WebSphere® IBM WebSphere Multichannel Bank Transformation Toolkit

IBM

Version 6.1.1

Migration guide from BTT version 5.2 to BTT version 6.1.1

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 21.

This edition applies to Version 6, Release 1, Modification 1, of *IBM WebSphere MultiChannel Bank Transformation Toolkit* (5724-H82) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send to the following address:

IBM China Software Development Lab
Bank Transformation Toolkit Product
Diamond Building, ZhongGuanCun Software Park, Dongbeiwang West Road No.8,
ShangDi, Haidian District, Beijing 100193 P. R. China

Include the title and order number of this book, and the page number or topic related to your comment.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Migration guide from BTT version 5.2 to BTT version 6.1.1

This migration guide provided by Bank Transformation Toolkit (BTT) version 6.1.1 helps users of BTT version 5.2 move up to the full power and versatility of IBM® WebSphere® Application Server. It provides the migration path to allow an enterprise to reuse much of its existing toolkit application code base and still take advantage of the WebSphere J2EE environment.

IBM recognizes that an enterprise cannot always upgrade its application software all at once. Because the enterprise can decide how much or how little toolkit functionality to retain, BTT version 6.1.1 enables the enterprise to upgrade to a full J2EE environment incrementally. With BTT version 6.1.1, the enterprise can bypass the toolkit functionality and access J2EE components directly upon requests.

BTT version 6.1.1 provides a migration tool that helps you to migrate your applications. The tool can migrate BTT version 5.2 definitions to BTT version 6.1.1 definitions and migrate BTT version 5.2 class packages and APIs to the corresponding BTT version 6.1.1 class packages and APIs.

This migration guide describes the concepts that you need to know before you migrate your applications to BTT version 6.1.1 and the tasks you need to take to roll out the migrations with the migration tool.

Concepts

Before you can perform the migration tasks, you need to understand the concepts that you might meet during the migration process. In this chapter, you can have an overview of the migration, find out how you can benefit from the migration, and get an understanding of the migration process. For detailed information about concepts in the migration from BTT version 5.2 to BTT version 6.1.1, see the following sections.

Migration Overview

BTT version 6.1.1 provides the following functions to help you migrate your applications from BTT version 5.2 to BTT version 6.1.1:

Definition Transformation
This feature transforms the definition files from BTT version 5.2 to BTT version 6.1.1. All the existing definition files are converted or migrated to the corresponding BTT version 6.1.1 definition files. For example, the definition file *dse.ini* is converted to another definition file *btt.xml*.

Java™ Code Migration
Because many components in BTT version 6.1.1 are different from those in BTT version 5.2, the migration tool in BTT version 6.1.1 provides a function to make the previous code migrated from the existing BTT version 5.2 program to BTT version 6.1.1 base programs. This reduces the workload for the application program migration.

Customer Extension Support
This is a key feature of the migration tool. The migration tool is rule-based, which means that the specific migration action depends entirely on the given rules. You can customize your own migration rules to meet the program migration requirements.

Default Migration Rules Provided
BTT version 6.1.1 provides default migration rules for migration from BTT version 5.2 to BTT version 6.1.1. You can leverage the default rules directly or extend it to meet your additional requirements.

Exclusions
UI features here refer to the JSP presentation logic and Swing based presentation logic. In BTT version 6.1.1, Swing based Java client is totally compatible with BTT version 5.2, so no migration is needed. As for JSP presentation logic, you need to migrate it manually.

Constraints
To reduce the complexity of the migration process, the migration tool limits the availability. The migration tool is rule based, so you are not able to migrate the migration scenarios that the rules cannot represent.

Benefits gained through the migration

Bank Transformation Toolkit version 6.1.1 provides support for distributed architecture. The adoption of a distributed architecture brings some immediate benefits to the system planner and developer.

You can benefit from the migration in the following aspects:

- Scalability improves by an order of magnitude. A version 6.1.1 system can scale from a small or medium enterprise having a single server to large enterprises with server farms distributed across multiple regional data centers.
- Performance can be tuned by balancing hardware and network usage against the expected user base. Rational® Application Developer and WebSphere Integration Developer provide many tools and guides for performing this tuning.
- Industry standards such as JCA, EJB, JMS, and Web services replace many previous architectures and facilities in the toolkit. This adds flexibility to expanding enterprise applications using third-party components that are built with the same industry standard architectures.
- To enhance its flexibility, the toolkit maintains much of its previous external design philosophy. It further adds more configurability to this mix by allowing its deployment information to be configured. Scaling and performance tuning are all performed using configuration settings rather than Java code changes.
- Platform flexibility is provided using this configurability as well. Version 6.1.1 can execute using Rational Application Developer for small scale deployments or it can make use of the enhanced features of WebSphere Integration Developer to expand the available options to much larger enterprise deployments.

For more information about the benefits brought by BTT version 6.1.1, see the Solution Architecture Document for BTT version 6.1.1.

Migration tool architecture and support

Bank Transformation Toolkit version 6.1.1 provides a migration tool to help you migrate your toolkit applications developed with pervious versions of the toolkit to the new version 6.1.1 architecture.

A high scalability architecture has been applied to this rule based migration tool. Under this architecture, you can easily extend the toolkit to meet your specific requirements. It is much like a generic migration toolkit, because it can be used in most Java application migration cases besides BTT. Therefore, this toolkit is much better than any other similar migration toolkit in BTT history. The following figure shows the architecture of this rule based migration tool.

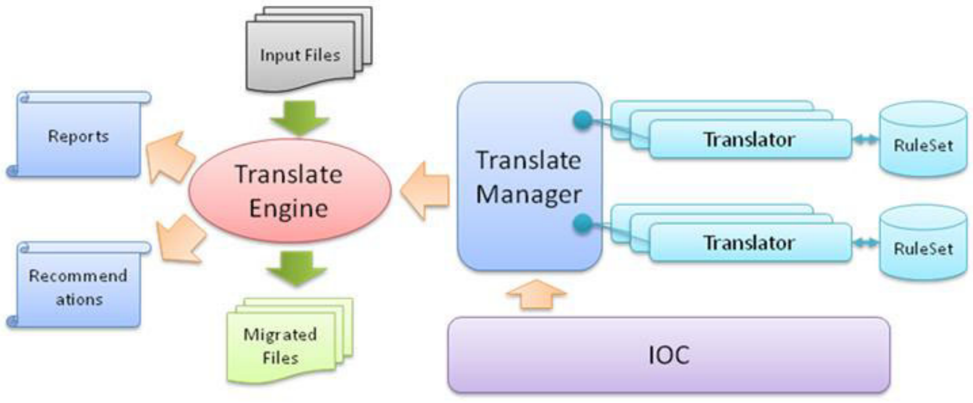


Figure 1. Migration tool architecture

The specific functional descriptions of the migration tool are listed as follows:

- Migrate Java code:

  From BTT version 5.2 to BTT version 6.1.1, there are inevitable API changes to the same functions. It is time-consuming to replace the old API calls with the new ones. Java code migration is the major function of the migration tool.
- Migrate the file *dse.ini*:

  Dse.ini contains most of the configurations for all the BTT components. New components are added in version 5.x and version 6.x. In BTT version 6.1.1, the *dse.ini* file is reorganized. The migration tool helps you to migrate this file.
- Migrate other XML files:

  In BTT, context, formatter, services, data element, and operation are externalized into XML files. The referenced package names and class names of these XML files are different in different versions. Migrating these XML files is another major function of the migration tool.
- Generate migration report:

  After you run the migration tool, a report is generated to record the tasks the migration tool performed.
- Generate migration recommendation:

  If some cases cannot be easily migrated by the toolkit, the migration tool gives recommendations on these cases to alert you. Then, you can follow these recommendations to migrate your applications.

