COGNOS®

Cognos

Impromptu(R)

Deploying Impromptu Applications

Product Information

This document applies to Impromptu (R) Version 7.1 and may also apply to subsequent releases. To check for newer versions of this document, visit the Cognos support Web site (http://support.cognos.com).

Copyright

Table of Contents

Welcome

What Is In This Document

This book provides guidelines to follow when you develop and deploy Impromptu reporting applications to a broad audience of report end users on a local or wide area network. When you complete it, you will understand how deployment is affected by application design decisions in the following broad areas:

- audience and report analysis
- database considerations
- environmental considerations
- support and maintenance considerations

The design considerations you will read about in this book are not absolutes. Every project has its own characteristics and requirements. Use the concepts in this book to help you determine requirements for your projects, before you begin significant development effort.

What You Need to Know to Use This Document Effectively

This book is for the administrator who sets up the Impromptu environment for users. Before reading this book, you should understand how to create and maintain Impromptu catalogs and reports. Read

- the *Administrator's Guide*
- the Impromptu Administrator online Help
- *Mastering Impromptu Reports*

Other Information

Our documentation includes user guides, tutorial guides, reference books, and other pieces to meet the needs of our varied audience.

All information is available in online help. Online help is available from the Help menu and Help button in Windows products.

The information in each online help system is available in online book format (PDF). However, the information from a given help system may be divided into more than one online book. Use online books when you want a printed version of a document or when you want to search the whole document. You can print selected pages, a section, or the whole book. Cognos grants you a non-exclusive, non-transferable license to use, copy, and reproduce the copyright materials, in printed or electronic format, solely for the purpose of providing internal training on, operating, and maintaining the Cognos software.

In Windows products, online books are available from the Windows Start menu (Cognos) and from the product Help menu (Books for Printing). All online books are available on the Cognos documentation CD. You can also read the product readme files and the installation guides directly from the Cognos product CDs.

Only the installation guides are available as printed documents.

An annotated list of other documentation, the *Documentation Roadmap*, is available from the Windows Start menu or the Impromptu Help menu.

Questions or Comments?

For the fastest response to questions about using Impromptu, contact customer support.

For information about customer support locations and programs, see Cognos on the Web on the Help menu or visit the Cognos Support Web site (http://support.cognos.com).

Chapter 1: Planning for Deployment

This chapter introduces Impromptu planning consideration for the following broad design areas:

- audience and report design considerations
- database considerations
- environmental considerations
- support and maintenance considerations

Requirements and constraints within each of these areas influence decisions regarding Impromptu catalogs, reports, templates, and overall project infrastructure. By answering key questions relating to these broad areas in advance, you make your reporting system easier to maintain, more portable, and of more use to its intended audience.

Planning for Enterprise Deployment

The successful deployment of an Impromptu reporting application depends on careful planning before you begin report creation and distribution. The decisions you make during the planning stage influence how easily you can deploy and maintain the system.

When planning an Impromptu reporting application, either as a standalone system or as part of a larger system, consider deployment issues in the following broad areas:

Audience and Reports

A large part of any reporting project is defining the audience and their reporting needs. Many of the decisions you make at this stage affect the deployment of the catalogs you develop and your ability to maintain systems of reports. See "Audience and Report Considerations" (p. 11).

Database

Understanding clearly where the source data comes from and how Impromptu accesses it influences your catalog design, and the access privileges that you grant to end users. For systems that you intend to port from one database environment to another, certain Impromptu features can make it easy to move between environments. See "Database Considerations" (p. 17).

Environment

The reporting system that you create is ultimately deployed to users' desktops, via a LAN or some distribution mechanism. Understanding all deployment and environmental issues allows you to plan for controlled, ongoing distribution. See "Environmental Considerations" (p. 31).

Support and Maintenance

Business Intelligence systems evolve constantly as the information flowing through a business or group changes. Once you've deployed reports to your user community, it's essential that you provide a way for them to give useful feedback and report any problems or enhancement requests. It's also important to constantly improve and add to the system as new requirements arise. See "Support and Maintenance Considerations" (p. 43).

