi provides a default capacity and cost per hour for each Participant Group, but you should adjust these settings before running simulations to reflect the workload in your environment.	Setting simulation properties for participant groups
Create simulation analysis scenarios	Specify the process models to include, the simulation properties and values (simulation profile) to use for each included process, the number of instances to simulate, and other values. The advantage of simulation scenarios is that they enable you to group and compare performance for different sets of processes.	Creating simulation analysis scenarios

Setting up simulation profiles

When you run a simulation, the results are based on the settings that you establish in a Simulation Profile. The following procedure explains how to edit or create a Simulation Profile.



You do not need to create a Simulation Profile to run a simulation. Lombardi includes a Default Simulation Profile that provides default simulation values.

Complete the following steps to set up a Simulation Profile:

1. Open a process (BPD) in the Designer in Lombardi Authoring Environment and click the Diagram tab.

2. Click the **Start** event in the diagram to select it.
3. Click the **Simulation** option In the properties.
4. Under Simulation Profile, click the **New** button to create a new profile or click the **Select** button to select an existing profile.

The simulation values that you provide for each item in your process are saved with the Simulation Profile that is currently selected.



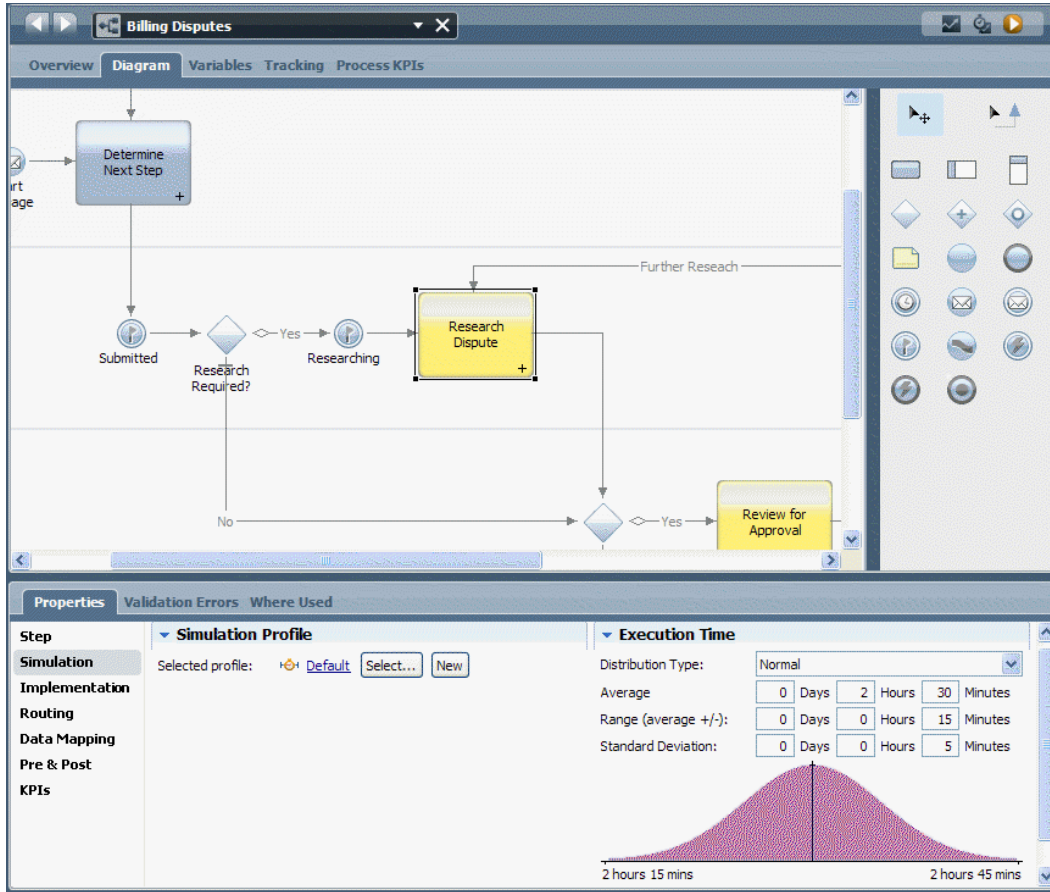
When you select a Simulation Profile for one item in a process, that profile automatically becomes the selected profile for all other items in the process.

5. Make sure the checkbox for **Event is simulated in this profile** is selected.
6. Under **Firing Delay**, set a Distribution type and then establish how often the process is initiated for simulation purposes.

For example, in the following image, the distribution type is **Fixed** and the set value is **10** minutes. This means that the Optimizer will simulate the process starting exactly every 10 minutes. With a Uniform or Normal distribution type, you can establish averages and ranges so that the timing of the process kick-off deviates and is not so precise, which may more accurately reflect conditions in your environment.



7. Click an Activity in the diagram to select it.
8. Click the **Simulation** option In the properties.



- Under **Execution Time**, for **Distribution Type**, select **Fixed**, **Uniform**, or **Normal**.

If you select...	Specify...	Lombardi Optimizer...
Fixed	The execution time in days, hours, and minutes.	Uses the same specified value each time.
Uniform	The average execution time and the range (average plus or minus the values that you specify) in days, hours, and minutes.	Is equally likely to use each value in the specified range.
Normal	The average execution time, the range (average plus or minus the values that you specify), and the standard deviation in days, hours, and minutes.	Is more likely to use values within the specified range that are closer to the specified average instead of values that are more or less than the average.

In the preceding image, the user has changed the Default Simulation Profile for the Research Dispute activity so that the average execution time is 2 hours and 30 minutes, the range is 15 minutes, and the standard deviation is 5 minutes. The graph at the bottom of the profile changes to reflect these values.



Be sure to select and set simulation properties for each activity in the process. If the implementation for an activity is a nested process, you can enable the **Simulate nested process** checkbox and then choose the simulation profile to use for the nested process. If you do not want to include the nested process in the simulation, do not enable the checkbox.

- For each event in the process, indicate whether to simulate the event and, if so, specify the firing delay.

The firing delay tells the Optimizer when to initiate the event. For example, a fixed firing delay of 15 minutes for an attached message event means that the Optimizer simulates the process as if that message event fires 15 minutes after the start of the associated activity and, if the event is repeatable and as long as the associated activity is not closed, every 15 minutes after the initial firing.

Events attached to an activity include a **Firing condition** option so that you can choose between a timed delay and a delay that depends on the percentage of completed activities. For example, the firing condition in the following image occurs when 25% of the activities to which the event is attached are complete:

- For each decision Gateway, Split, and Join, indicate the probability of the run-time process flowing from the gateway in one direction rather than another. The probability is expressed in percentage for each attached sequence line.

You can create multiple Simulation Profiles for a single process. Then, when you create or edit a Simulation Analysis Scenario, you can choose which of the Simulation Profiles to use for the current scenario.



If you run a simulation for a single BPD by choosing **Playback > Simulate (Single) Process** from the main menu, Lombardi uses the currently selected Simulation Profile in the Simulation properties. If no profile has been explicitly set, Lombardi uses the Default Simulation Profile.

Setting simulation properties for participant groups

Before running simulations, you should set the simulation properties for the participant groups assigned to the swimlanes in the process models that you are analyzing. Doing so ensures that the simulation results reflect the performance expectations for the groups in your organization.

Complete the following steps to set simulation properties for participant groups:

1. In the Designer in Lombardi Authoring Environment, open a participant group involved in the processes that you want to simulate.
2. Under Simulation Properties, provide the following information:

Field or control	Description
Capacity	Use the drop-down list to choose either Use Estimated Capacity or Use Provider Users . If you select Use Estimated Capacity , enter the maximum number of users that this group can include in the associated field. If you select Use Provider Users , Lombardi sets the capacity so that it is equal to the number of members in the participant group.
Availability	Specify the percentage of this group's working hours that are available to complete Lombardi tasks resulting from the processes you are analyzing.
Efficiency	Specify the efficiency of this group as a percentage.
Cost per Hour	Provide the cost (in dollars and cents) to your organization for each hour of work performed by this group.



3. Click **Save** in the main toolbar.

Creating simulation analysis scenarios

You can create simulation analysis scenarios and store them in the Lombardi library. When you define a simulation analysis scenario, you provide information that the Optimizer requires such as the processes to include in the simulation, the simulation properties and values (simulation profile) for each process, and so on.

1. In the Designer in Lombardi Authoring Environment, click the plus sign next to Processes and select **Simulation Analysis Scenario** from the list of components.
2. Enter a name for the scenario and click **Finish**.
3. In the Scenarios editor, provide the following information:

Dialog area	Field or control	Description
Common	Documentation	Optionally provide a description in this field.
Simulation Data Filters	Start Time	Use the calendar and clock counter to indicate a start time for the scenario.
	Limit running time	Click this option if you want to limit the simulated running time for the processes included in the analysis. If so, provide the running time in days, hours, and minutes.
	Limit process instances	Click this option if you want to limit the number of process instances that the simulation runs. If so, select the process and then provide the number of instances.
Processes Apps to Include in Analysis	Processes Apps to Include in Analysis	Click the Add button to choose the process applications that you want from the Lombardi repository. (Lombardi lists the process applications to which you have Read access.) Choose the process applications that contain the processes that you want to analyze. Be sure to select the correct snapshot (version) to analyze. Select (Current) to analyze the current working version. To remove a process application, click the application name and then click the Remove button.

Dialog area	Field or control	Description
		 <p>If you select multiple snapshots (versions) to analyze, the first version listed in the table determines the participant group definition used. You can use the Up and Down buttons to change the order of the process applications if you know you want to use the participant group definition from a different version for your scenario.</p>
	Processes to Include in Analysis	<p>Click the Add button to choose the processes that you want to analyze. (Lombardi lists the BPDs that reside in the selected process applications.) If the Simulation Profile associated with the process that you choose is not the profile that you want, click the name of the profile in the right column, which enables you to choose another profile using a drop-down menu. If no profiles have been defined, only the Default profile is available. To remove a process, click the process name and then click the Remove button.</p>
Participant Group Overrides	Add/Remove	<p>Click the Add button to choose one or more of the participant groups from the selected process applications. Then change the values that you want to override, such as the capacity, cost per hour, and so on. (In the Capacity column you can type + <i>value</i> or -<i>value</i> to increase or decrease the capacity by a certain number of participants. You can also type just a <i>value</i> to specify an absolute capacity such as 1.0.) This table enables you to run a scenario with different settings for your participants without changing the actual Participant Group definitions, which can help you simulate different workloads. To remove overrides that you've set, click a participant group name and then click Remove.</p>  <p>If you specify Participant Group Overrides for multiple participant groups and a member belongs to one or more of those groups, that member uses the simulation overrides specified for the first participant group in the table. You can use the Up and Down buttons to change the order of the participant groups in the Participant Group Overrides table.</p>

To run a simulation using a defined scenario, see [Running simulations, historical analyses, and comparisons](#).

Configuration requirements for optimization

Using historical data captured by Lombardi Performance Data Warehouse, the Optimizer pinpoints areas in your process models where you can make design changes to help streamline execution and, thus, improve performance. If you are planning to deploy one or more processes and you want to capture the performance data that will enable you to optimize those processes, you must complete the following tasks in the order shown:

Task	Description	See
Ensure autotracking is enabled	With autotracking enabled, Lombardi captures data that automatically includes tracking points	Tracking performance data for the Optimizer

Task	Description	See
	at the entry and exit of each item in a process definition (such as activities and gateways). This data enables the Optimizer to analyze run-time task duration as well as compare how a process performs when it follows one path versus another.	
Specify the business data (variables) to track	To capture the value of business data at every point as it flows through the process, specify the variables to track. Doing so ensures that you get the most value out of your process analysis. For example, knowing which of your suppliers is causing the most exceptions in your quality assurance process is valuable information.	Tracking performance data for the Optimizer
Send tracking definitions	The data described in the preceding tasks is captured to the Performance Data Warehouse only if you send tracking definitions as instructed in the following procedures.	Tracking performance data for the Optimizer
Create historical analysis scenarios	After your processes have been running for a while and you want to analyze the collected data using the Optimizer, create historical scenarios to specify the process models to include, the business data (variables) by which to filter the analysis results, and whether to include only completed or also currently running process instances. The advantage of historical scenarios is that they enable you to group and compare performance for different sets of processes.	Creating historical analysis scenarios

Optional configuration for optimization

Lombardi provides the following configuration options for the Optimizer, which may prove useful or helpful in your environment:

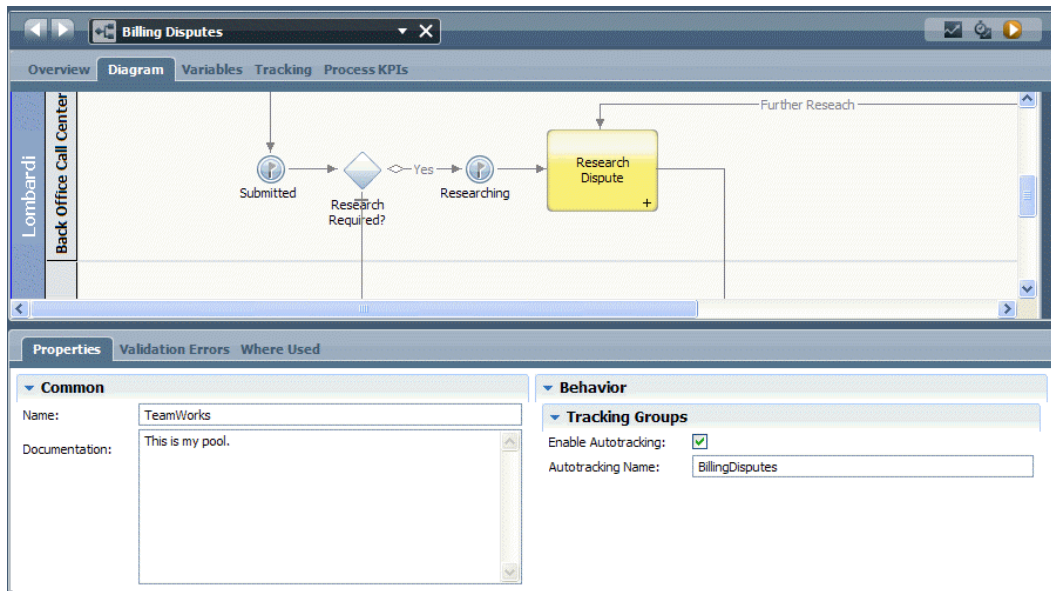
Option	Description	See
Generate historical data	You can set up a Simulation Profile and generate historical data based on that profile if you want to run historical analyses based on data that you generate. This option is helpful if you want to simulate your processes using business data versus the timing data that is normally available with simulations.	Generating historical data
Analyze data from Performance Data Warehouses in runtime environments	You can configure Lombardi to enable the Optimizer to perform its analysis on the data from Performance Data Warehouses in runtime environments. This option is helpful if you want to be able to select a Performance Data Warehouse other than the local warehouse when running an analysis.	Analyzing data from Performance Data Warehouses in runtime environments

Tracking performance data for the Optimizer

To track performance data, you should ensure autotracking is enabled, specify the business data to track, and then send the tracking definitions to the Performance Data Warehouse as described in the following sections.