The migration tool is rule-based, which means that the migration depends on the given rules. You can change the rule set according to your needs. You can also use this tool to migrate other Java applications.

Migration procedures

Before you start migrating your applications from BTT version 5.2 to BTT version 6.1.1, you need to follow the migration procedures.

The whole migration process consists of the following four phases:

- Phase 1: Analysis and preparation
- Phase 2: Customizing the migration tool
- Phase 3: Migrating your applications

- Phase 4: Adding business logic and fixing errors

The phases are iterative and contain tasks that might be refined during the project life cycle.

Phase 1: Analysis and preparation

In this phase, you need to analyze the differences between BTT version 5.2 and BTT version 6.1.1, especially the application logic layer.

Based on the documentation of the existing application system, you can identify the parts that can be migrated automatically and the parts that cannot be migrated automatically, and then make necessary preparation for the migration. To get well prepared for the migration, you need to do the following work:

1. Gather all the definition files of the existing application system version 5.2. The migration tool can migrate the definition files on the server side from version 5.2 to version 6.1.1. The migration tool can process the definition contents based on the given rules. BTT version 6.1.1 provides default migration rules, but if there are any special tag definitions that are not included in the default rules, the migration tool might not be able to migrate them successfully or properly. In this case, you need to customize the migration tool rules to extend the process to cover those special tag definitions for the application or migrate them manually later.
2. The migration tool can migrate the business logic Java code of the application to BTT version 6.1.1 directly. BTT version 6.1.1 uses the processor, operation, and operation step on the server side of BTT. There is much enhancement and refinement in the architecture of BTT version 6.1.1, so it has different package names, class names, and changes of APIs. The migration tool can migrate BTT version 5.2 classes and APIs to the corresponding BTT version 6.1.1 classes and APIs.
3. Migrate the BTT version 5.2 invoker. BTT version 5.2 leverages the invoker framework to invoke the business logic. BTT version 5.2 leverages EJB as the business logic implementation. The invoker of BTT version 5.2 is mainly used for EJB calling from the BTT runtime server. In BTT version 6.1.1, the architecture is improved. The request from the channel side will directly call the BTT processor or operation in the BTT runtime container, where you can handle some channel aware logic or business logic. If you want to call EJB, JMS, and BPEL, BTT provides you an invoker to invoke them. BTT version 5.2 invoker framework is not used in BTT version 6.1.1, because a more powerful and ease-of-use invoker framework is implemented in BTT version 6.1.1. You should migrate your applications to adopt the new BTT version 6.1.1 invoker.
4. BTT version 6.1.1 provides more powerful Context and Formatter, and the API usage is different with that in BTT version 5.2. The migration tool of BTT version 6.1.1 can help you migrate the Context, Formatter, and APIs or provide recommendations for the migration.
5. Migrate BTT version 5.2 services. The supported services of BTT version 6.1.1 on the server side support all of the BTT version 5.2 services. The migration tool helps to migrate the package names, class names, APIs, and so on. If you have some customized services, you can customize the migration rule to meet your migration requirements.
6. The event mechanism is changed to fit version 6.1.1 framework so that it can be distributable and spans different servers. You must do some manual migration to run the BTT version 5.2 event in BTT version 6.1.1.
7. There are two kinds of client applications in BTT version 5.2. There is no change to the Swing based Java client, so you do not need to migrate it to BTT

version 6.1.1. For JSP/HTML based client, the migration tool does not support its migration, so manual migration is needed.

8. Migrate the BTT trace. BTT version 6.1.1 enhances the trace features to support more powerful functions and comprehensive APIs. BTT version 6.1.1 trace is completely compatible with BTT version 5.2 trace. If you do not want to use the new trace framework, the old trace can still work in BTT version 6.1.1 runtime container. If you want to use the new trace, use the migration tool to achieve the trace by defining specific migration rules.
9. If the future application only runs on WebSphere Application Server, use Rational Application Developer to migrate the application. If the future application runs on WebSphere Process Server, use WebSphere Integration Developer to migrate the application.

Phase 2: Customizing the migration tool

In this phase, you need to customize the migration tool to extend its functionality.

To customize the migration tool, take the following steps:

1. Customize the Java code migration rules. Because the migration tool is based on migration rules, if you need extend the BTT functionality, you should customize the migration rules to meet your special requirements.
2. Customize the definition file migration rules. Because the migration tool is based on migration rules, if you need to extend the BTT functionality and have some special tags in the definitions, you should customize the migration rules to meet these special requirements.

Phase 3: Migrating your applications

In this phase, you can start migrating your applications with the migration tool.

To migrate your applications from BTT version 5.2 to BTT version 6.1.1, take the following steps:

1. Replace the BTT version 5.2 jar files with BTT version 6.1.1 jar files, and change the build path of the project to the new jar files.
2. Import the application that you want to migrate to the workspace as a project, and configure the migration tool. The migration tool will migrate the application directly based on the original project.
3. Migrate the configuration file *dse.ini* to *btt.xml*. In BTT version 5.2, the entrance of the configuration file is *dse.ini*, but in BTT version 6.1.1, it is renamed as *btt.xml*. See the following figure.

BTT5.2 dse.ini
Update/Content
BTT6.1 btt.xml

Figure 2. Migrating the configuration file

4. Migrate the definition files of the context, the data element, the type data, and the format.
5. Migrate the business logic. BTT version 6.1.1 uses process, operation, and operation step as channel aware business logic, which is similar with BTT version 5.2. In BTT version 6.1.1, the architecture is refined, and some components are re-designed, so the package names, class names, and some APIs in BTT version 6.1.1 are different from those in BTT version 5.2. The migration tool will automatically migrate these differences.

6. Manually migrate the components that cannot be completely migrated by the migration tool, based on the recommendation provided by the migration tool.

Phase 4: Adding the business logic and fixing errors

After you have migrated the applications from BTT version 5.2 to BTT version 6.1.1, you must add the business logic and fix the errors.

To add the business logic and fix errors, take the following steps:

1. Fix the Jave build errors. There might be some API mismatch problems between BTT version 5.2 and BTT version 6.1.1. The migration tool can handle most of them or provide recommendations for manual migration. If not, please leverage RAD or WID to fix the problem. For detailed information about API usage, see the related tasks.
2. Do JSP migration manually with the help of RAD or WID. The migration tool does not support JSP migration.

Tasks

Bank Transformation Toolkit version 6.1.1 provides a migration tool to help you migrate your toolkit applications that are developed with BTT version 5.2 to the new version 6.1.1 architecture.