Gathering Information from Key Resources

Seek input from the following people as you plan your system:

Impromptu Users

User input is essential in ensuring that you deliver something they'll find useful. Interview at least one or two Impromptu users from each user group or department to determine their reporting requirements.

Impromptu Report Designers

Advanced Impromptu users who create reports and templates for other users. These are advanced power users who understand reporting requirements and Impromptu's capabilities.

Impromptu Administrator

This is you: the person responsible for overseeing the creation and deployment of Impromptu catalogs, reports, templates, and related files for the reporting system. You should be an Impromptu expert, and someone who understands the business processes for which you're attempting to create a reporting solution.

Database Administrator (DBA)

The person or people responsible for administration of your corporate database systems. With their in-depth understanding of RDBMS's, they can provide information about data sources, data retrieval techniques, and query optimization tips.

Application Developers

The Impromptu experts who design and implement your Impromptu reporting system. They can help you maximize the system's effectiveness by using Impromptu's advanced features.

LAN Administrator

The person or people responsible for the physical infrastructure (LAN, WAN). They can provide you with LAN resources as required to support your application.

Web Administrator

The person or people in charge of a corporate web site or intranet. They can set up web pages and server aliases from which users can run HTML reports that you've provided with Impromptu.

Senior Management

To ensure the success of your Impromptu application, you'll need support from the senior management who are funding the project.

Impromptu Deployment Checklist

As you prepare to design your application, consider the following questions:

- Do you have a plan in place for deploying all the required files for the reporting system? Impromptu systems can include catalogs, reports, initialization (INI) files, macros, database access drivers, hotfiles, snapshots, and HTML files. Additionally, your completed system may include other programs and files (for example, Visual Basic programs, bitmaps, and text documents) not strictly required by Impromptu. These are discussed in detail in "Impromptu Applications" (p. 31).
- Is there a consistent PC configuration among your Impromptu users (applications, LAN directory mappings)? If all users map their drive letters consistently to the same LAN shares, Impromptu will more easily locate catalogs, reports, and other required files at runtime. Where Impromptu supports it, use Universal Naming Conventions (UNCs) to specify LAN locations.
- Does each user have the appropriate connectivity software (ODBC driver and DNS entry, SQL*Net, Sybase Open Client, and so on...) to access the target RDBMS?

- For distributed catalogs, will your network support the increase in network traffic incurred when users copy master catalogs to their machines? If your catalogs are large (1 MB or larger), this can become a performance issue.
- Are users located at many diverse geographical locations? If so, you must ensure that they all have access to the completed system. You can place the application on a central shared LAN, or copy it to different locations as required. For more information, see "Where Should You Store Application Files?" (p. 31).
- For catalogs that use hotfiles, snapshots, and remote snapshots, are these available to all users? Will hotfiles be deployed with the catalogs or reside on a shared LAN drive?
- Do the catalogs and reports you have created use macros? If so, will you deploy the macros with the reports and catalogs they support, or will you locate them on a central shared LAN drive?
- What class of machine is the system targeted at? Be sure that you're aware of configuration issues. A report that runs well on the 64 MB, 233 Megahertz Pentium Pro PC that is standard for the application development team might not run well on a 16 MB, 90 Megahertz laptop that is standard for the sales team.
- Do all users run at the same video resolution? With the same color palette support? With the same system fonts installed? All of these can affect the look of a report when it appears on a machine for which it was not designed.
- Will all users always have access to the LAN, or will mobile users need to access reports when they're off-line? For off-line access, you can create Distributed catalogs that users can download to their own systems. Users can then take the catalog with them, for example to a client's site or to a conference.
- Do users have or need specialized software? In an enterprise reporting environment, it's likely that users will need software in addition to Impromptu.
- Will you need to automate the report refresh/distribution process? Plan to implement some kind of automated report refresh and distribution system.
- Does the reporting system you're creating need to integrate with some larger system (Lotus Notes, for example)? Consider the integration implications. Impromptu can act as both an OLE client and server, and it also provides a complete OLE Automation API for automating both report and catalog creation.
- How will you administer the installation of Impromptu across your organization?

Chapter 2: Audience and Report Considerations

Understanding who you are implementing a system for and what reports the system will provide are the most critical pieces of information required to successfully deploy your Impromptu reporting system.