Using autotracking

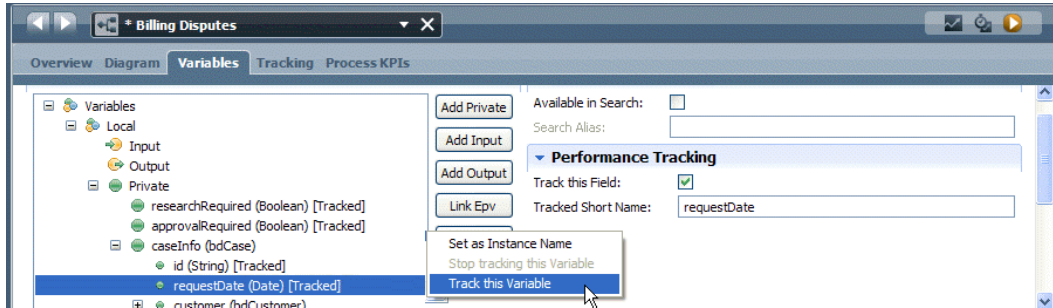
Autotracking is enabled by default. You can open the process diagram in the Designer in Lombardi Authoring Environment, click the Lombardi pool, and verify that the **Enable Autotracking** checkbox is set in the properties tab. Because you want to add variables to track so that you can analyze performance data according to particular business variable values, enter an autotracking name in the Properties tab as shown in the following image:



Lombardi uses the autotracking name to create a view in the Performance Data Warehouse database to hold the tracked data. For the preceding example, Lombardi creates a `BillingDisputes` view in the database that includes a column for each variable that you elect to track as described in the following section. (For more information about the Performance Data Warehouse database, see [Performance Data Warehouse database architecture](#).)

Specifying the business data to track

To specify the business data (variables) to track, go to the Variables tab for your process, right-click each variable that you want to track, and select **Track this Variable** as shown in the following image:



Lombardi creates a column in the `BillingDisputes` view for each tracked variable, using the variable name shown in the **Tracked Short Name** field.

At a minimum, you should track:

- The unique identifiers within the process that represent keys into your system of record data.
- The data that you will want to analyze later on such as customer name, customer segment, product type, or request type.
- Values that you expect to change during the course of a process. For example, in a financial dispute process, you might track the approved amount of the dispute as the dispute moves through the process.

Sending tracking definitions to the Performance Data Warehouse

After enabling autotracking and specifying the variables to track, save the process and then send your newly defined tracking requirements to the Performance Data Warehouse. From the Lombardi main menu, select **File > Send Definitions to Performance Data Warehouse**.





You should send definitions whenever you make changes to your process diagrams or when you change the tracking or business data in your processes, including creating or editing scenarios.

When you install process application snapshots in a runtime environment, the Process Server in that environment automatically sends tracking definitions to its corresponding Performance Data Warehouse. See [Installing process applications: online Process Servers](#) to learn how to install snapshots and ensure that definitions are sent as expected. After definitions are sent and process instances are up and running, you can analyze data for those processes in that runtime environment as described in [Analyzing data from Performance Data Warehouses in runtime environments](#).

Creating historical analysis scenarios

When you define a historical analysis scenario, you provide information that the Optimizer requires such as the processes to include, the business data (variables) by which to filter the analysis results, and whether to include only completed or also currently running process instances.

1. In the Designer in Lombardi Authoring Environment, click the plus sign next to Processes and select **Historical Analysis Scenario** from the list of components.
2. Enter a name for the scenario and click **Finish**.
3. In the Scenarios editor, provide the following information:

Dialog area	Field or control	Description
Common	Documentation	Optionally provide a description in this field.
Historical Data Filters	Include Process Instances	By default, the All option is enabled, which means that data for both completed and currently running process instances is analyzed by the Optimizer. If you want to analyze data for only currently running process instances, select In-Flight Only . If you want to analyze data for only completed process instances, select Completed Only .
	Time Range	Select a time range for the data that the Optimizer will analyze such as Last Week . Select Custom to use the calendars to pick a Start and End Date.
	Process Apps to Include in Analysis	<p>Click the Add button to choose the process applications that you want from the Lombardi repository. Choose the process applications that contain the processes that you want to analyze. Be sure to select the correct snapshot (version) to analyze. If you want to analyze all versions, select the snapshot named (All) for the process application that you want. To remove a process application, click the application name and then click the Remove button.</p>  <p>If you do not add any process applications to this table, all process applications in the Lombardi repository to which you have Read access are included, which means that you can analyze processes from any of those applications. If you select multiple snapshots (versions) to analyze, the first version listed in the table determines the participant group definition used. You can use the Up and Down buttons to change the order of the process applications if you know you want to use the participant group definition from a different version for your scenario.</p>
	Processes to Include in Analysis	<p>Click the Add button to choose the processes that you want. The processes available are the ones that reside in the process applications that you added in the preceding table. To remove a process, click the process name and then click the Remove button.</p> <p>Tip: Be sure to add sub-processes so that you can drill down during analysis.</p>  <p>If you do not add any processes to this table, all processes that reside in the process applications that you added in the preceding table are analyzed.</p>
	Business Data	Click the Add button to select the tracked variables by which you want to filter the results of this scenario. The variable names that you add must be tracked variables for each of the processes included in this scenario. If a process included in the scenario does not have a matching tracked variable for each variable name that you add here, no instances of that process will be returned in the analysis results. Choose an operator using the drop-down list in the Comparison column and then enter the Value to compare. To remove a variable, click the variable name and then click the Remove button.

Analyzing data from Performance Data Warehouses in runtime environments

When using the Optimizer, you can run your historical analyses using data from any of the Performance Data Warehouses in your Lombardi configuration. For example, if you have several runtime environments (staging, test, and production) in which your processes are running, you can choose to analyze processes using the stored data from those environments.

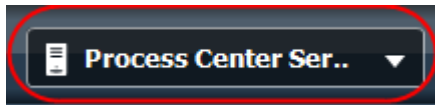
The following must be true in order to analyze processes using data from a Performance Data Warehouse in a runtime environment:

- Process Servers in runtime environments must be connected to the Process Center as described in *Lombardi Runtime Environment Installation Guide*.
- You must meet the configuration requirements for historical analyses as described in [Configuration requirements for optimization](#).
- Lombardi must be tracking and storing data in the Performance Data Warehouse in the runtime environment.

When you install process application snapshots in a runtime environment, the Process Server in that environment automatically sends tracking definitions to its corresponding Performance Data Warehouse. See [Installing process applications: online Process Servers](#) to learn how to install snapshots and ensure that definitions are sent as expected. After definitions are sent and process instances are up and running, you can analyze data for those processes in that runtime environment using the Optimizer.

When you meet the requirements listed above, the Optimizer includes a menu for the runtime servers connected to the Process Center. To analyze data from runtime environments:

1. Open the Optimizer as described in [Running simulations, historical analyses, and comparisons](#).
2. Click the menu shown in the following image to choose the runtime environment that you want.



3. Run your analysis as described in [Running simulations, historical analyses, and comparisons](#). The results displayed reflect the performance data from the environment that you chose in step 2.



The Optimizer enables you to analyze data from one warehouse at a time.

Generating historical data

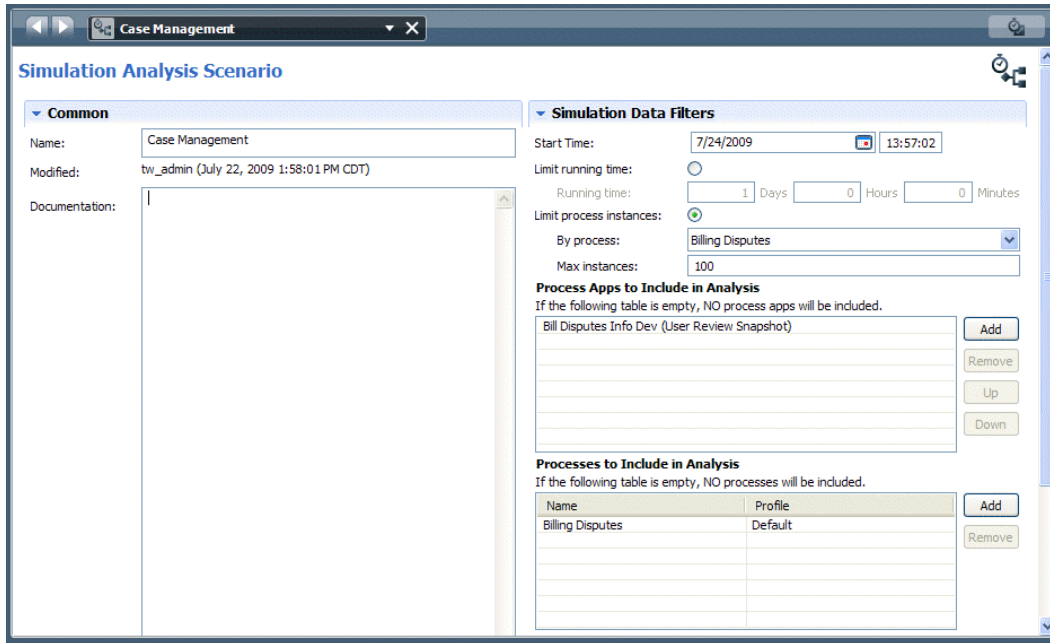
Normally, you would run instances of your processes in a production environment for some time in order to generate and store meaningful performance data in the Performance Data Warehouse. However, if you want to simulate your processes using business data versus the timing data that is normally available with simulations, you can use a Lombardi utility to automatically generate performance data using simulation analysis scenarios.

To generate historical data using a simulation analysis scenario:



Before generating historical data, be sure to use autotracking, specify the business data (variables) to track, and send tracking definitions to the Performance Data Warehouse as described in [Tracking performance data for the Optimizer](#).

1. From the Lombardi library, double-click a simulation analysis scenario to open it. For this procedure, we'll use the sample scenario shown in the following image:



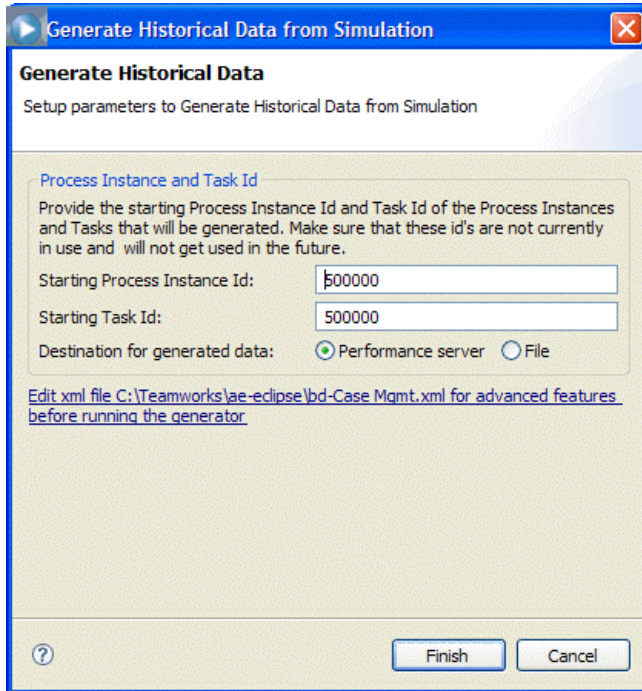
In the preceding scenario, you can see that we'll be generating data for one version of a process called Billing Disputes. The Optimizer will generate data as if 100 instances of the process actually ran, to match the Max instances setting in this scenario. To learn more about the Billing Disputes sample process, see [Sample simulations](#).



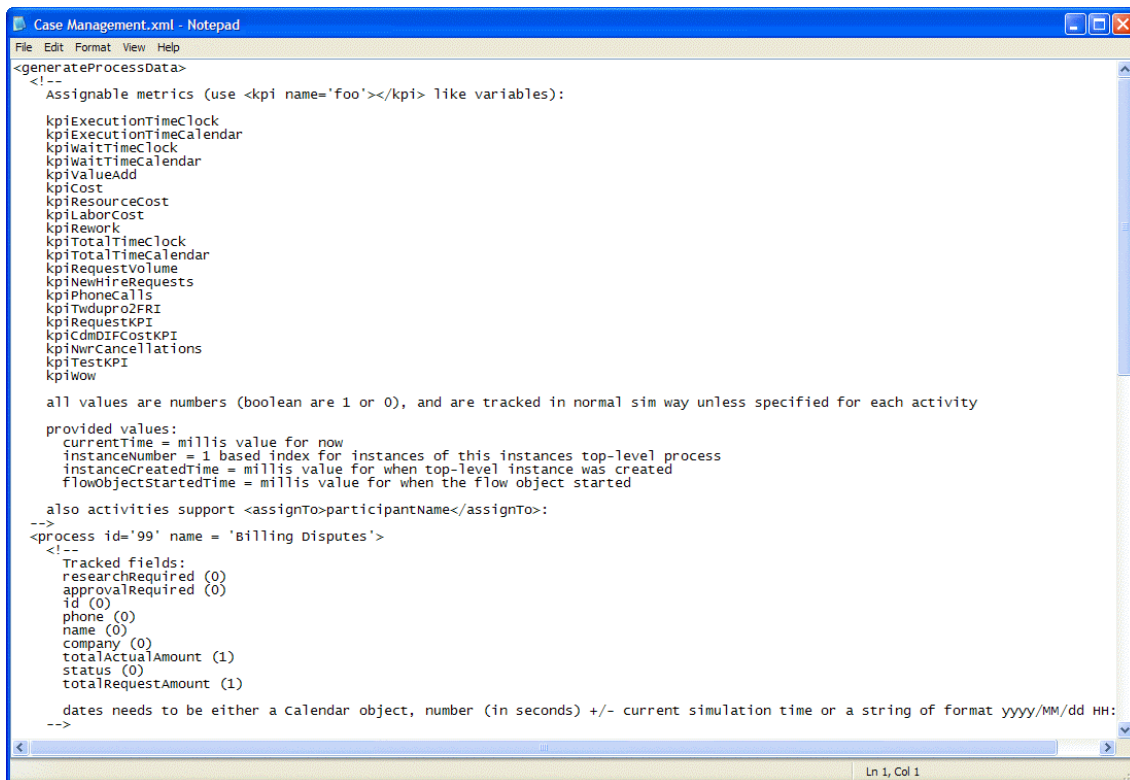
If you select multiple versions in the preceding dialog, this utility generates data for all of the versions that you include. However, in the generated data, you cannot specify different data per version. So if you want to generate different data per version, you should run the utility and alter the data as needed for each version.