After you get to know the concepts in the migration process, you can use the migration tool to migrate your applications from BTT versoin 5.2 to BTT version 6.1.1. In this chapter, you can find information about how to set up the migration tool and how to use the migration tool to perform the migration tasks.

Setting up the migration tool

Before you can migrate your applications from BTT version 5.2 to BTT version 6.1.1, you need to set up the migration tool.

If you have installed the toolkit before you install IBM Rational Application Developer or IBM WebSphere Integration Developer on your workstation, you need to copy the following plug-in files to the $D(RAD)/plugins or the $D(WID)/plugins directory manually after you install IBM Rational Application Developer or IBM WebSphere Integration Developer:

- **com.ibm.btt.tools.migration_6.1.1**
- **com.ibm.btt.core_6.1.1**

To set up the migration tool, take the following steps:

1. Start IBM Rational Application Developer or IBM WebSphere Integration Developer.
2. Import the project that you want to migrate.
3. Click **Window** → **Preferences** on the menu bar of IBM Rational Application Developer or IBM WebSphere Integration Developer.
4. In the dialog box that is displayed, select **BTT Migration** in the left panel.
5. In the right panel of the dialog box, select the migration rule definition file that you want to apply. In the migration tool plug-in, there is a **config** category, and you can find BTT version 5.2 to BTT version 6.1.1 migration rule definition files under the related category.
6. Click **OK**.
7. In the **Project Explorer** view, right-click the BTT definition file and select **BTT Migration**.

The migration report and migration results will be displayed in the **Project Explorer** view if there are any.

Using the migration tool

After you set up the migration tool, you can migrate your applications with the tool.

The migration tool can perform the following tasks to help you migrate your applications:

- Migrate the file *dse.ini* from BTT version 5.2 to the file *btt.xml* in BTT version 6.1.1.
- Migrate the definition files in BTT version 5.2 to the definition files in BTT version 6.1.1, including context definitions, formatter definitions, and so on.
- Migrate the application Java codes, including package names, class names, APIs, and so on from BTT version 5.2 to BTT version 6.1.1.

The migration tool is rule-based. It migrates BTT version 5.2 applications to BTT version 6.1.1 based on the given migration rules. You can customize the migration rules to meet your special requirements.

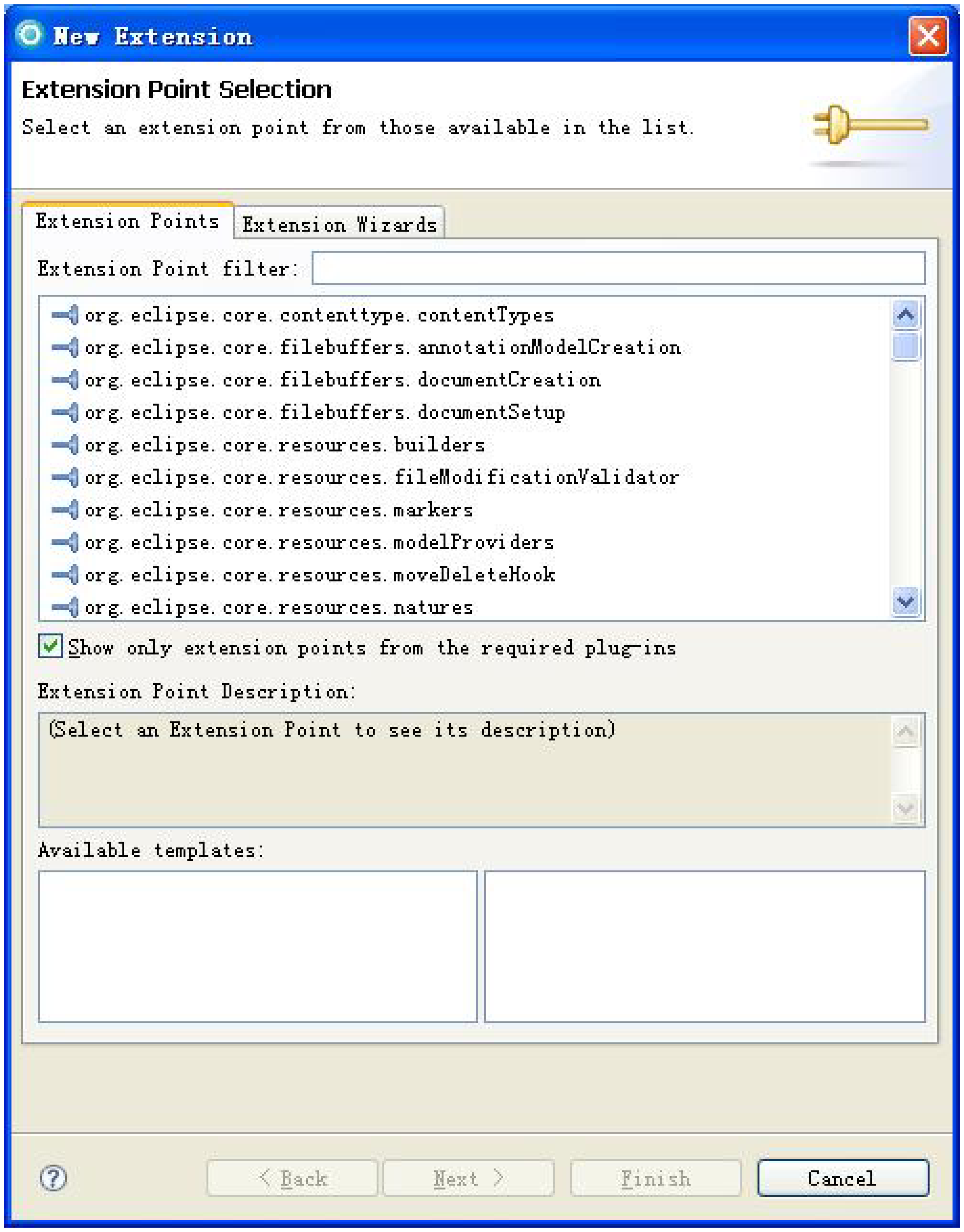
The migration tool is extendable. It provides three extension points, by which you can implement your own migration rule engine to meet your special migration requirements.

Adding translators

You can extend the migration tool and add your own translators in your plug-in project.

To add a translator, take the following steps:

1. Create a plug-in project.
   a. In menu bar of IBM Rational Application Developer, click **File** → **New** → **Others...**.
   b. Expand **Plug-in Development**, select **Plug-in Project**, and click **Next**.
   c. Enter the name of the project, and click **Next**.
   d. In the dialog box that is displayed, click **Finish**, and the plug-in project is created.
2. Add new extensions.
   a. Click the **Extensions** tab.
   b. Under the **Extensions** tab, click **Add...**. The **New Extension** dialog box is displayed. See the following figure.


New Extension

Extension Point Selection

Select an extension point from those available in the list.