Understanding User Requirements

From a deployment planning viewpoint, this is the starting point. By knowing who the system is aimed at and what information it will provide, you can work towards a solution that meets your users' needs. You will also identify who and what the system is not intended for.

User Reporting Activity Profiles

Understanding your audience and their needs requires that you consider the following questions:

- Will users only view reports, or do some ad hoc reporting on their own? Is there a mix of such users?
- Is there any requirement for users to make modifications to the catalogs that you provide? For example, might some users want to administer user classes for their own groups, or add their own custom folders?
- Will all users always be connected to the LAN on which you'll store the finished system, or will some require disconnected access? Will all users always be connected to the target database?
- Do you have a clear understanding of the reports that end users require? Can you express these requirements clearly?

Early in the planning stages of your project, schedule user interviews during which you can derive answers to these important questions. Plan to interview at least one or two representative users from each affected user group. See "Catalog Deployment Considerations" (p. 13)

Security and Reporting Privileges

The makeup of your user community impacts the way in which you'll grant reporting privileges to users. As you gather information about their reporting activities, you must consider who requires access to what set or subset of data. To do so, ask the following questions:

- Do all users require access to the same source data, or subsets of the same source data? If so, implement user classes within a common catalog; if not, consider two or more catalogs.
- For a large reporting application, it's likely that different groups of users will have different reporting needs. How many distinct classes of users can you identify? By defining user classes, you can facilitate information deployment to users, as you set up security by folder, by table, and by value within the catalog.
- Are there power users within your organization that you'd consider allowing to create their own personal catalogs? If so, they'll require the Administrator edition of Impromptu, and appropriate training.
- What is the general level of database literacy within your user community? Will you have to implement query privilege restrictions to avoid runaway queries?

The answers to these questions help you to map out effective catalogs and user classes that deliver the right data and reports to the right audience. See "Setting Up Catalog Security" (p. 15)

While user requirements are the main drivers of your security architecture, you must also consider environmental constraints. A data warehouse environment, for example, generally supports ad hoc queries better than legacy systems. As you analyse your security requirements, consider the potential impact of queries on the target database.

Taking Inventory of Reports

As you begin your planning, remember that Business Intelligence (BI) projects are always iterative in nature. If you try to build an entire corporate BI reporting system as a single project, it will likely last for years and fail. BI projects must deliver business benefits to their intended users quickly. Once you've achieved this, users will expand the scope of their requirements as they begin to understand the power of having critical BI information at their fingertips.

Instead of trying to exhaustively inventory all of the system's reports, document the domain of data that users think they need, and provide some representative standard reports. Where possible, use current paper-based reporting systems and on-line systems in conjunction with your user interviews to model your set of representative reports. As you build reports and solicit feedback from users, you can alter them as required. You can also use Impromptu's advanced features to add value to existing reports. This iterative approach will continue through the life of the BI application.

Key questions to consider when researching reporting requirements are:

- How large are the reports your users will execute? Will they generate large queries that may require substantial database resources to execute?
- Are there business rules that you can define and integrate into your Impromptu catalogs? For example, a business rule might be: a "tier-1 customer" is any client that orders more than $250,000.00 worth of product in a single year.
- For information about how to set up catalog conditions, filters, and prompts, see the *Impromptu Administrator's Guide*.
- Do users need summary-level reports, detail reports, or a combination of both?
- Impromptu can deliver summary reports, detail reports, crosstabs, nested crosstabs, and reports that provide both summary and detail information. However, certain kinds of reports execute more quickly than others.
- For information about the impact of summaries and calculations on queries, see the *Impromptu Administrator's Guide*.
- Do users simply view reports, or do they perform ad hoc reporting? See "Catalog Deployment Considerations" (p. 13)
- Will users ever associate reports with catalogs other than the one with which the reports were originally created? If so, you must be aware of how Impromptu binds report columns to the objects in the catalog. See "Associating Reports with Different Catalogs" (p. 34) See also "Where Should You Store Application Files?" (p. 31).
- Are there catalog and report interdependencies you must consider when deploying reports?
- As part of your reporting requirements, consider report interdependencies, as when one report drills through to another, or when a report is used to provide picklist prompt values for another. These dependencies are important when you begin deploying reports to users. See "How Impromptu Searches for Dependent Files" (p. 33)
- How often will you have to update the reports you're creating, and how will you control these updates? See "Support and Maintenance Considerations" (p. 43).
- Have you considered corporate reporting standards and a common look and feel across all of the reports that you provide? See "Templates" (p. 37).
- Have you considered potential environmental issues, such as LAN access, application integration, and the packaging of required supporting files? See "Environmental Considerations" (p. 31).
- Is there a requirement to publish Impromptu reports in HTML format so that users can access them via a web browser? See "Storing and Distributing HTML Reports" (p. 40).