2. From the Lombardi main menu, select **Scenario > Generate Historical Data**.
3. In the Generate Historical Data dialog, set the Instance and Task ID values to previously unused ranges (this is necessary in order to avoid data overwrites where data has been previously generated for the selected scenario). If this is the first time data has been generated for this scenario, you don't need to change these values.
4. Accept the default **Performance Data Warehouse** option as the Destination for generated data to go ahead and edit the local XML file that Lombardi generates.

You can select the **File** option if you want to save initial data to the local XML file, open and edit the XML file in your favorite editor, and then come back to this dialog later to generate and send the historical data to the Performance Data Warehouse.



5. Click the **Edit xml file link** to edit the local XML file by providing specific values for the steps in the process included in your scenario. If the XML file already exists, the Optimizer opens the file. If the file does not exist, the Optimizer creates a new file and then opens it.
6. Lombardi Optimizer opens the local XML file. The following image shows an example file:



The XML file has the same name as the simulation analysis scenario that you're using to generate historical data. In the preceding sample you can see all the data that will be tracked, including the default Key Performance Indicators (KPIs) and the tracked fields for the process included in the sample scenario (Billing Disputes).

- You need to scroll through the XML file and provide values to initialize the data for each activity and flow component in the process. For example, you can initialize the customer name for the Gather Dispute Information activity using standard Javascript as shown in the following image:

```

<flowObject id='bpdid:5e6cd2e3e2fad952:29c8c36a:1138c758934:-7f5c' name='Start'>
</flowObject>
<flowObject id='bpdid:5e6cd2e3e2fad952:29c8c36a:1138c758934:-7f4d' name='Gather Dispute Information'>
  <!-- Activity -->
  <!--<assignTo>'bd online call center west'</assignTo-->
  <var name='name'>
    var arr = new Array('BJR Supplies', 'Majestic', 'ABC Inc', 'Acme');
    arr[Math.floor(Math.random()*arr.length)];
  </var>

```

The Javascript randomly iterates through the names in the supplied array.

- You can provide a simple calculation to determine the `totalActualAmount` and `totalRequestAmount` for the Gather Dispute Information activity as shown in the following image:

```

<var name="totalActualAmount">
  500 + Math.floor(75 * (Math.random()- .5));
</var>
<var name="totalRequestAmount">
  500 + Math.floor(75 * (Math.random()- .5));
</var>

```

- You can also establish the process flow. For example, you can provide algorithms to determine the values for variables like `researchRequired` and `approvalRequired` as shown in the following image:

```

<var name='researchRequired'>
  var s;
  if (Math.random() &lt; .90) { s = 'Yes'; } else { s = 'No'; }
  s;
</var>
<var name='approvalRequired'>
  var s;
  if (Math.random() &lt; .60) { s = 'Yes'; } else { s = 'No'; }
  s;
</var>

```

Then you can branch the flow of the process according to values of these variables by scrolling down to the gateways and indicating what should happen for each condition. In the following example, you can see that we've established what should happen for the Yes condition for the Research Required gateway:

```

<FlowObject id='bpdid:5e6cd2e3e2fad952:29c8c36a:1138c758934:-7be4' name='Research Required?'>
  <!-- Gateway 1, conditions needed:1-->
  <conditions>
    <condition name="Yes" bpmnId="bpdid:5e6cd2e3e2fad952:29c8c36a:1138c758934:-7bd7">researchRequired=='Yes'</condition>
  </conditions>
</FlowObject>
<FlowObject id='bpdid:fc87b608e236270a:-6209de48:113b7dd45d9:-7beb' name='Researching'>
</FlowObject>
<FlowObject id='bpdid:fc87b608e236270a:-6209de48:113b7dd45d9:-7c01' name='Submitted'>
</FlowObject>
  
```

- When your edits are complete and you are ready to generate and send data to the Performance Data Warehouse, click the **Finish** button.



If you're simulating a large number of instances (for example, 50,000 instances) generating historical data can take up to 10-15 minutes per scenario. If you notice slower performance, do not perform any other work while this utility is running.

Lombardi Optimizer generates historical data using the specifications from the simulation analysis scenario and the values from the XML file and sends the data to the Performance Data Warehouse. (Be sure to set the Performance Data Warehouse as the destination when you're ready to generate and send data.)

After the data is sent to the Performance Data Warehouse, you can perform historical analyses for the included processes.



If you change the structure of the processes that you are analyzing or if you change the processes included in the simulation analysis scenario, you must delete the existing XML file before generating historical data for the revised processes or scenario.

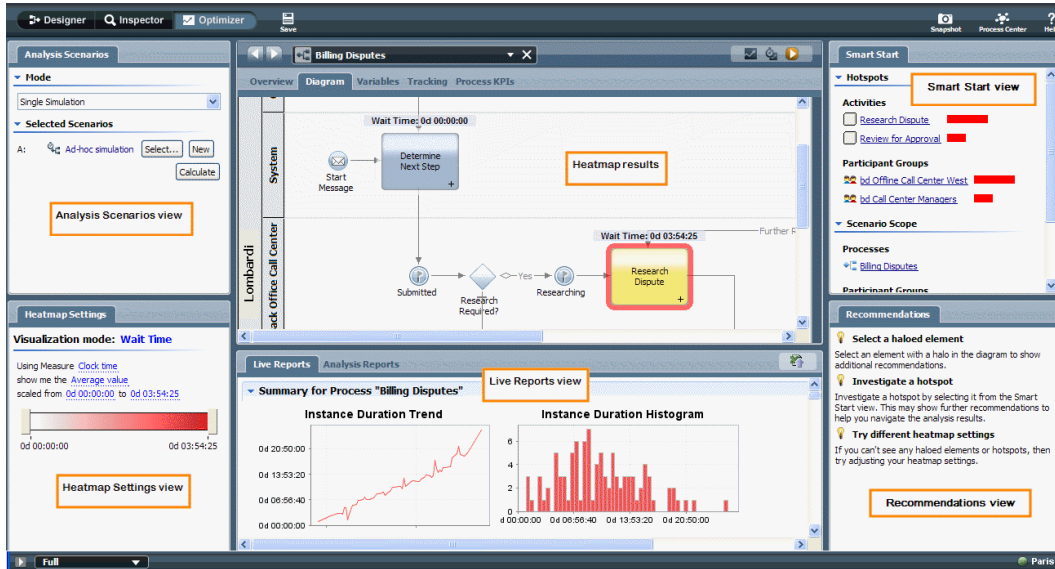
Running simulations, historical analyses, and comparisons

Before you begin

Before getting started, you should:

- See [Configuration requirements for simulation](#) to understand the configuration tasks required to run simulations.
- See [Configuration requirements for optimization](#) to understand the configuration tasks required to run historical analyses and comparisons.
- Access the Optimizer by selecting **Optimizer** from the drop-down menu at the top of Lombardi Authoring Environment.

The following image shows the views in the Optimizer that you will work with:



To see all views in the Optimizer, including the Smart Start view and the Recommendations view, select **Full** from the drop-down menu in the lower left corner of Lombardi Authoring Environment as shown in the preceding image. Select **Simple** from the drop-down menu if you want to hide the Smart Start and Recommendations views.

Running scenarios

1. From the drop-down list in the Analysis Scenarios view, select the mode that you want.

Available modes include:

Single Simulation	Simulate the processes included in the Simulation Analysis Scenario that you select in the following step.
Simulation vs. Simulation	Simulate and compare the processes in one Simulation Analysis Scenario to those in another scenario per your selections in the following step.
Single Historical	Analyze the stored performance data for the processes included in the Historical Analysis Scenario that you select in the following step.
Historical vs. Historical	Analyze and compare the stored performance data for the processes in one Historical Analysis Scenario to those in another scenario per your selections in the following step.
Historical vs. Simulation (How did I do)	Compare the stored performance data for the processes in the Historical Analysis Scenario to the simulations for the processes in the Simulation Analysis Scenario per your selections in the following step.
Simulation vs. Historical (What if)	Compare the simulations for the processes in the Simulation Analysis Scenario to the stored performance data for the processes in the Historical Analysis Scenario per your selections in the following step.



To run historical analyses using data from other configured environments, see [Analyzing data from Performance Data Warehouses in runtime environments](#).

2. Click the **Select** button beneath **Selected Scenarios** to select the analysis scenario(s) that you want to run.

If you're running a comparative analysis, you must select a baseline scenario (B) and a sample scenario (A) to compare to the baseline.


3. In the Heatmap Settings view, click the currently displayed Visualization Mode to see a list of available modes.

Visualization Modes enable you to establish the criteria for the heat maps and live reports that the Optimizer generates for the processes included in your scenario(s).

The Optimizer displays Wait Time, Execution Time, and Total Time only for those activities that generate end-user tasks. The Optimizer does not display Wait Time, Execution Time, or Total Time for activities that are implemented using sub-processes.

By default, the KPI thresholds used by the Visualization Modes are the thresholds from the current working version of your process application or toolkit. If you want to use the KPI thresholds from the snapshot (version) of your process application or toolkit that was most recently executed and tracked, change the Optimizer preference setting (for KPI threshold values) to: Use the KPI threshold values from the actual version of the Process App/Toolkit. You can access preference settings from the main Authoring Environment File menu: **Preferences > Lombardi > Optimizer**.

Select the Visualization Mode that you want from the list:

Wait Time	Measures the time that elapses between Lombardi generating a task and an end user opening that task. For example, the amount of time that a task sits in an end user's Inbox in Lombardi Process Portal before it is opened is considered Wait Time.
Execution Time	Measures the time that elapses between an end user opening a task and then completing and closing that task. For example, the amount of time that it takes an end user to fill in required data on a Coach form is considered Execution Time.
Total Time	Measures the time that elapses between Lombardi generating a task and an end user closing that task (Wait Time + Execution Time).
Efficiency	Compares the expected execution time established in the Execution Time Key Performance Indicator (KPI) to the actual execution time. For example, if the actual execution time for a task is 2 hours and the expected execution time set in the Execution Time KPI is 4 hours, the Optimizer displays an efficiency of 200%.  You can set KPIs in the KPI tab for each activity. If you don't set values for each field in a KPI, Lombardi uses default values. You can open each KPI to see the default values. For more information, see Associating KPIs with activities .
Waiting Activities	Displays the count or total volume of tasks generated by Lombardi that have not yet been opened by an end user.
Executing Activities	Displays the count or total volume of tasks generated by Lombardi that have been opened by an end user.
Completed Activities	Displays the count or total volume of tasks generated by Lombardi that have been closed by an end user.
Happy Path	Shows how often the happy paths (best case routes) through a process are taken.
Exception Path	Shows how often exception paths (alternative routes) through a process are taken.
Path	Displays results for all paths (Happy Path + Exception Path).
SLA	Displays results based on Service Level Agreement (SLA) violations. For more information, see Creating SLAs .

Rework	Displays results based on Activities that violate the Rework KPI. By default, an Activity is considered rework if it is executed more than once during a process instance. You can change the default settings for the Rework KPI in the KPI tab for each Activity.
--------	---



When you're running in a non-comparison mode (single simulation or single historical) and you view resulting heat maps, items highlighted in red are problematic. When you're running in a comparison mode, items highlighted in blue represent the sample scenario (A) and items highlighted in red represent the baseline scenario (B).

4. Edit the settings for the Visualization Mode that you select.

For example, if you select **Wait Time**, set the measure, value, and scale that you want from the following specification:

Using *measure*, show me *value*, scaled from *days:hours:minutes* to *days:hours:minutes*.

The selections for *measure* include:

Clock time	Includes all elapsed time.
Calendar time	Includes only business hours from the elapsed time.

The selections for *value* include:

% of instances outside of range	Shows the percent of process instances outside the activity threshold or fixed range that you designate. (You can designate activity thresholds using the KPIs tab in the Process Modeler's properties.)
Average value	Shows the average wait times within the scale that you designate.
Total value	Shows the total wait times within the scale that you designate.

For the scale, you can specify the low and then high values of the range of wait times for which you want to check. For the Wait Time mode, specify the range in *days*, *hours*, and *minutes*.

5. In the Analysis Scenarios view, click the **Calculate** button.
6. Examine the results of the analysis as outlined in [Reviewing results](#).

Reviewing results

Lombardi Optimizer presents analysis results in:

- Heat Maps
- Live Reports
- Recommendations
- Smart Start Hotspots



To see all views in the Optimizer, including the Smart Start view and the Recommendations view, select **Full** from the drop-down menu in the lower left corner of Lombardi Authoring Environment.

The following sections describe how to interpret these results.