Extension Points | Extension Wizards

Extension Point filter:

- org.eclipse.core.contenttype.contentTypes
- org.eclipse.core.filebuffers.annotationModelCreation
- org.eclipse.core.filebuffers.documentCreation
- org.eclipse.core.filebuffers.documentSetup
- org.eclipse.core.resources.builders
- org.eclipse.core.resources.fileModificationValidator
- org.eclipse.core.resources.markers
- org.eclipse.core.resources.modelProviders
- org.eclipse.core.resources.moveDeleteHook
- org.eclipse.core.resources.natures

[✓] Show only extension points from the required plug-ins

Extension Point Description:

(Select an Extension Point to see its description)

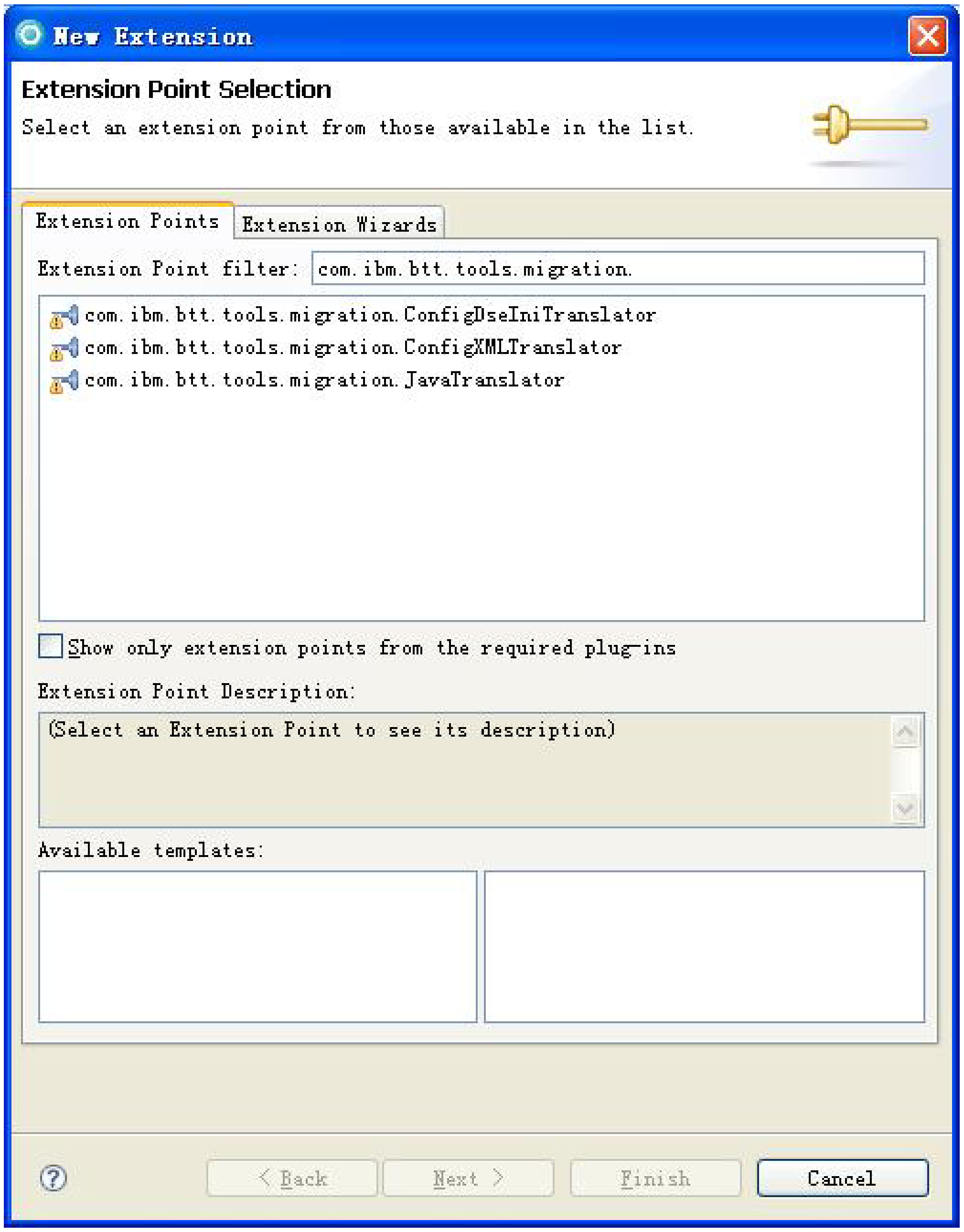
Available templates:

< Back | Next > | Finish | Cancel

Figure 3. Adding new extensions

c. Clear the **Show only extension points from the required plug-ins** check box.

d. Select the extension points, and click **Finish** to add the selected extension points. There are three extension points available: `ConfigDseIniTranslator`, `ConfigXMLTranslator`, and `JavaTranslator`. See the following figure.



New Extension

Extension Point Selection

Select an extension point from those available in the list.

Extension Point filter: com.ibm.btt.tools.migration.

com.ibm.btt.tools.migration.ConfigDseIniTranslator
com.ibm.btt.tools.migration.ConfigXMLTranslator
com.ibm.btt.tools.migration.JavaTranslator

Show only extension points from the required plug-ins

Extension Point Description:

(Select an Extension Point to see its description)

Available templates:

Figure 4. Selecting extension points

3. Create the extension instance (taking JavaTranslator for example here).
   a. Right-click the extension point `com.ibm.btt.tools.migration.JavaTranslator`, and select **New** → **Translator**. Then the extension instance is created. See the following figure.