Focusing your Application

Once you've identified the audience and their reporting requirements, restrict the focus of the application accordingly. If you allow the reporting requirements to expand without control at the outset, the project will likely stall. For example, you might focus on a sales reporting application or marketing analysis reports. For a sales reporting application, your target audience might be sales managers, sales analysts, sales team leaders, and sales representatives. For a marketing analysis application, your target audience would be the marketing personnel involved in strategic and tactical marketing initiatives.

Catalog Deployment Considerations

The type of report consumer influences the type of catalog(s) that you should create. A successful catalog is one that presents information to users in a meaningful fashion, and encapsulates "business rules" as calculations and filter criteria that users can apply as required.

The *Impromptu Administrator's Guide* provides information about the kinds of catalogs you can implement and what they're intended for. In addition, the *Mastering Impromptu Reports* online book includes information about modifying catalogs. This section provides a synopsis of catalog design considerations from a deployment perspective.

When deploying a reporting system to a wide group of users, consider using one of the following catalog types:

- Shared
- Distributed
- Secured

Personal catalogs (the fourth type) are used by one user. You can access them only with the Administrator version of Impromptu. Personal catalogs are not useful for deployment in a large group or organization.

Shared Catalogs

Consider whether any users will need to modify the catalog(s) that you create. If not, then you should use shared catalogs, which are ideal for enterprises or departments with multiple users who access the catalog on a central LAN location. This allows users to perform ad hoc or view-only reporting, but leaves the administrator in complete control of the catalog content.

Distributed Catalogs

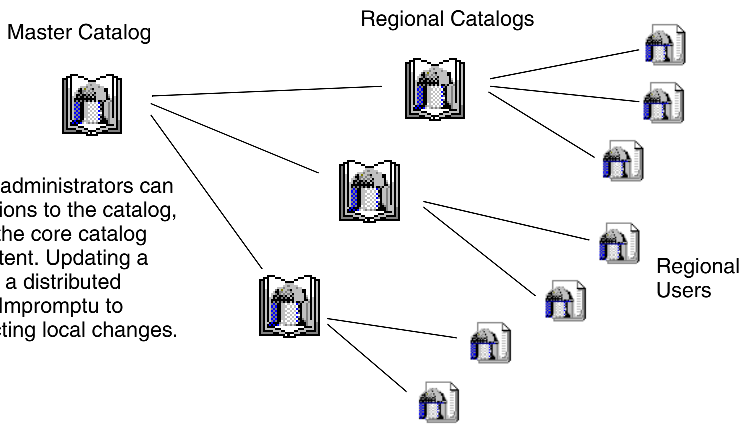
Distributed catalogs are ideal when some users must modify the catalogs that you create, or when users require access while they're disconnected from the corporate LAN. Whenever a user opens the master copy of a distributed catalog, they're prompted for a location in which to create their own local copy of this catalog. The local copies remain synchronized with the master distributed catalog, which the Impromptu Administrator maintains. Once a user has created a local copy (or you have created one for that user), they can enhance the catalog with new user classes and folders.

When deploying distributed catalogs, you must provide clear guidance to users about the locations in which they store their local catalogs. In most situations, unless you can be sure that users create local catalogs in a consistent local location, you should create local catalogs for your users. That way, you can be confident the catalogs will be stored in the same physical location on each personal computer throughout the organization.