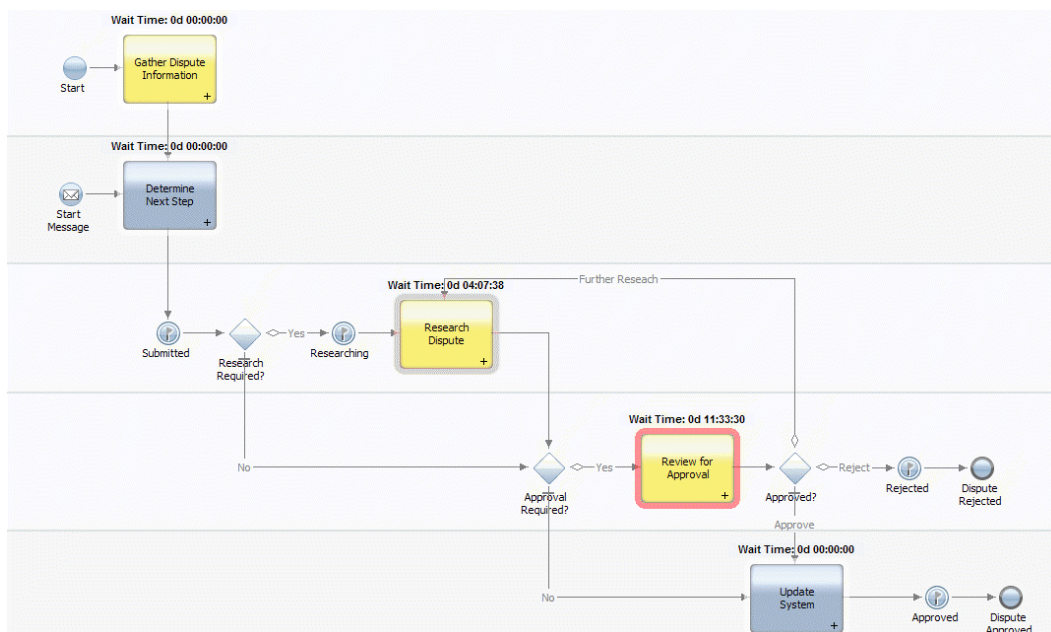
Understanding heat maps

The Optimizer displays a color-coded heat map to visually illustrate where bottlenecks and other problems exist in the processes included in your scenario, and how severe those issues are. The darker the halo around an activity, the closer it is to the high end of the scale or range that you specified in the Heatmap Settings view.



To display heat maps for other processes in the scenario, you can use the Smart Start view. Click an activity shown in the Hotspot list to go directly to the process in which that problematic activity resides. You can also click a process name shown in the Scenario Scope. However, some processes included in the scenario may not produce issues (halos around activities) or show results per the current Visualization Mode. Details about the Smart Start view are provided in a following section.

The following example heat map shows two activities identified as bottlenecks because their wait times all exceed the valued specified at the high end of the scale for the Wait Time Visualization mode (3 hours):



You can see details about the data used to render the heat map color coding by mousing over an activity that is surrounded by a halo. When you do, the Optimizer displays relevant data in easy to read charts and graphs. These same charts and graphs are also included in the Live Reports view.

For simulations, the Optimizer uses the default simulation data or the simulation values that you provided when creating simulation profiles to show you where bottlenecks are likely to occur. For historical analyses and comparisons, the Optimizer uses stored performance data to indicate problem areas in your processes.

Understanding live reports

The data displayed in the Live Reports view depends upon the current editor and selection. For example, if you are examining a heat-mapped process diagram and you have selected an activity in that diagram, the Live Reports view shows data specific to that activity.

The following figure shows a live report for a process. To view such a report, click the process pool in a heat-mapped process diagram.



A live report for a process includes the sections shown in the preceding figure. The first two charts illustrate duration data for instances of this process and the final pie chart shows all processes that the users involved in this process worked on.

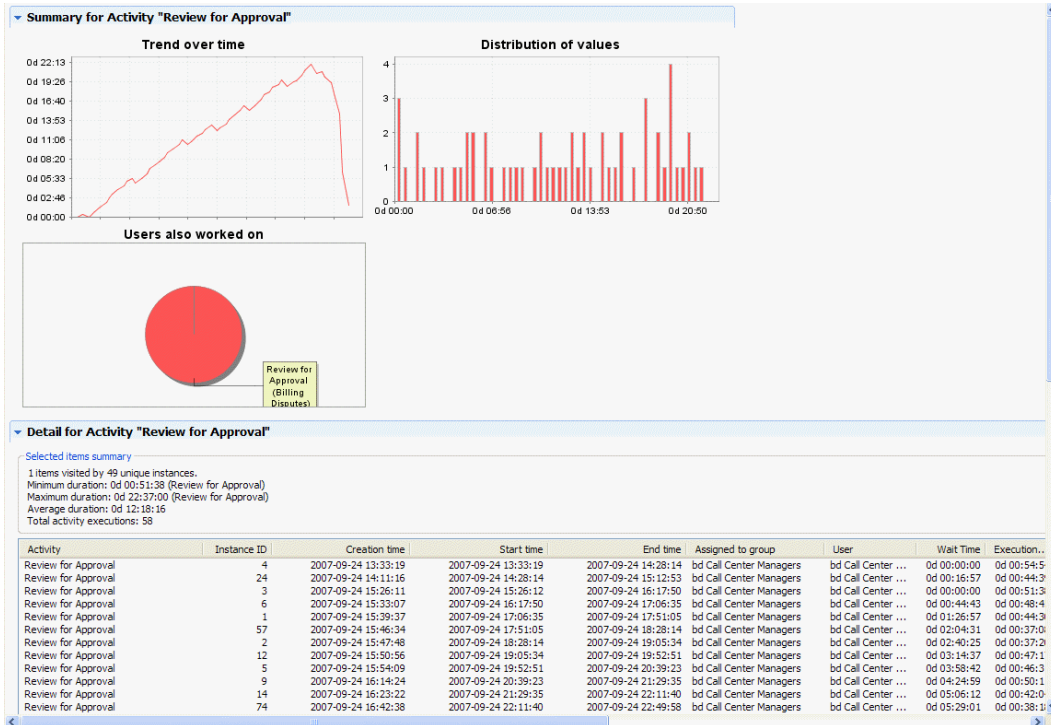


When you run a comparative analysis, the first two charts include data in red and blue where blue represents the performance of the favorable scenario.

The other sections and information include:

Section	Information provided
Instance Analysis	Shows the number of executing and completed instances as well as duration.
KPI Analysis	Provides information regarding the Key Performance Indicators (KPIs) tracked for each activity in this process.
Activity Analysis	Displays data for each activity in the process, including the number of waiting, executing and completed activities, and the minimum, maximum, and average waiting and execution times for each activity.

The following figure shows a Live Report for an activity. To view such a report, click an activity in a heat-mapped process diagram.



The first two charts displayed in a Live Report for an activity are the same charts that you see when you mouse over an activity in a heat-mapped process diagram. The final pie chart shows all activities that the users for this activity worked on.

The Details table for the activity includes different columns of information, depending on which Visualization Mode is currently set. For the example report shown in the preceding figure, the Visualization Mode is Wait Time.

You can click other elements in your process diagram to see live report data for those elements. For example, if you click on a path from a gateway, the Live Report view shows the process instances that followed that path.

When you select multiple activities or other process elements in a process diagram, the Detail table in the Live Report view includes data for each selected element. The same is true when you select a swim lane in a process diagram. Selecting a swim lane causes the Live Report view to show data for each process element in that lane.

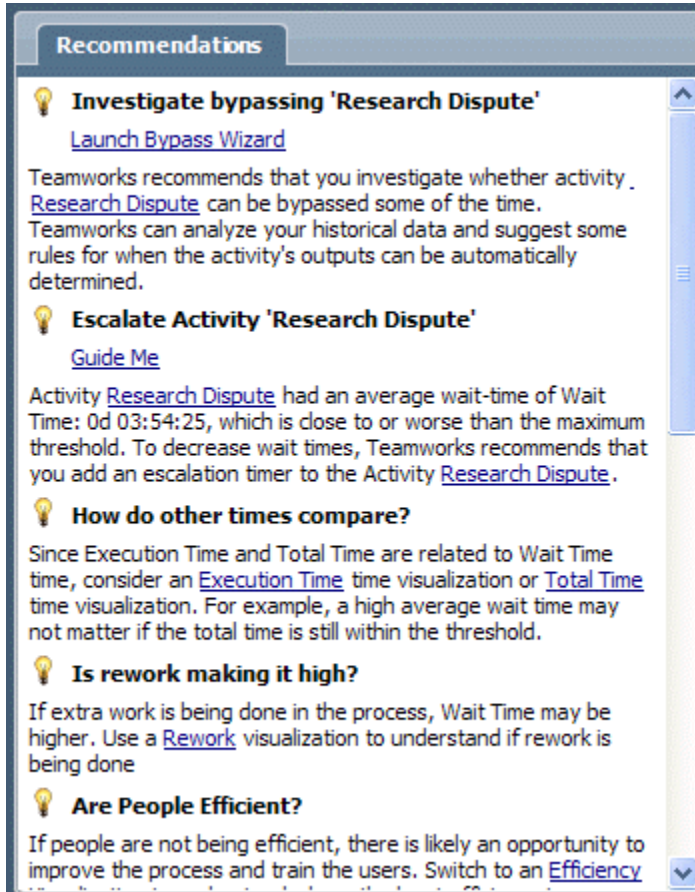
When you run a comparative analysis, such as Simulation versus Simulation, the Live Reports view shows two Detail tables—one for Scenario A and another for Scenario B.



The Details table in a live report includes only the first 1,000 rows of available data. However, when you open the report in Excel, all rows are included. To open a report in Excel, click the Excel icon at the top of the report.

Reviewing recommendations

To get recommendations for a problematic Activity or other element in a process, click an element with a halo around it in the heat map. The following image shows an example Recommendations view from the Lombardi Optimizer:



To see all views in the Optimizer, including the Recommendations view, select **Full** from the drop-down menu in the lower left corner of Lombardi Authoring Environment.

The Recommendations view makes practical recommendations for addressing issues that are identified in your processes, and suggestions for how to optimize your process models. The recommendations may encourage you to examine other visualization modes to gain a better understanding of a particular pattern or behavior in your processes.

Resolving identified issues can involve questions such as:

- Would different resource allocations resolve my current bottlenecks? (Time and resource consumption)
- Are my processes taking the paths that I expect them to? What changes will ensure that they do? (Path optimization)
- How are my largest loans going through the process? How does that compare to my smaller loans? Why are very large loans always late? (Segment optimization)

Some recommendations are presented with a cheat sheet that guides you through performing the recommended actions, step by step. To open the cheat sheet, click **Guide Me**. Other recommendations provide instructions for refining your analysis or suggestions for improving process performance, such as prioritizing tasks, training resources, and so on.

The data displayed in the Recommendations view depends upon the current editor and selection. For example, if you run a scenario, the Recommendations view initially instructs you to select a haloed element

or investigate hotspots. The preceding image shows an example of what the Recommendations view looks like after running an Analysis Scenario and then selecting a highlighted activity in a heat-mapped process diagram.

The cheat sheets in the Recommendations view provide three types of interactive Help:

Tell Me	Provides step-by-step instructions for completing a task within the Authoring Environment graphical interface.
Show Me	Provides step-by-step instructions that include actions to take you to the part of the interface where steps are performed.
Do It For Me	Provides executable Help actions which, when clicked, perform a task or parts of a task for you.

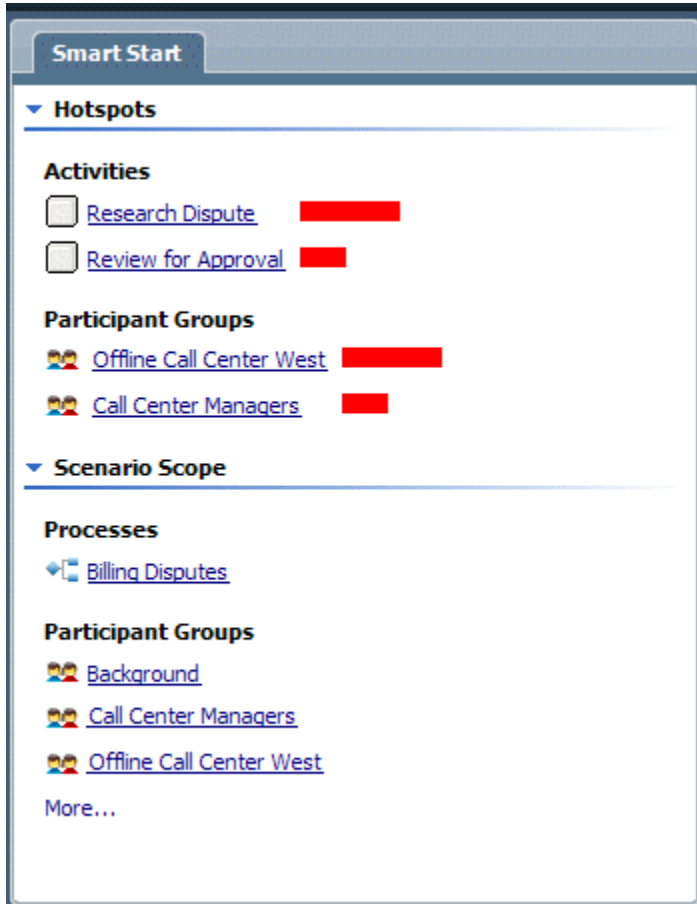
The example in the preceding image shows the recommendations for an Activity that is bottlenecked. Review all the recommendations to determine which action would best address the bottlenecks for the Activity. For example, you may not have the option to add more resources to the Activity, so you would consider other alternatives that are presented in the Recommendations view.



Some recommendations include Guided Optimization. To learn more, see the section about using the Guided Optimization Wizard in [Sample historical analyses and comparisons](#).

Using the Smart Start view

The following image shows the Smart Start view:



To see all views in the Optimizer, including the Smart Start view, select **Full** from the drop-down menu in the lower left corner of Lombardi Authoring Environment.

The Smart Start view directs you to the activities and processes that deserve a closer look based on the most recently executed analysis scenario and the current visualization mode. The Smart Start view enables you to directly access:

- Hotspots identified by the Optimizer when you run an analysis scenario
- Processes and participant groups included in the most recently executed analysis scenario

For example, if several activities in several different BPDs exceed a range that you establish in the Heatmap Settings, you can click each activity shown in the Hotspots list to go directly to the BPD in which that activity resides. The problem activity is shown in red in the heatmap of the BPD.

The available Hotspots in the Smart Start view are determined by the Visualization Mode you choose in the Heatmap settings. For example, if you choose Wait Time as the Visualization Mode, the Hotspots list includes:

- Activities — displays activities that meet or exceed the criteria that you establish for the most recently executed Analysis Scenario. Click a listed activity to see the BPD in which it resides.
- Participant Groups — displays participant groups that meet or exceed the criteria that you establish for the most recently executed analysis scenario. Click a listed participant group to open it.

However, if you choose Exception Path, Happy Path, or Path as the Visualization Mode, the Hotspots list shows paths that meet or exceed the criteria that you establish instead of activities.

The Scenario Scope in the Smart Start view includes:

- Processes — displays a list of each of the processes included in the most recently executed analysis scenario. Click a listed process to view the BPD.
- Participant Groups — displays a list of each of the participant groups included in the most recently executed analysis scenario. Click a listed participant group to open it.