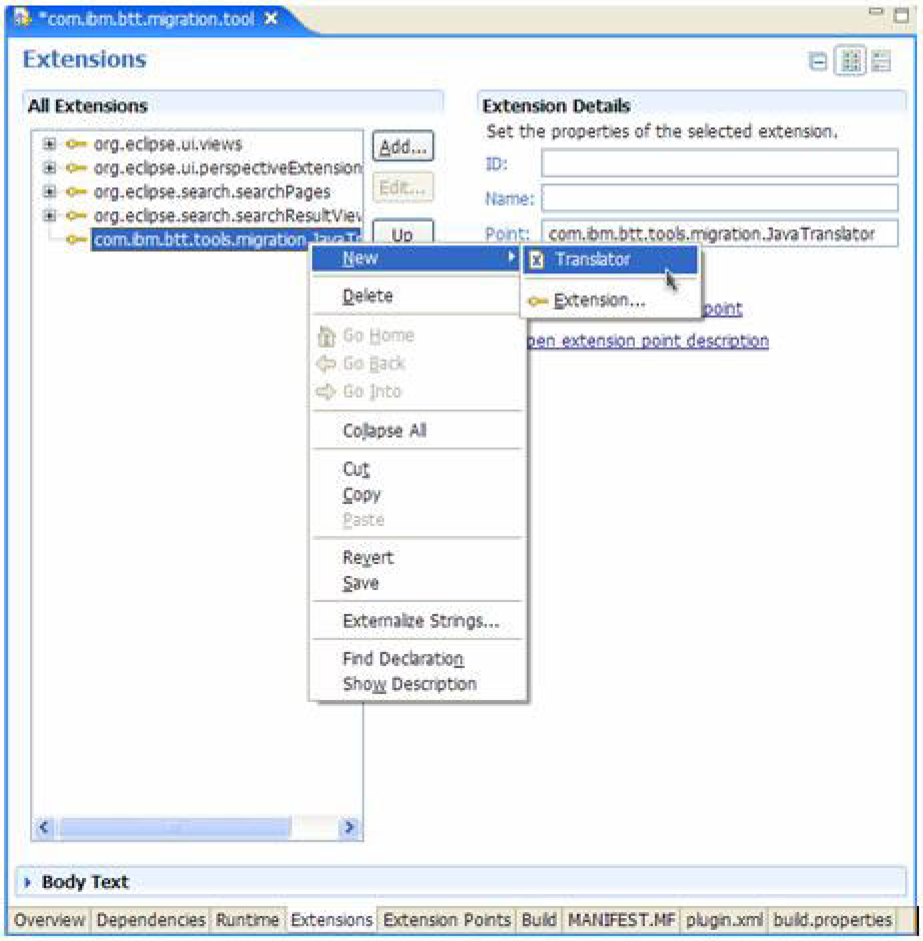


Figure 5. Creating extension instance

4. Implement the corresponding translator interface.
   a. Open the panel **Extension Element Details** on the right.
   b. Select the extension type from the **type** dropdown list, and then click **Browse...**. See the following figure.

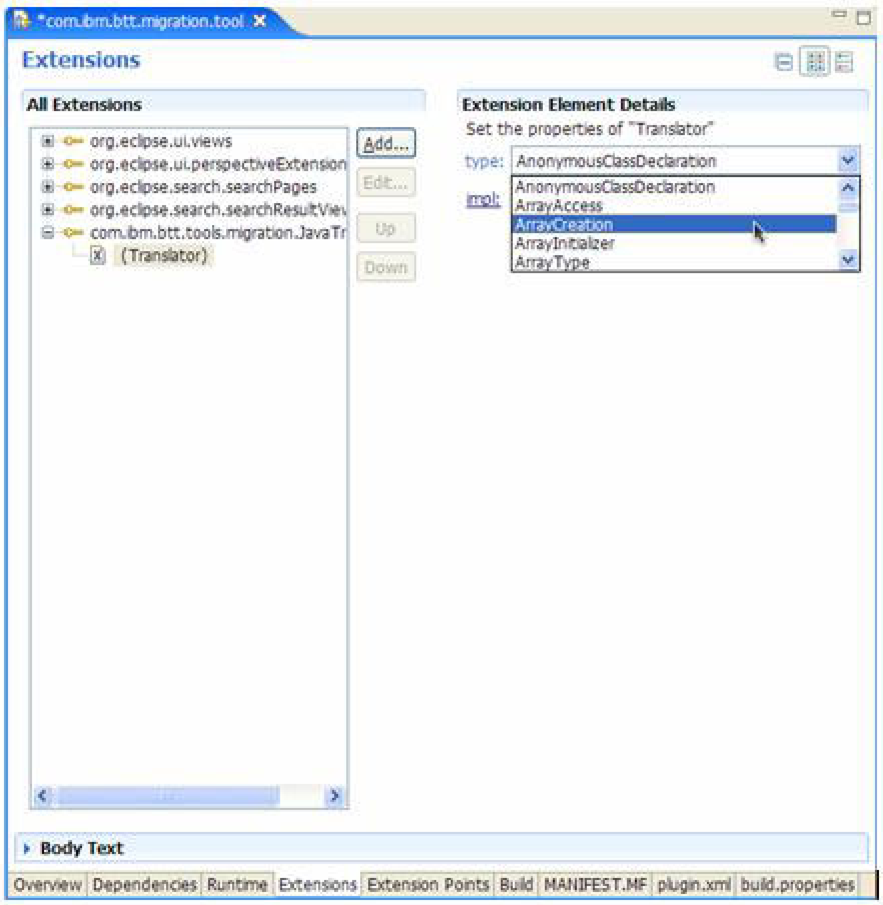

*com.ibm.btt.migration.tool
Extensions
All Extensions
org.eclipse.ui.views
org.eclipse.ui.perspectiveExtension
org.eclipse.search.searchPages
org.eclipse.search.searchResultViev
com.ibm.btt.tools.migration.JavaTr
(Translator)
Add...
Edit...
Up
Down
Extension Element Details
Set the properties of "Translator"
type: AnonymousClassDeclaration
impl:
AnonymousClassDeclaration
ArrayAccess
ArrayCreation
ArrayInitializer
ArrayType
Body Text
Overview
Dependencies
Runtime
Extensions
Extension Points
Build
MANIFEST.MF
plugin.xml
build.properties

Figure 6. Extension element details

Note: **ConfigDseIniTranslator** and **ConfigXMLTranslator** do not have this attribute.

Adding rules

The migration tool provides three types of migration: Java migration, BTT definition XML file migration and dse.ini file migration. The migration is implemented according to the configured migration rules.

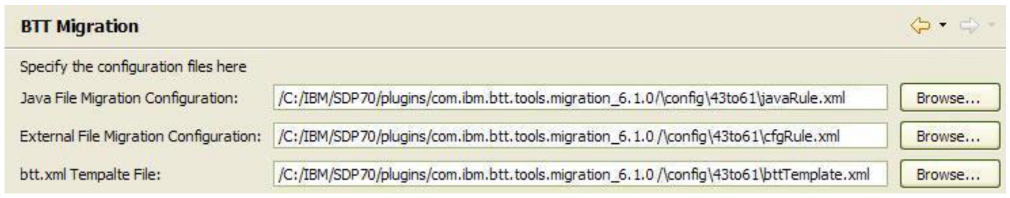
Two series of rule sets are provided: version 4.3 to version 6.1.1 migration rule set and version 5.2 to version 6.1.1 migration rule set. The rule sets are in the $D(RAD)\plugins\com.ibm.btt.tools.migration_6.1.1\config directory. By default, the 4.3 to 6.1.1 rule set is applied.

There are three configuration files for this Rule-based Migration Tool:
- javaRule.xml: rules for Java code migration
- cfgRule.xml: rules for XML and dse.ini file migration
- bttTemplate.xml: btt.xml template.

If the default rules can not meet your migration needs, you can add new rules and configure them as the migration rules in the Preference setting:

To configure your own migration rules, do the following:

- Click **Window** → **Preferences...** in Rational Application Developer.
- In the dialog box that pops up, click **BTT Migration** in the left panel.
- In the right panel, click **Browse...** to select your rules.



BTT Migration

Specify the configuration files here

Java File Migration Configuration:	/C:/IBM/SDP70/plugins/com.ibm.btt.tools.migration_6.1.0/\config\43to61\javaRule.xml	Browse...
External File Migration Configuration:	/C:/IBM/SDP70/plugins/com.ibm.btt.tools.migration_6.1.0/\config\43to61\cfgRule.xml	Browse...
btt.xml Tempalte File:	/C:/IBM/SDP70/plugins/com.ibm.btt.tools.migration_6.1.0/\config\43to61\bttTemplate.xml	Browse...

Migrating Java code

Java code migration can be done in two ways: Java file level migration and package level migration.

To migrate Java code, take the following steps:

1. Select the target Java files or packages, and right-click them.
   a. If you are doing Java file level migration, click **BTT Migration** → **Migrate Java File** in the menu that is displayed. See the following figure.