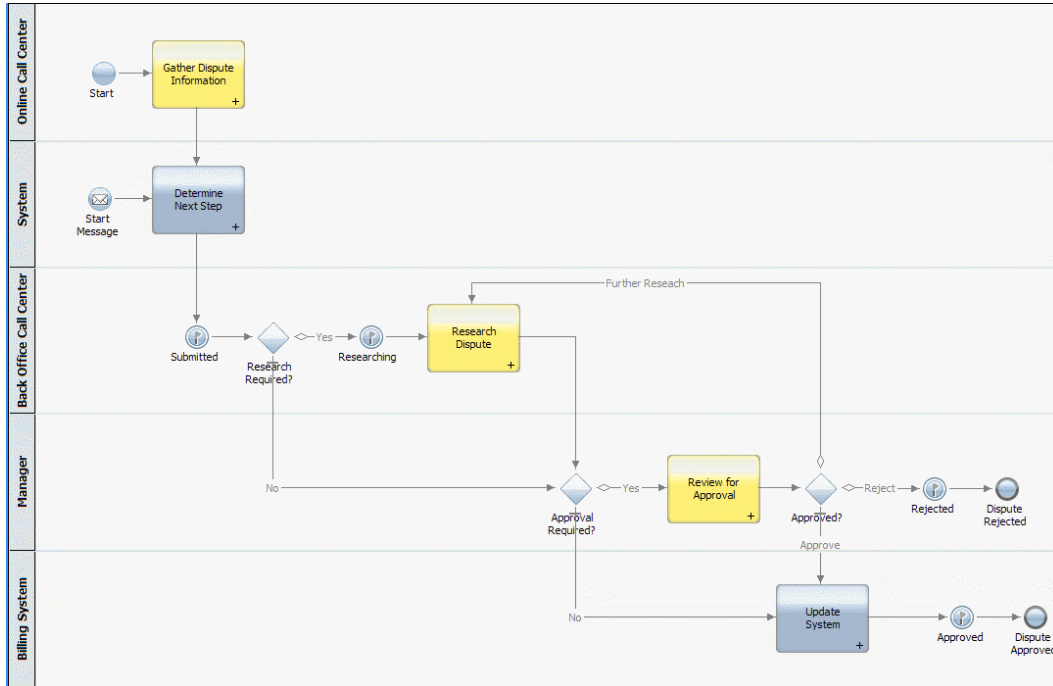
Sample simulations

Running a quick simulation

Before you actually implement a process, simulations can help you pinpoint potential issues such as bottlenecks caused by resource constraints or a particular path being taken more often than is optimal. In this sample, we'll walk through setting simulation values for the elements in a process and then quickly running a single simulation of the process.

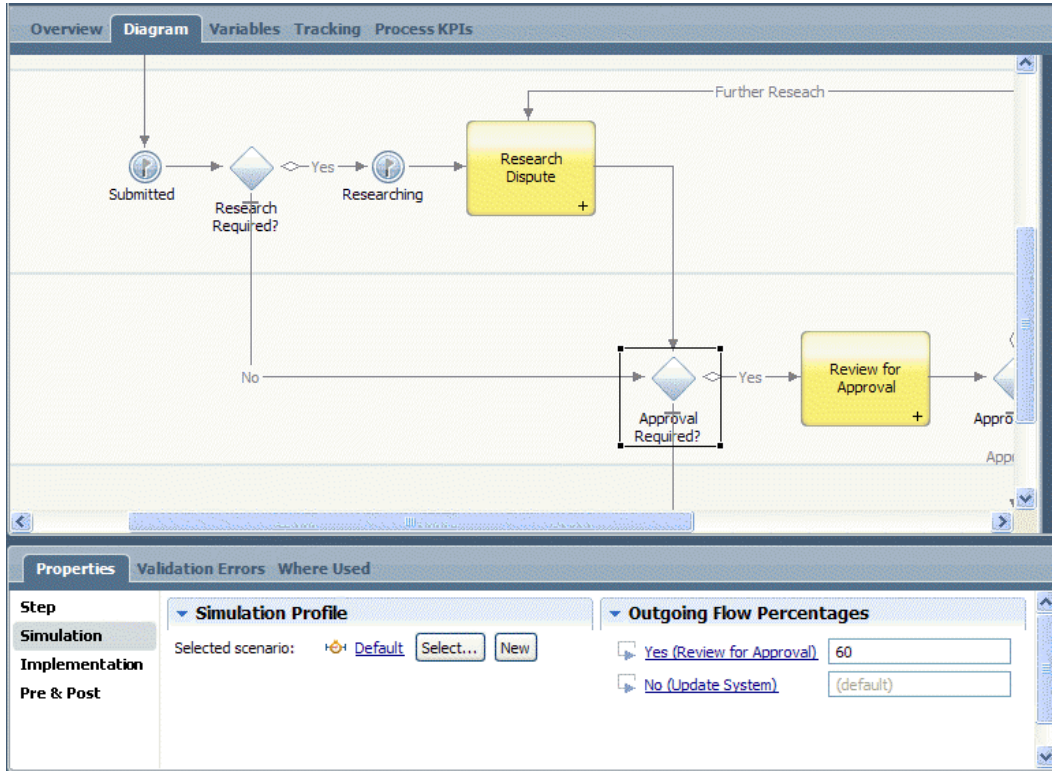
The following image shows the diagram of the sample process that we'll be simulating. The process is used to determine whether to reject or accept a billing dispute. The flow of the process is as follows:

- The process begins with someone in the call center gathering dispute information from a client or as the result of a message received by the billing disputes system.
- The billing disputes system determines if further research is required and also if approval is required.
- If research is required, the offline call center performs the research and passes the dispute on to the call center managers for review (if approval is required).
- If managers review and approve the dispute, the billing system performs the necessary updates to enact the transaction.
- If managers review and determine that more research is required, the dispute goes back to the offline call center for further research.
- If research and review are not required, paths exist to take the dispute directly to the billing system as approved.
- If research is required and review is not required, a path exists from the research activity to route the dispute to the billing system as approved.
- If managers review and reject the dispute, the process ends.



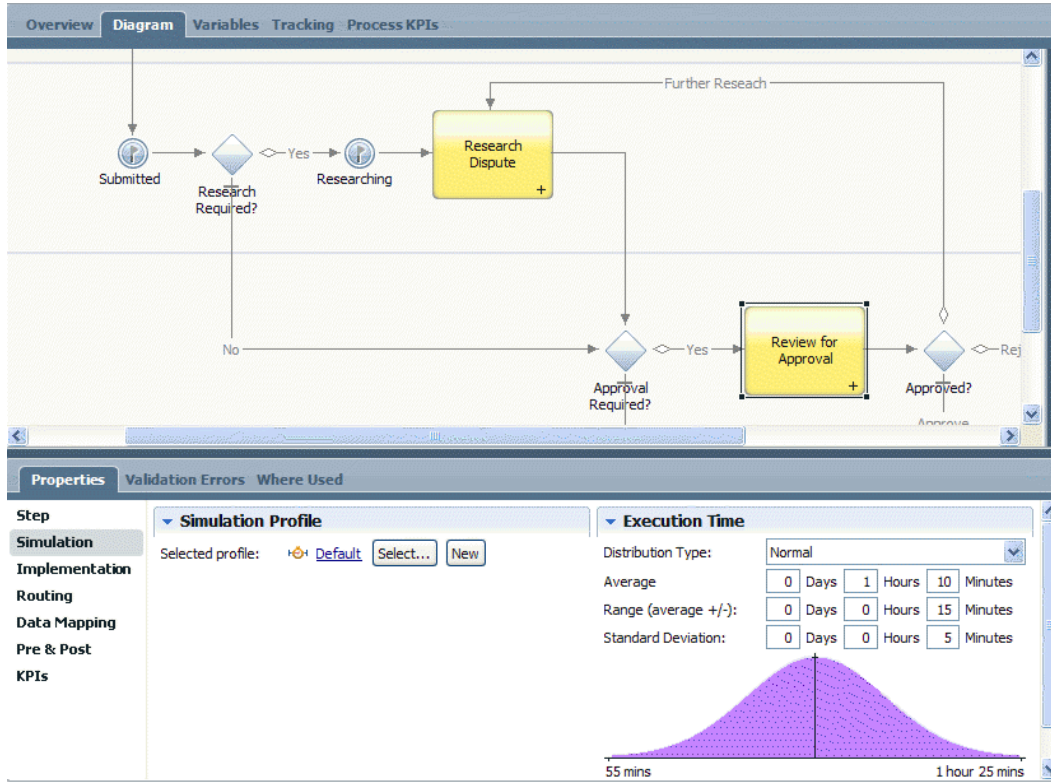
To start, let's set simulation values:

1. Select the **Approval Required** gateway in the diagram and then select the **Simulation** option in the properties.
2. To facilitate running a quick simulation, we leave the Selected scenario at **Default** and then estimate that the Outgoing Flow Percentage for the Yes (Review for Approval) path is **60** per cent. This means we believe that 60 per cent of the disputes that don't require research are going to require approval as shown in the following example.

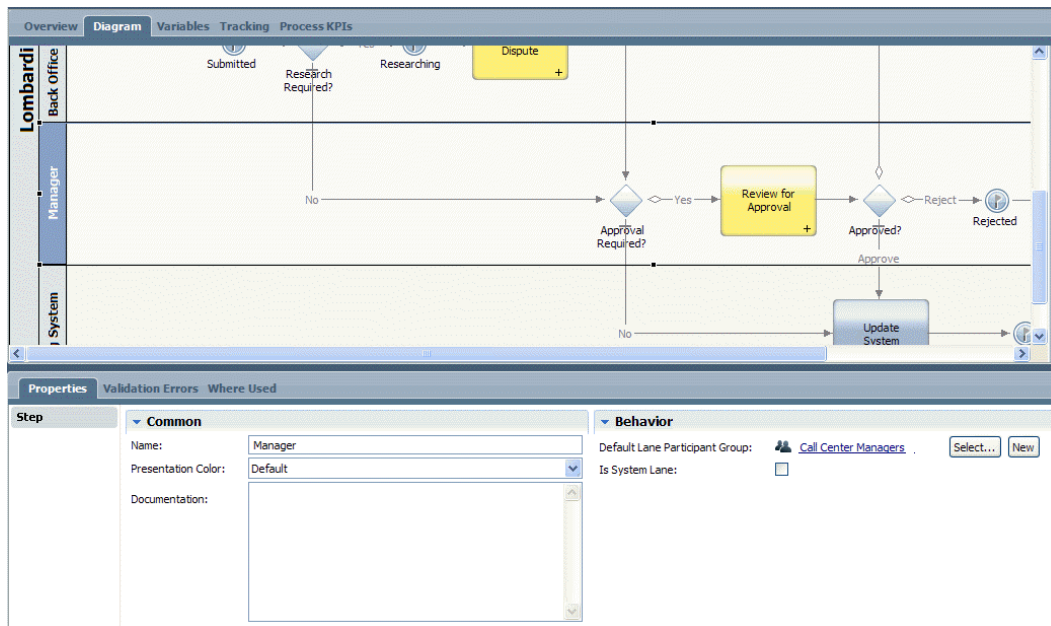


3. Select the **Review for Approval** activity in the diagram.

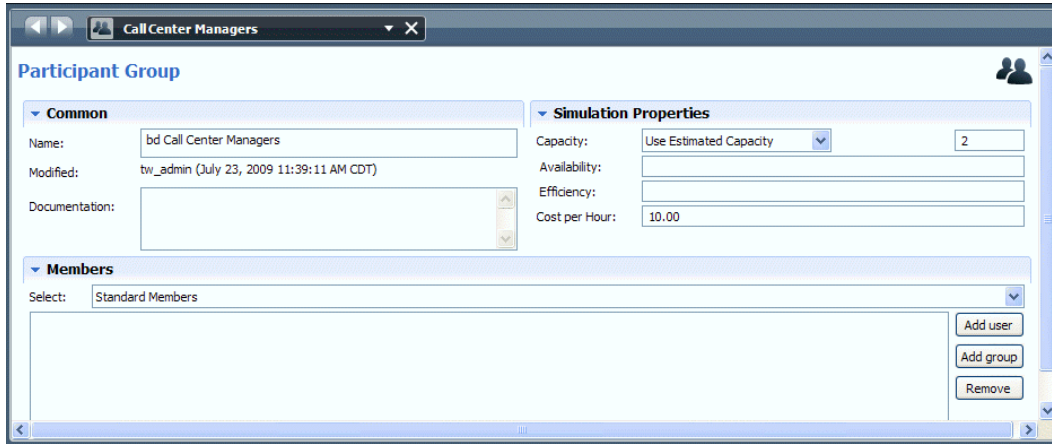
In the Simulation properties, we want to use the **Default** scenario for this quick simulation so we leave that setting and then estimate the Execution Time. We select **Normal** for the distribution type and we think it will take managers an average of **1 hour and 10 minutes** to approve or reject a dispute.



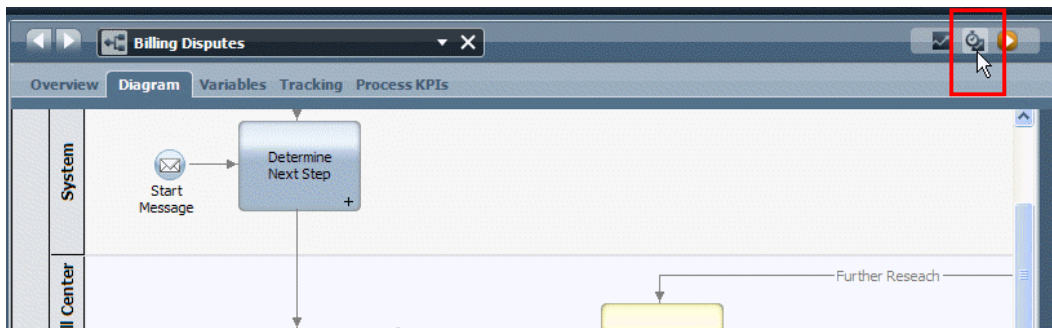
- To estimate the number of managers for our simulation, we click the **Manager** swimlane. Click the default participant group for the lane (**Call Center Managers**) to open and edit the Simulation properties for the group.



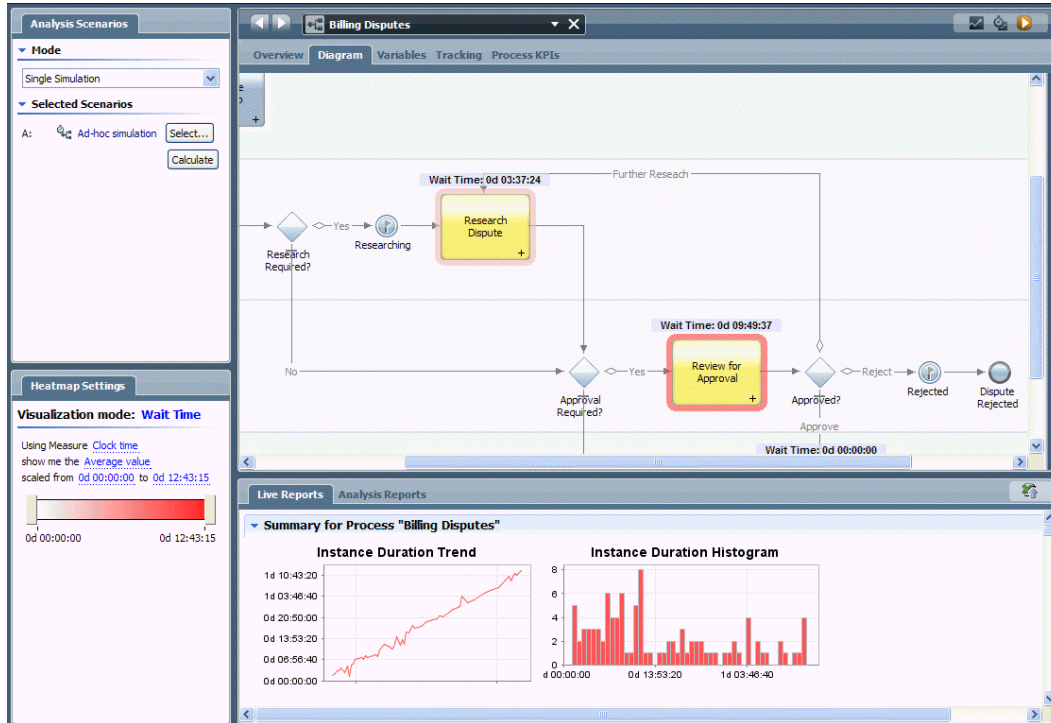
We expect to have at least two managers available to handle this work and so under Simulation Properties, select **Use Estimated Capacity** and enter **2** in the corresponding text box.



- To run a simulation using the values provided in the preceding steps, go to the BPD diagram and select the icon illustrated in the following image to start a new simulation for the current process.

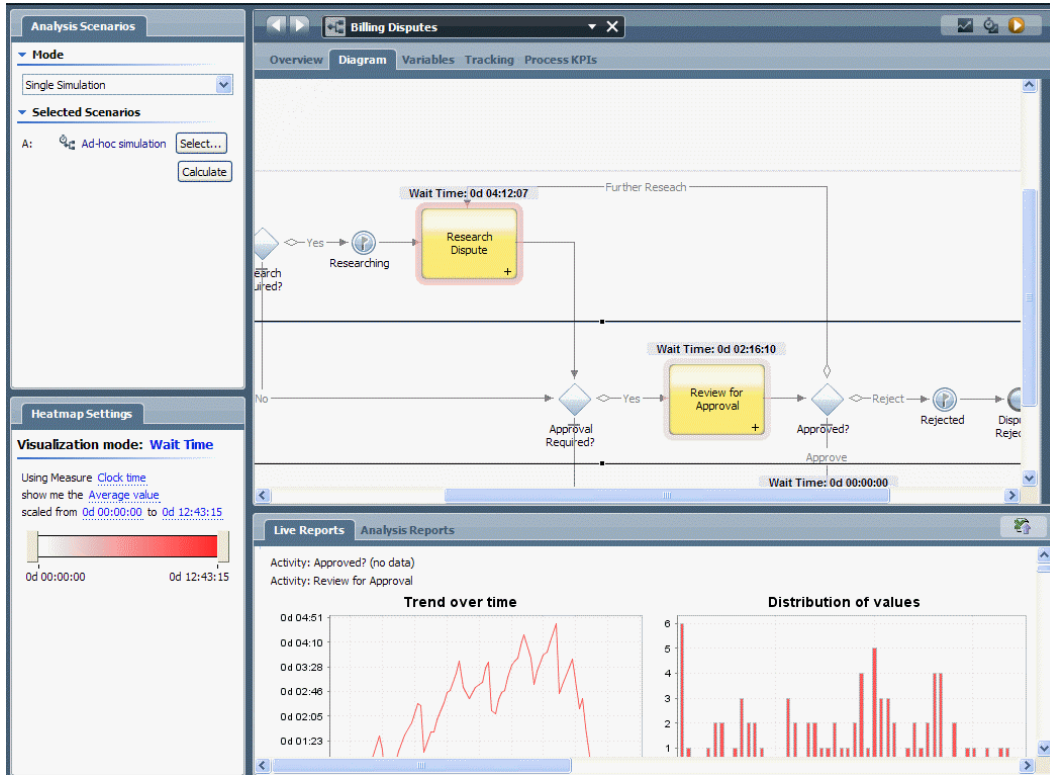


- When you run a simulation for your current process using the toolbar icon, the default Visualization mode is Wait Time. So, the heat map results highlight tasks that could potentially cause gaps in process execution because a person or system is not available to complete those tasks:



The preceding heatmap shows that the Review for Approval activity has the longest wait time of approximately 9 hours.

7. To improve the WaitTime, we go back to the properties for the Call Center Managers participant group and change the estimated capacity from 2 to 4 managers.
8. We run the simulation again from the toolbar, and this time the results are better:



By doubling the available resources, we've reduced the wait time for the Review for Approval activity from 9 hours to 2 hours.

The Optimizer enables you to quickly manipulate simulation settings to ensure that you are able to closely represent the environment in which your process will run. In some cases, it may be necessary to alter the process design to accommodate other constraints. For example, in the preceding sample you may know that it is impossible to increase the number of available managers for this process. So, you might consider adding a Timer Event to the Review for Approval activity to remind managers of pending approvals at defined intervals. This would make sense if the pending approvals for billing disputes are higher priority than other tasks for members of this management group.

The advantage of the Optimizer is that you can create and save several Simulation Profiles to represent different staffing levels and other variables, which enables you to demonstrate predicted performance to your management or other corporate teams invested in the success of the process models. The following section illustrates the flexibility provided by Simulation Profiles and Simulation Scenarios when you need to analyze predicted performance to foster decisions about proposed process designs.

Taking advantage of simulation profiles and scenarios

After running some quick simulations in the preceding section, we know that adding resources to our process results in acceptable wait times. But, we don't believe it will be possible to actually increase resources for the problematic activities. We could, however, devise a way to categorize billing disputes so that a smaller percentage of disputes would actually require approval. Because we want to demonstrate the options and outcomes to the corporate automation team, we decide to create two separate Simulation Profiles for this single process:

- One simulation profile called **Current Flow** to represent the current flow of disputes that require approval.

- An additional simulation profile called **Improved Flow** to represent improvements gained from categorizing disputes prior to submission.

To create the simulation profile called Current Flow, follow these steps:

1. Open the Billing Disputes process in the Designer in Lombardi Authoring Environment.
2. Select the **Approval Required** gateway in the diagram.
3. Click the **Simulation** option in the properties.
4. Under Simulation Profile, click the **New** button to create a new profile to represent the current flow of disputes that require approval.
5. Name the new profile **Current Flow** and click **OK**.
6. In the Outgoing Flow Percentage area, enter **60** for the Yes (Review for Approval) flow:

The screenshot displays the Lombardi Authoring Environment interface for the 'Billing Disputes' process. The top window shows the process diagram with several elements: a decision gateway 'Research Required?', an activity 'Researching', another decision gateway 'Approval Required?', an activity 'Review for Approval', and a final decision gateway 'Approved?'. The 'Properties' panel at the bottom is open to the 'Simulation' tab. Under 'Simulation Profile', the 'Current Flow' profile is selected. Under 'Outgoing Flow Percentages', the 'Yes (Review for Approval)' flow is configured with a percentage of 60, while the 'No (Update System)' flow is set to its default value.

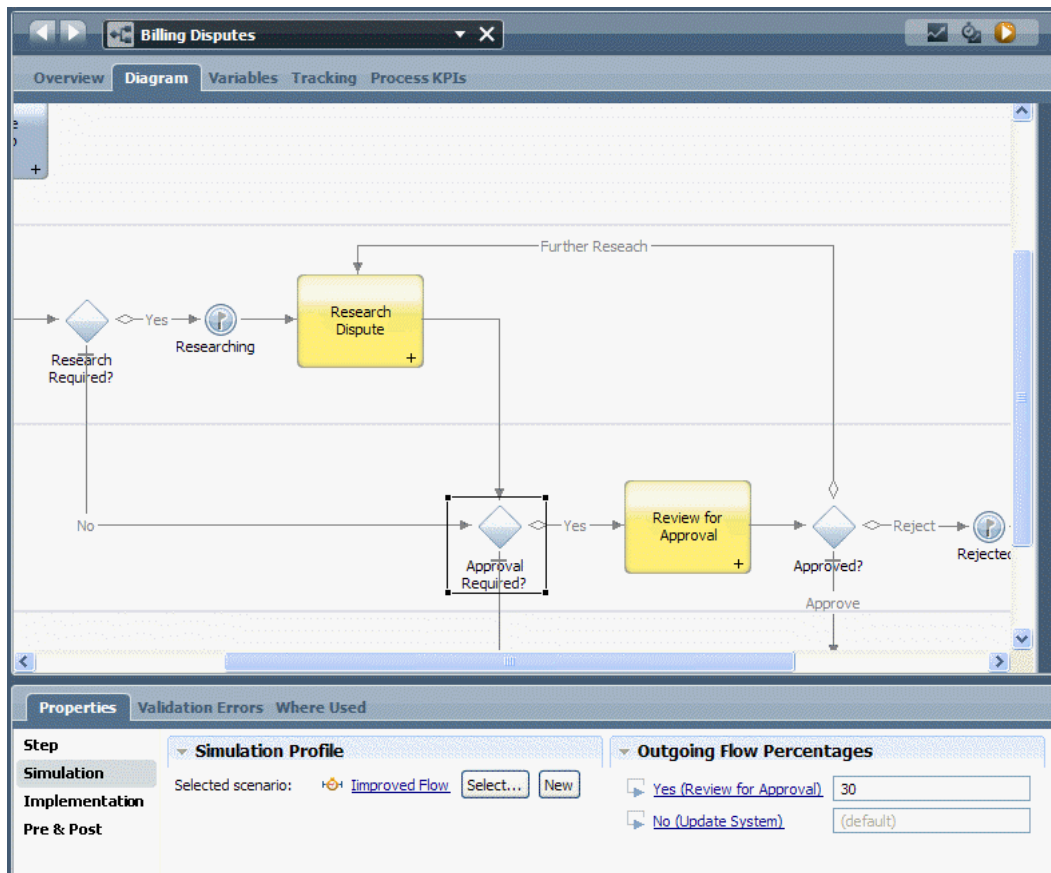
7. Select each activity and gateway for which you want to specify simulation values for the Current Flow profile.



When specifying Simulation properties for each process element, be sure that the Current Flow profile is selected.

To create the simulation profile called Improved Flow, follow these steps:

1. Select the **Approval Required** gateway in the Billing Disputes process diagram.
2. Click the **Simulation** option in the properties.
3. Under Simulation Profile, click the **New** button to create a new profile to represent the improved flow of disputes.
4. Name the new profile **Improved Flow** and click **OK**.
5. In the Outgoing Flow Percentage area, enter **30** for the Yes (Review for Approval) flow:



6. Select each activity and gateway for which you want to specify simulation values for the Improved Flow profile.



When specifying values in the Simulation properties for each process element, be sure that the Improved Flow profile is selected.

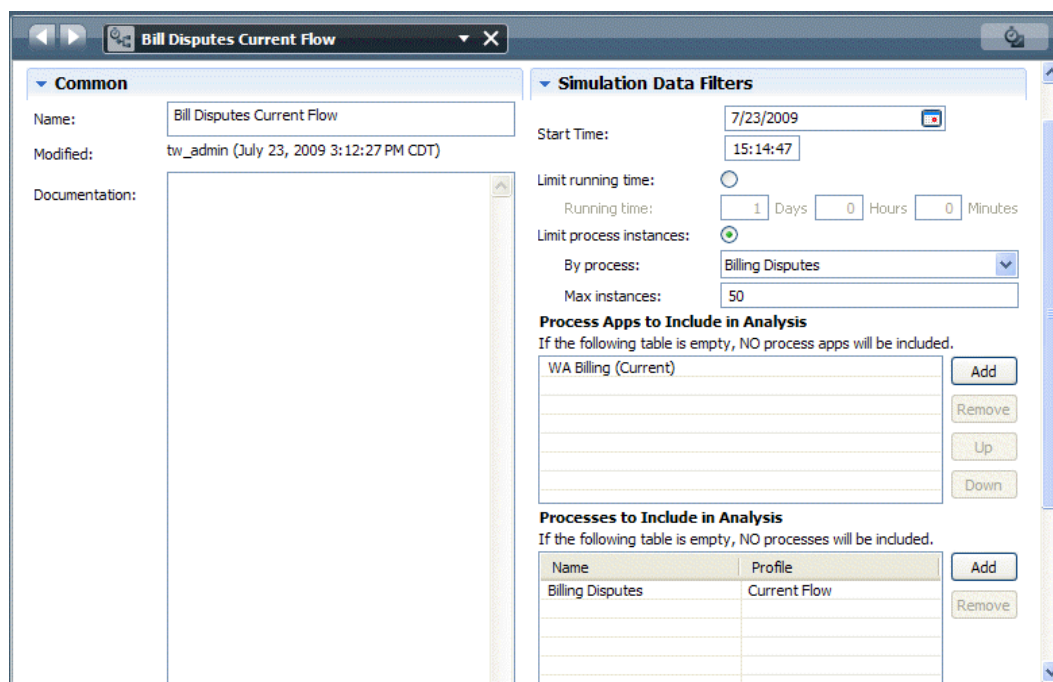
We also need to create two separate simulation scenarios so that we can run a comparative analysis.

To create the first scenario, follow these steps:

1. In the Designer in Lombardi Authoring Environment, click the plus sign next to Processes and select **Simulation Analysis Scenario** from the list of components.