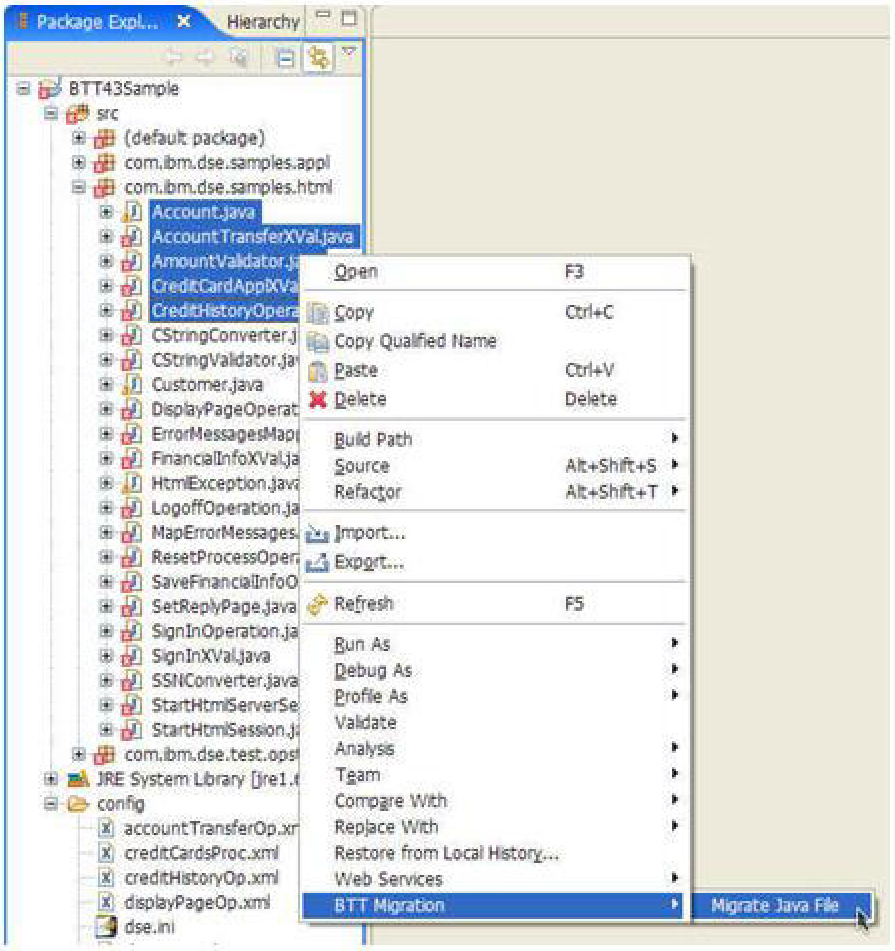


Figure 7. Migrating Java code on the file level

b. If you are doing package level migration, click **BTT Migration** → **Migrate Java Files**. See the following figure.

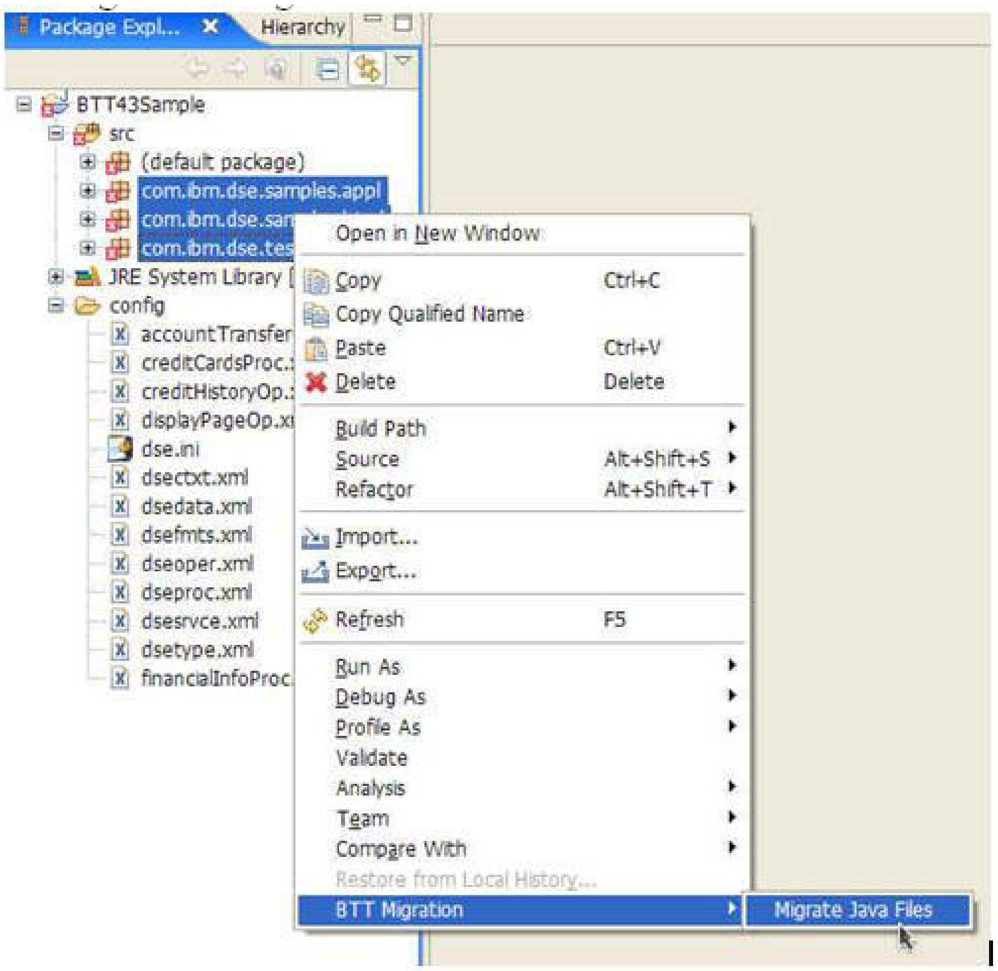


Figure 8. Migrating Java code on the package level

2. Then the Java code is migrated. The migrated Java code will replace the original Java code.

Migrating the *dse.ini* file

To migrate the *dse.ini* file, take the following steps:

1. Right-click the *dse.ini* file.
2. In the menu that is displayed, select **BTT Migration** → **Migrate dse.in**. The *dse.ini* file is then migrated to the **config.migrated** folder.

Migrating the XML files

To migrate the XML file, take the following steps:

1. Right-click the XML file.
2. In the menu that is displayed, select **BTT Migration** → **Migrate XML file**. Then the XML file is migrated to the config.migrated folder.

Note: Batch migration is also supported for the migration of XML files.

Generating reports and recommendations

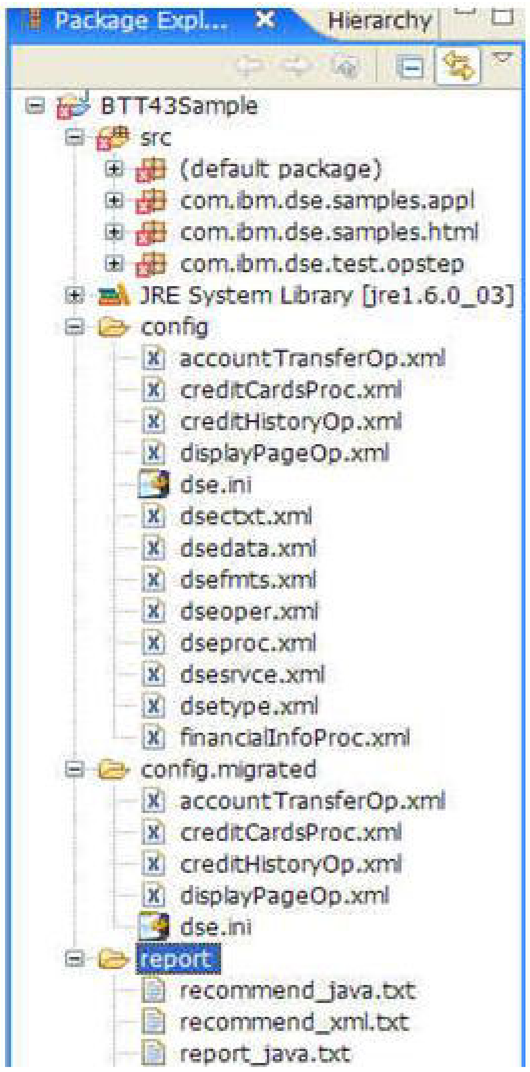
After the migration, reports are generated to record what has been done during the migration process.

There are separate reports for Java code migration and BTT XML definition file migration.

Note: The report and recommendation files will not be generated in dse.ini migration.

Recommendations for special migration cases are also generated to guide you through the manual migration.

All the generated report files and recommendation files are stored in a folder named **report**. See the following figure.



```
BTT43Sample
  src
    (default package)
    com.ibm.dse.samples.appl
    com.ibm.dse.samples.html
    com.ibm.dse.test.opstep
  JRE System Library [jre1.6.0_03]
  config
    accountTransferOp.xml
    creditCardsProc.xml
    creditHistoryOp.xml
    displayPageOp.xml
    dse.ini
    dsectxt.xml
    dsedata.xml
    dsefmts.xml
    dseoper.xml
    dseproc.xml
    dsesrvce.xml
    dsetype.xml
    financialInfoProc.xml
  config.migrated
    accountTransferOp.xml
    creditCardsProc.xml
    creditHistoryOp.xml
    displayPageOp.xml
    dse.ini
  report
    recommend_java.txt
    recommend_xml.txt
    report_java.txt
```

Figure 9. Report folder

Context Usage Migration

This topic introduces how you can migrate BTT version 5.2 Context to BTT version 6.1.1 Context.

You do not need to change the context, type, and data definitions of BTT version 5.2 when you migrate them to BTT version 6.1.1, because these definitions are the same in BTT version 5.2 and BTT version 6.1.1. In BTT version 6.1.1, a class table tag for context implementation class and an initializer tag are added in the CHA configuration in the BTT XML definition file, such as the file *btt.xml*.

The APIs in BTT version 5.2 and BTT version 6.1.1 are almost the same except the way of constructing the context instance. In BTT version 6.1.1, the context is constructed from ContextFactory. BTT version 5.2 CHA definition tag *startMode* (value ="PersistenceShared" or "MemoryShared") is replaced by the tag *isPersistenceEnabled* (value="yes" or "no"). An example of migrating BTT version 5.2 context to BTT version 6.1.1 context is as follows:

1. Creating BTT version 5.2 context.

   ```
   Context ctxt = (Context) Context.readObject("myContext",true);
   Context ctxt2 = new com.ibm.dse.base.Context();
   ctxt2.setName("myContext2");
   ```

2. Creating BTT version 6.1.1 context.

   ```
   Context diiTestCtx = ContextFactory.createContext("myContext", true);
   Context ctxt2 = new com.ibm.btt.base.Context();
   ctxt2.setName("myContext2");
   ```

The following is an example of BTT version 6.1.1 CHA configuration. Please note the bold text that is different from BTT version 5.2 CHA configuration.

```
<kColl id="cha">
    <!-- indicates the implementation class of context-->
    <kColl id="classTable">
      <field id="context" value="com.ibm.btt.cha.ejb.RemoteContextImpl" />
    </kColl>

    <!-- indicates the initialization class of CHA-->
    <field id="initializer" value="com.ibm.btt.base.ContextInitializer" />
    <!-- indicates the external file of CHA-->
    <field id="extFile" value="dsectxt.xml" />

    <!-- indicates the JNDI looking up properties of CHA-->
    <field id="ejbInitialContextFactory"
      value="com.ibm.websphere.naming.WsnInitialContextFactory" />
    <field id="EJBProviderURL" value="iiop://localhost:2809" />

    <!-- indicates the home interface of CHAEJB-->
    <field id="CHASessionLocalHome"
      value="java:comp/env/ejb/CHASession" />
    <field id="CHAInstanceLocalHome"
    value="java:comp/env/ejb/CHAInstance" />
    <field id="CHAControlLocalHome"
    value="java:comp/env/ejb/CHAControl" />
    <field id="CHASessionHome"
    value="ejb/com/ibm/btt/cha/ejb/CHASessionHome" />
    <field id="CHAInstanceHome"
    value="ejb/com/ibm/btt/cha/ejb/CHAInstanceHome" />

    <!-- indicates the initialized context tree of CHA-->
    <field id="initTailContextName" value="branchServer" />

    <!-- indicates whether to cleanup the context instance from the DB tables-->
    <field id="cleanupCHAServer" value="true" />

    <!-- indicates whether to use local interface when invoke CHAEJB-->
    <field id="isLocalCall" value="true" />

    <!-- indicates whether the remote CHA supports persistence-->
    <field id="isPersistenceEnabled" value="false" />
    <!--field id="startMode" value="MemoryShared"/-->
</kColl>
```

Formatter Usage Migration

There are three approaches to migrate the Formatter component.

1. Use the old Formatter.

   If you want to use the old Formatter, you need to migrate the configuration that is related to the Formatter in the file *dse.ini* in BTT version 5.2 to the file *btt.xml* in BTT version 6.1.1 and keep all the other code the same as before.
2. Use the new formatter.

   If you want to use the new Formatter, you leverage the new features provided by the new Formatter component. Fist of all, you must migrate all the implemented FormatElement that extends from *com.ibm.btt.base.FormatElement* or its subclasses. You need to rewrite the FormatElement to make it extends from *com.ibm.btt.format.CompositeFormat*, *com.ibm.btt.format.FieldFormat*, or *com.ibm.btt.format.BaseDecorator*. Then, you must change the method call from *FormatElement.format(DataElement)/FormatElement.format(Context)* or *FormatElement.unformat(DataElement)/FormatElement.unformat(Context)* to *com.ibm.btt.format.FormatElement.format(ReadAdapter)* or *com.ibm.btt.format.FormatElement.unformat(WriteAdapter)*.
3. Make the old Formatter and the new one coexist in the system.