2. Name the scenario **Bill Disputes Current Flow** and click **Finish**.
3. Click the **Limit process instances** radio button and set the **Max instances field** to a value of **50**.
4. Under **Process Apps to Include in Analysis** click the **Add** button and then select the process application that contains the Billing Disputes process.
5. Under **Processes to Include in Analysis** click the **Add** button and then select the **Billing Disputes** process from the list.
6. Click **Default** under **Simulation Profile** to display a list of Simulation Profiles that have been created for the Billing Disputes process and then select the **Current Flow** profile.
7. Leave the Participant Group Overrides section blank.

The following image shows the completed scenario:



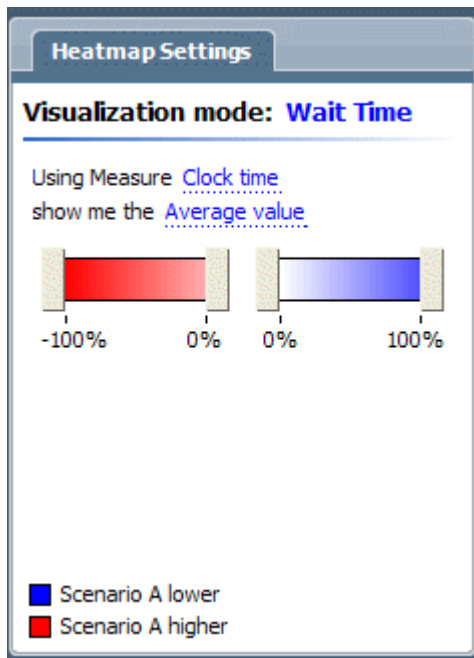
8. Click Save icon in the main toolbar.

To create the second scenario, follow these steps:

1. Click the plus sign next to Processes and select **Simulation Analysis Scenario** from the list of components.
2. Name the scenario **Bill Disputes Improved Flow** and click **Finish**.
3. Mimic the settings from the previous scenario, except this time, choose the **Improved Flow** Simulation Profile.
4. Click Save in the main toolbar.

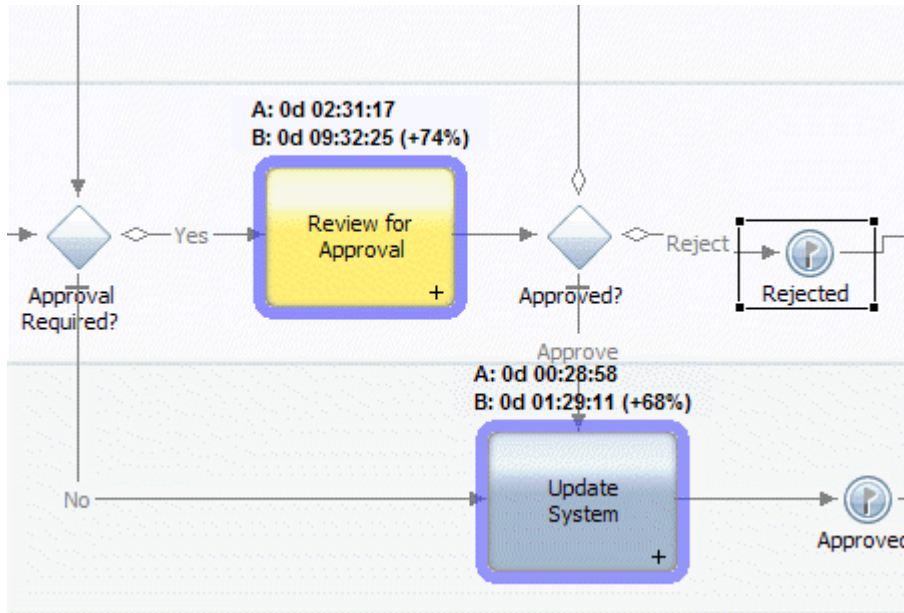
Now we can run in Simulation versus Simulation mode to compare wait time, execution time, and so on.

1. If you haven't already done so, open the Optimizer using the drop-down list at the top of Lombardi Authoring Environment.
2. In the Analysis Scenarios view, choose **Simulation vs. Simulation** from the Mode drop-down list.
3. Under **Selected Scenarios**, for sample A click the **Select** button and choose **Bill Disputes Improved Flow** from the list.
4. For baseline B, click the **Select** button and choose **Bill Disputes Current Flow** from the list.
5. In the Heatmap Settings view, we want the Visualization mode setting to be **Wait Time** with the specific settings shown in the following image:



6. Click the **Calculate** button in the Analysis Scenarios view.

The heat map shows that reducing the percentage of disputes that require approval decreases the wait time for the Review for Approval activity from approximately 9.5 hours to 2.5 hours.



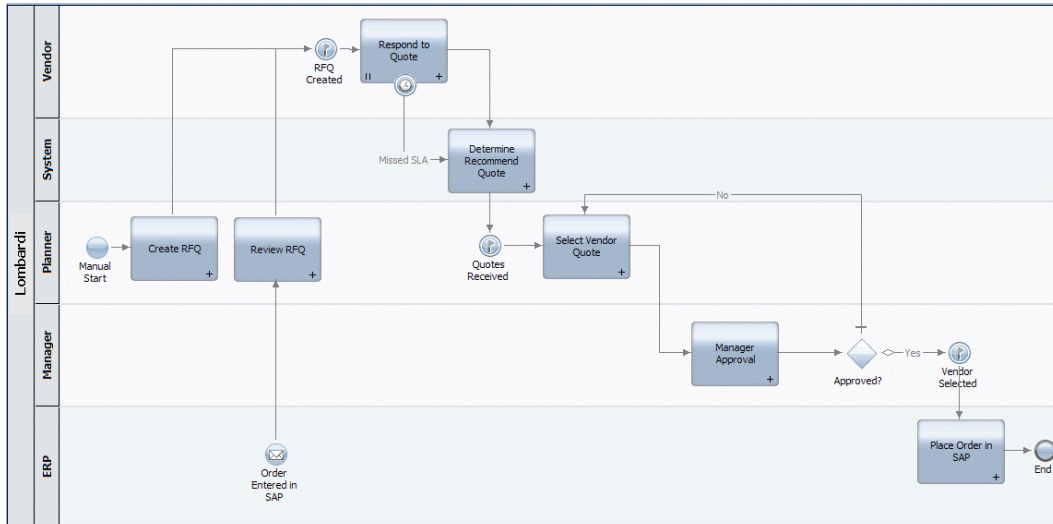
So, we are able to demonstrate that reducing the percentage of disputes routed for review to 30% (we theorize that we can do this by adding a task and business variables to categorize disputes), improves wait times and process execution in general. The Optimizer enables you to quickly demonstrate the performance issues, plus Lombardi enables you to quickly build and then demonstrate proposed design solutions.

Sample historical analyses and comparisons

Running an historical analysis

When you're running instances of your processes and those processes have been configured to track data, you can run historical analyses in Lombardi Optimizer to determine how well those processes are performing. With simulations, you provide values that enable the Optimizer to estimate execution times and potential delays for activities. With historical analyses, you are able to analyze your processes based on the historical data that Lombardi tracks and stores in the Performance Data Warehouse.

The following image shows the diagram of the sample process that we'll be analyzing:



The flow of the sample process is as follows:

1. The process begins by a Planner creating a Request for Quote (RFQ).
2. A quote solicitation is sent out to a number of vendors.
3. Each vendor responds with a proposed quote.
4. The proposals are delivered to the Planner, who reviews them and selects one for approval.
5. The selected proposal is sent for Manager approval.

If approved, the Inventory system is updated and the order is placed.

If rejected, the Planner is asked to choose an alternative proposal. This loop is repeated until a proposal is approved.

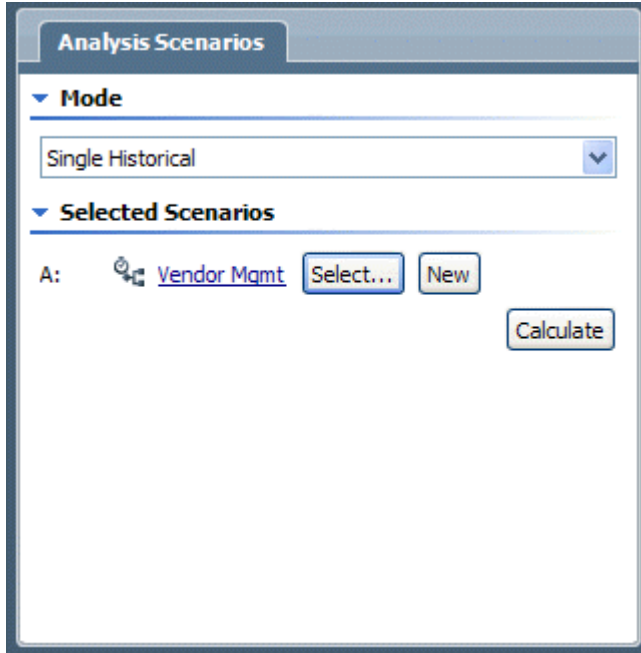
In the following example, you'll analyze historical data for this process to determine if any activities are bottlenecked. For this analysis, we need to configure one Historical Analysis Scenario as outlined in the following steps:

1. In the Designer in Lombardi Authoring Environment, click the plus sign next to Processes and select **Historical Analysis Scenario** from the list of components.
2. Name the scenario **Vendor Mgmt** and click **Finish**.
3. For the Include Process Instances option, select **All**. We want to analyze data for both completed and currently executing (in flight) process instances.
4. From the Time Range drop-down list, select **All Available**. We want to analyze all data stored in the Performance Data Warehouse, instead of data from a particular timeframe such as last month or last quarter.
5. Under **Process Apps to Include in Analysis**, click the **Add** button and then select the process application that contains the process.
6. Under **Processes to Include in Analysis**, click the **Add** button and then select the process that you want to analyze.

- Click **Save** in the main toolbar to save the scenario.

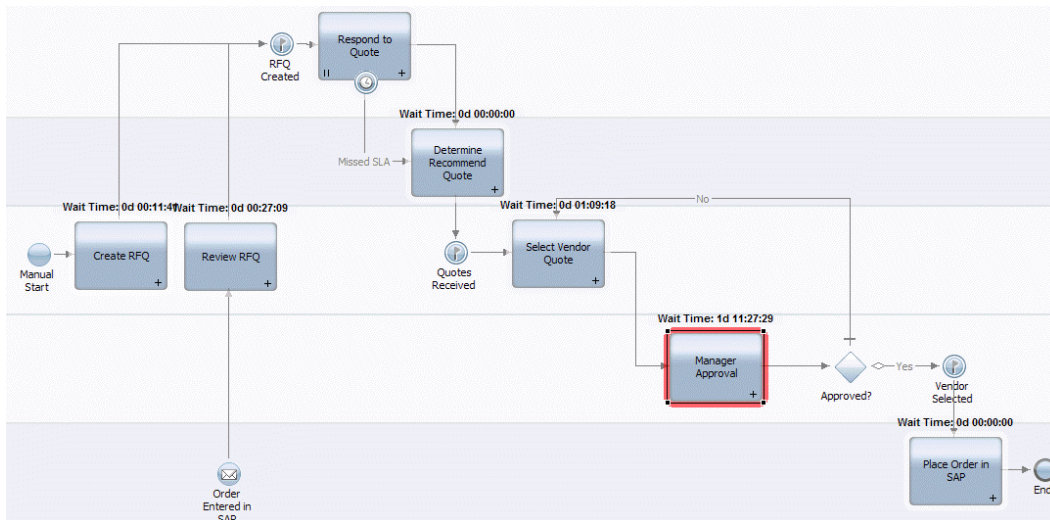
To run the analysis, follow these steps:

- Run the Optimizer with the Analysis Scenarios view settings shown in the following image:



- In the Heatmap Settings view, set the Visualization Mode to **Wait Time** and choose to show the Average value.
- Click the **Calculate** button in the Analysis Scenarios view.

The heat map shows analysis results like the following:



You can see that the wait time for the Manger Approval activity is over 1 day and 11 hours. This means that, on average, 1 day and 11 hours are elapsing between the time that a Manager Approval task is

received by managers and they are able to open and start the task. Considering that this analysis is based on actual data from this process running in your environment, this wait time is an average you'd want to improve. The Guided Optimization Wizard can help, as outlined in the following section.



After running a historical analysis as shown in the preceding example, you can save the Historical Analysis Scenario as a simulation. When the Optimizer finishes running a historical analysis, a **Save as Simulation** option appears in the Analysis Scenarios view. Click the option and then provide a name for the scenario in the New Simulation Analysis Scenario dialog. The Optimizer opens the Simulation Analysis Scenario so you can make any desired changes and save it for future use.

Using the guided optimization wizard

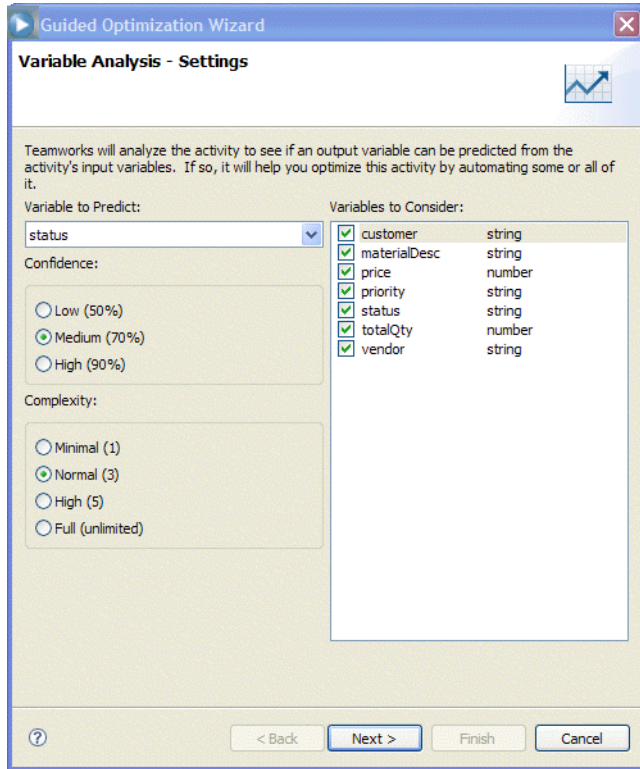
Lombardi provides guided optimization from the Recommendations view in the Lombardi Optimizer. The wizard helps you analyze your processes to determine if there are correlations between business data values and potential process outcomes. For example, the wizard may help you discover that you approve all claims under a particular amount for a particular vendor. In such cases, you could improve your process by automatically approving those claims and bypassing the manual review. The following procedure demonstrates how to use guided optimization to continue the process analysis that we started in the preceding section.