   If you have the old Formatter and the new one coexist in the system, you leverage the new features provided by the new Formatter, while keeping the existing code not modified. For detailed information, see Integrating with other BTT components.

Invoker Migration

This topic introduces how you can migrate BTT version 5.2 invoker to BTT version 6.1.1 invoker.

BTT version 5.2 invoker is a hard coded EJB invocation framework. BTT version 6.1.1 provides a common invocation framework for POJO, EJB, Web Service, and JMS. These two frameworks are totally different. The API usage and configuration are also totally different. The EJB invocation information that is defined in the resource bundle file can be mapped to the EJB invoker definition in the file *invoker.xml*. BTT version 6.1.1 invoker is decoupled with BTT context and formatter. The usage of BTT version 6.1.1 invoker is simple and flexible comparing with the usage of BTT version 5.2 invoker. For detailed information about BTT version 6.1.1 invoker, see Invoker.

Event Migration

This topic introduces how you can migrate BTT version 5.2 Event to BTT version 6.1.1 Event.

You can migrate BTT version 5.2 Event to BTT version 6.1.1 Event by defining the Event manager server operation in the operation definition file. See the following example:

```
<operation id="eventManagerServerOperation"
    implClass="com.ibm.btt.event.EventManagerServerOperation">
<refFormat name="csReplyFormat" refId="EventManagerCSFormat" />
</operation>
```

Session Management Migration

This topic introduces how you can migrate BTT version 5.2 Session Management to BTT version 6.1.1 Session Management.

BTT provides the following utility method in the *com.ibm.btt.sm.CSSessionHandler class*:

Public static void *markSessionExpired(HttpSession aSession)* throws BTTSMException. After this method is called, the session will be marked as expired and will be removed at the end of operation. You need to call this method in the logoff operation. Then, the session context will be cleaned and the HTTP session object will be destroyed after you log off the application.

Trace Migration

This topic introduces how you can migrate BTT version 5.2 trace to BTT version 6.1.1 trace.

The original trace API of BTT version 5.2 can be still used in BTT version 6.1.1. BTT version 6.1.1 trace facility can automatically map the old API to the new API of BTT version 6.1.1 trace facility. BTT version 6.1.1 traces by package, but the original trace is by the Component ID in BTT version 5.2. BTT version 5.2 trace can trace to multiple targets at the same time, but BTT version 6.1.1 supports tracing to one target at the same time only when the BTTLogFactoryImplementClass is set.

The implementation of trace to File is changed to using Java util API, so the original parameter of trace to File is not supported in BTT version 6.1.1. The original function and implementation of trace to Display is kept in BTT version 6.1.1, so the parameter for trace to Display is still supported.

The original 'HML' trace level is not supported in BTT version 6.1.1. The original trace type is mapped to a new trace level. See the following table.

Table 1. Trace level map

BTT 6.1.1 trace level	BTT 5.2 trace type	WAS trace level	Common logging	Log4J
FATAL	Severe	Fatal	Fatal	Fatal
ERROR	Error	Severe	Error	Error
WARN	Warning	Warning	Warn	Warn
INFO	Information Display	Info	Info	Info
DEBUG	Debug AllTypes	Detail*	Debug	Debug

An example of Migrating BTT version 5.2 trace to BTT version 6.1.1 trace is as follows:

BTT version 5.2 trace configuration

```
<kColl id="traces">
    <field id="initializer" value="com.ibm.btt.base.TraceInitializer" />
    <field id="traceToFile" value="yes" />
    <field id="traceToDisplay" value="yes" />
    <field id="traceToWAS" value="yes" />
    <field id="traceWindowTitle" value="Server Trace" />
    <field id="showOriginator" value="yes" />
    <field id="showWarningMessage" value="no" />
    <field id="traceLevels" value="debug" />
    <field id="traceFileName" value="C:\dse\dselog.txt" />
    <field id="traceMaxLogFiles" value="5" />
    <field id="font" value="monospaced" />
    <field id="createBackup" value="yes" />
    <field id="fileNumberOfLines" value="4000" />
    <field id="displayNumberOfLines" value="2000" />
    <field id="linesOfBuffer" value="7000" />
    <field id="lineLength" value="200" />

    <kColl id="requestersComponents">
      <traceRequester id="#CHA" trace="yes" traceLevels="HML" traceTypes="FATAL" />
      <traceRequester id="#CS" trace="yes" traceLevels="HML" traceTypes="DEBUG" />
    </kColl>
</kColl>
```

BTT version 6.1.1 trace configuration after migration

```
<kColl id="traces">
    <field id="initializer" value="com.ibm.btt.base.TraceInitializer" />
    <field id="traceTargetFactoryImplClass" value="com.ibm.btt.base.BTTLogFactoryToDisplayImp" />
    <field id="displayNumberOfLines" value="2000" />
    <kColl id="requestersComponents">
      <traceRequester id="com.ibm.btt.base.*" trace="yes" traceLevels="FATAL" />
      <traceRequester id="com.ibm.btt.channel.*" trace="yes" traceLevels="DEBUG" />
    </kColl>
</kColl>
```

BTT version 5.2 trace application code

```
if (Trace.doTrace(Constants.CHACOMPID,Trace.High,Trace.Debug))
    Trace.trace(Constants.CHACOMPID,Trace.High,Trace.Debug,Settings.getTID(),
              " CHA Debug .........");
if (Trace.doTrace(Constants.CHACOMPID,Trace.High,Trace.Information))
    Trace.trace(Constants.CHACOMPID,Trace.High,Trace.Information,Settings.getTID(),
             "CHA info .........");
```

BTT version 6.1.1 trace application code after migration

```
BTTLog   log=BTTLogFactory.getLog("com.ibm.btt.base.LocalContextImp");
If (log.doDebug())
  log.debug("CHA Debug......");
If (log.doInfo())
  log.info("CHA info......");
```

References

For the references related to migration from BTT version 5.2 to BTT version 6.1.1, see the following topics in this chapter.

Manual migration

The migration tool cannot migrate everything from BTT version 5.2 to BTT version 6.1.1 automatically. You must do some manual migration, including:

- Some new features in BTT version 6.1.1, such as Web 2.0, Rich Client, and so on. The base functionality of the migration tool only migrates the existing applications to BTT version 6.1.1 and makes sure that they can run in BTT version 6.1.1. You must re-design your applications and implement them manually.
- The existing services that are out of the scope of BTT version 6.1.1. These services in BTT version 5.2 application system are not included in the migration tool, so you need to migrate them manually.
- The server side event mechanism of BTT version 6.1.1 is changed. You can define event related migration rules in BTT version 6.1.1, but for some complicated cases that migration rules cannot handle, you must migrate them manually.

- The communication service access uses JCA connector in BTT version 6.1.1. You must modify it manually.

Diagnosis for the migration tool

After each migration task, the migration tool will save the migration messages into a report file. The file is stored in the project workspace under the category named **report**. The recommendations for manual migration are saved in the recommendation file under the same category. In this file, the parts that need manual migration are listed out, and recommendations for manual migration are provided. You can perform the necessary manual migration according to the recommendations.

Notices

IBM may not offer the products, services, or features discussed in this document in all 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 document does not give 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 (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, 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 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.

Trademarks