Guided optimization is available only when you perform historical analyses in Lombardi Optimizer.

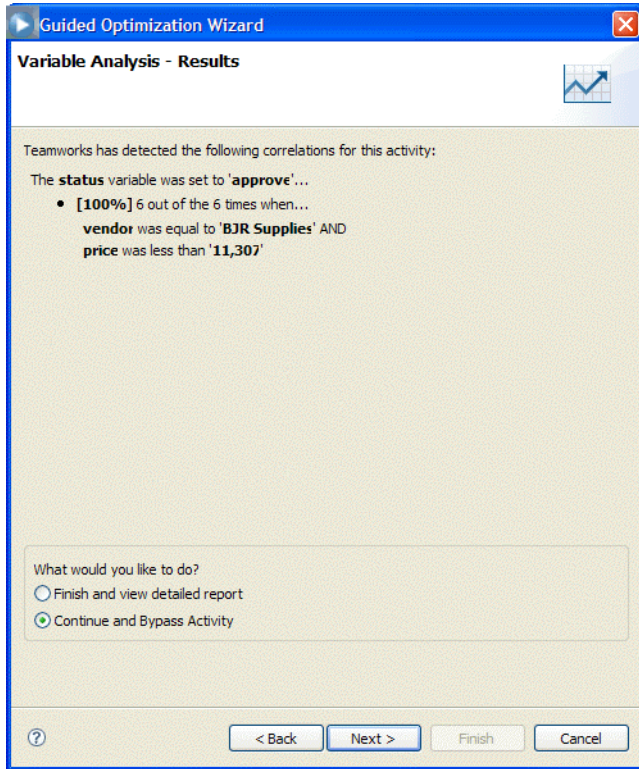
1. From the Recommendations view, scroll down to the recommendation named **Investigate bypassing 'Manager Approval'** and click **Launch Bypass Wizard**.
2. In the Variable Analysis - Settings dialog, select the **Variable to Predict** from the drop-down list. You should choose a variable that the activity changes. For example, in the following image, we've selected the status variable because the Manager Approval activity sets the value of this variable to either approve or reject. You want to choose a variable for which a certain set of business data might always create the same result. If so, you could just bypass the work done in the activity and automatically set the predicted value.
3. From the **Variables to Consider** section of the Variable Analysis - Settings dialog, clear selections for any variables that you do not want to be considered in the predictive analysis. You want to pick all the variables that you think may significantly affect the outcome of the activity. For example, in the following image, we've selected all variables since each could impact the outcome for this activity.
4. For this example, accept the default selections for the Confidence and Complexity levels.

The Confidence setting tells the Optimizer to only suggest bypasses that are correct the shown percentage of time. The Complexity setting determines how complex the suggested bypass designs might be.

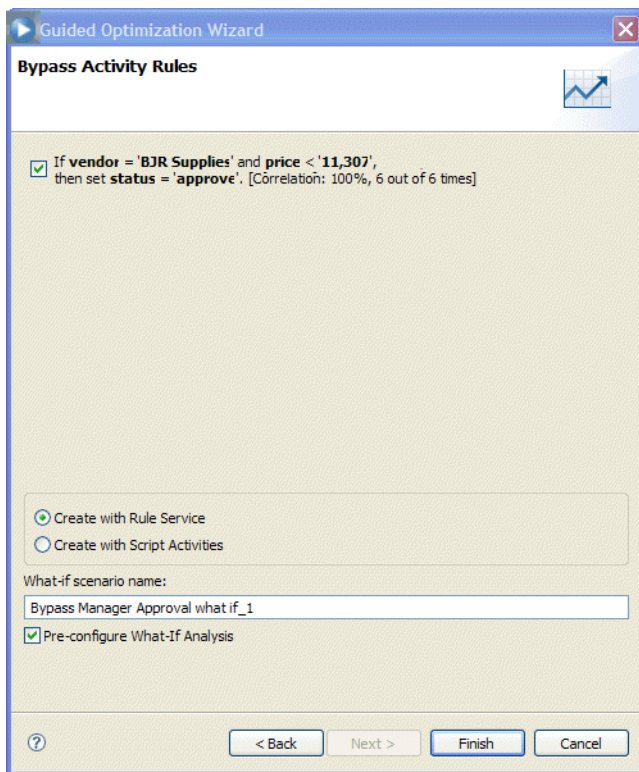


5. Click **Next**.

The Optimizer runs its calculations to find the correlations between the selected variable and the historical data. The analysis results for this example show that the analyzed activity resulted in an approved quote 100% of the time when the vendor was 'BJR Supplies' and the prices was less than '11, 307':



6. Select the **Continue and Bypass Activity** option and then click **Next**.
7. In the Bypass Activity Rules dialog, the wizard displays the rules that it recommends:



Click the checkbox next to the rule to automatically accept quotes that are from BJR Supplies and are less than 11,307. Only when the price matches or exceeds this price, will Manager approval be required for quotes from BJR Supplies.

- In the Bypass Activity Rules dialog, choose between creating a Rule Service or a Script for your process.

An embedded Script causes fewer visible changes to your process, but a Rule Service enables you to tell, just from looking at your diagram, where a rule is used in a process. A Rule Service is also a reusable library item, unlike an embedded Script. In this example, we'll create a Rule Service.

- In the Bypass Activity Rules dialog, select the **Pre-Configure What-If Analysis** checkbox.

The wizard automatically creates a new simulation scenario that incorporates the previously selected rules and then performs a what-if analysis by comparing the simulation of the rules to the historical data for the process.

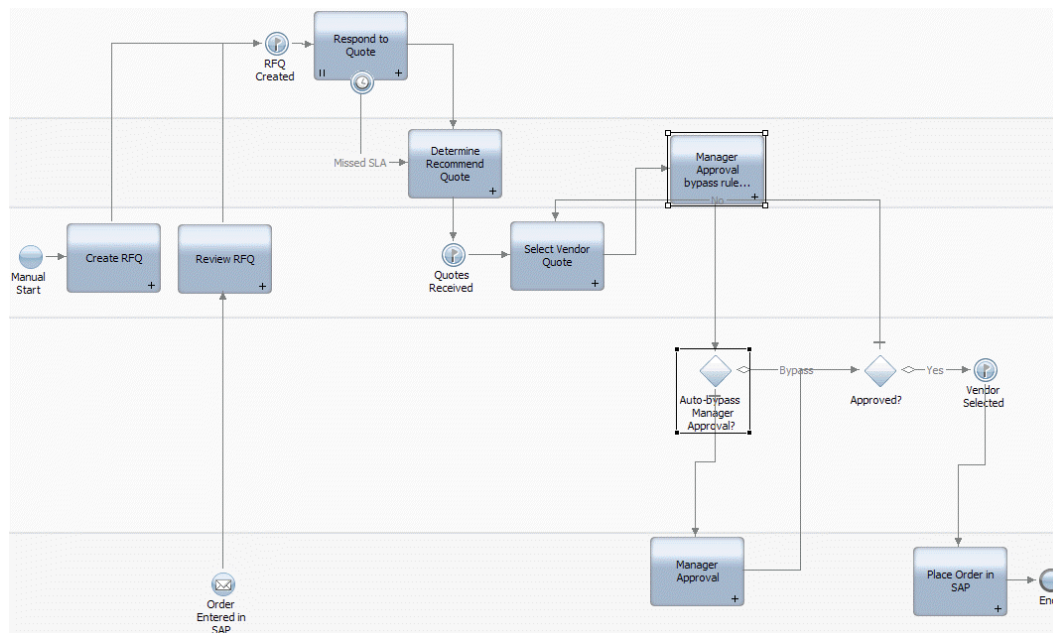
The Pre-configure What-If Analysis option, when selected, makes the appropriate selections in the Analysis Scenarios view, but the analysis does not run until you click the **Calculate** button in the Analysis Scenarios view.



For simulation scenarios created by the Guided Optimization Wizard, the results are more meaningful if the process being analyzed is executed often (high frequency) and is initiated on a regular schedule (flat distribution).

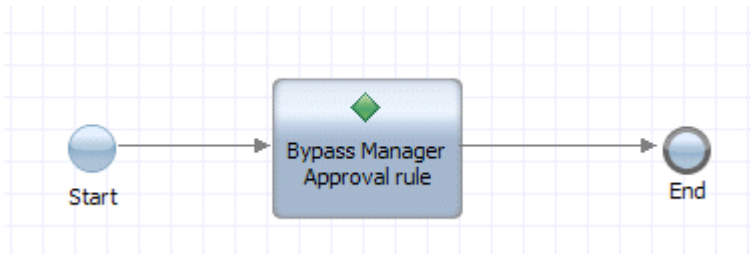
- Click **Finish**.

The Optimizer creates the required components for the Bypass Rule Service and adds them into your process diagram, complete with sequence lines and appropriate component properties (the new components are selected in the following image to highlight them):



To review the changes introduced by the bypass wizard:

1. From the Designer, double-click the **Manager Approval bypass rule service** activity to view the attached Rule Service:



2. Double-click the Bypass Manager Approval Rule Service to view its structure:

The screenshot shows the 'Rules' tab in a software interface. The 'Rule Conditions (IF)' section contains a table with three rows:

	vendor	price	Action Requirement
1	"BJR Supplies"	< '11,307'	Auto-generated
2	-	-	Auto-generated: otherwise, don't aut...
3			

Below the table is the 'Action (THEN)' section, which includes a 'Requirement' field with the value 'Auto-generated' and an 'Action' field containing the following code:

```

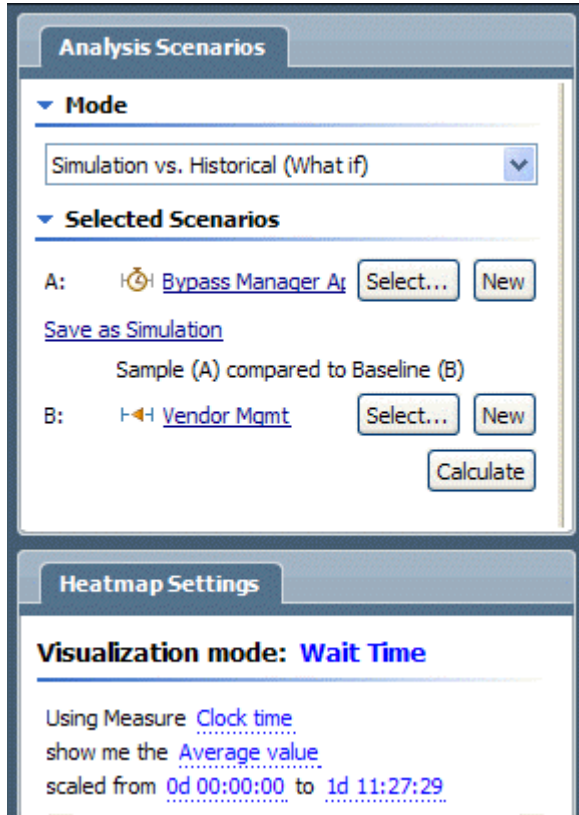
tw.local.outputVar = "approve";
tw.local.bypassOnReturn = true;
  
```

3. Select a rule condition from the table and then click the Action section to view the auto-generated action that each rule runs when it evaluates to true at run time.

Running a Simulation vs. Historical comparison

To run the Simulation vs. Historical (What if) analysis that was automatically generated by the wizard in the preceding sample, follow these steps:

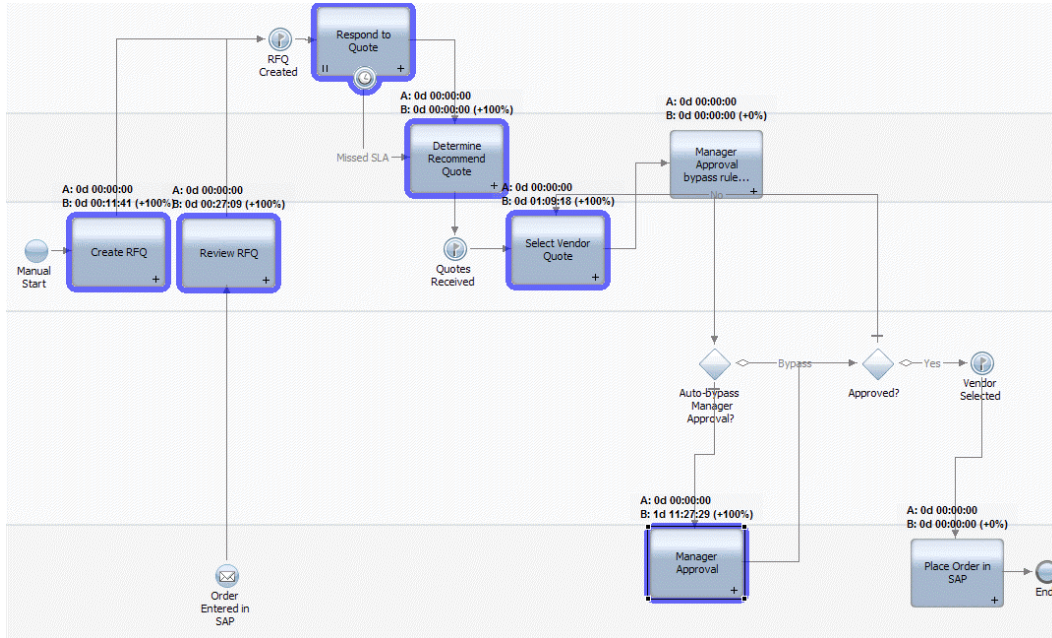
1. Go to the Analysis Scenarios view in the Optimizer to see the settings established by the wizard:



You can see that the simulation data for the revised process (with the built-in service to bypass the approval step) will be compared to the Historical Analysis Scenario that originally revealed the Wait Time issue.

2. Click the **Calculate** button in the Analysis Scenarios view.

The heat map now shows that the Approval Activity is no longer bottlenecked:



You can run similar what-if comparisons by:

1. Creating different versions of your process with additional services or other workarounds.
2. Creating a simulation profile for each version of the process.
3. Creating a Simulation Analysis Scenario in which you can pick and choose from the profiles created in the preceding step.
4. Running a Simulation vs. Historical (What if) comparison using a default Historical Analysis Scenario (like All Available) for the baseline and the Simulation Analysis Scenario from the preceding step as the sample.

Notices and Trademarks

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*Intellectual Property Dept. for WebSphere Software
IBM Corporation
3600 Steeles Ave. East
Markham, Ontario
Canada L3R 9Z7*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. enter the year or years. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.