Deploying the Deployment of Catalogs

In some environments, you may want to create a centrally-administered catalog over which the Impromptu Administrator retains control, but which regional administrators can customize for their users.

You do this by creating a central distributed catalog that you make available to specific users, each of whom acts as regional or departmental administrator.



Note: Regional administrators can make only additions to the catalog, not changes to the core catalog structure or content. Updating a regional copy of a distributed catalog causes Impromptu to overwrite conflicting local changes.

This approach deploys the deployment of Impromptu catalogs and reports as follows:

1. A central IS team creates the master distributed catalog that controls access to data for several territorial or branch offices.
2. The central IS team publishes distributed copies of the master catalog for regional administrators. Impromptu creates local (regional) copies of the catalog for those administrators.
3. The regional administrators make changes to their local catalog copies as required, for example, adding user classes for their branch office, or creating new folders for their users.
4. The regional administrators make the modified catalog available to the users in their branch offices. The local copy of the distributed catalog becomes a shared catalog for the members of the branch who access it.
5. Periodically, the central IS team updates the regional catalogs with new information from the master distributed catalog. Any new tables, columns, or other catalog entities are added to the regional catalogs in this way.

Secured Catalogs

One of the defining aspects of reporting is whether users create their own reports, or if they only view and print standard reports that you provide. If you're deploying to users with view-only requirements, use a secured catalog. This allows users to view, print, and save in various file formats the reports that you provide. The secured catalog feature turns Impromptu into a report viewer rather than a report design and creation tool.

Deploying a secured catalog greatly extends your control over the queries that Impromptu sends to the target RDBMS. By denying users the ability to create ad hoc reports, you eliminate the possibility that they inadvertently generate costly queries (other than ones you've provided) and negatively impact the target RDBMS. For example, an inexperienced user might generate SQL statements that attempt to summarize millions of records, or generate huge cross-product reports. You must weigh this advantage against the requirements of individual users who require some level of ad hoc reporting capability.

Using LAN Security to Lock Reports

An alternative to using secured catalogs is using your LAN security features to store catalogs and reports in read-only folders. This method prevents users from changing reports that you've published, but allows them to save their own copies in another location.

Custom Catalogs

Impromptu catalog types are not entirely mutually exclusive. You cannot create a catalog that is both distributed and shared, but you can create a single catalog that is both shared and secured.

The initial creation of either a Shared or Secured catalog sets default privileges for its user classes. For a secured catalog, new user classes are by default not granted the ability to create or edit reports. For a shared catalog, new user classes are granted report design and creation privileges. As the administrator, you can change the privileges for a specific Impromptu user class using Impromptu's Governor settings.

For example, a shared catalog may contain eight user classes, six of which are granted report modification privileges, and two of which are not. From an end user's point of view, the catalog is secured for the end users without report modification privileges, while it is a shared catalog for members of the other user classes. Similarly, if you create a secured catalog in which no user classes can create or modify reports, you can selectively grant such privileges to some subset of user classes. The catalog remains secured for certain users, but is shared for users you've granted report design privileges.

Setting Up Catalog Security

A major step in deploying Impromptu is to determine what users or groups of users require access to specific pieces of data.

As a rule of thumb, strive for a single catalog per application. A single Impromptu catalog can provide many different groups of users access to a single data source, even if the information requirements vary from group to group. For example, users from Sales, Human Resources, Finance, and Marketing may all require access to a single database. The information they require may overlap to a certain extent, while the tables, columns, or the required presentation differ.

For a large enterprise, target the user communities you've identified in the planning stage, and then create user classes for those groups. Where there are different reporting needs within groups, implement hierarchical sub-classes within the user classes that you've defined. A typical hierarchy might involve:

- Executives and senior decision makers, who receive broad access to corporate data, but require it summarized and packaged concisely. This often means summary financial reports across an entire enterprise, summarized sales forecasts, summary productivity reports, and so on. These people generally do not require ad hoc report creation privileges; they view summary reports regularly.
- Senior resources (regional or territorial managers, directors, analysts), who receive broad access to the data for their domains, but are restricted from accessing data that does not pertain to them. This often means access to a subset of the data that is available in the catalog, and it includes summary reports for the region, group or territory. It also includes any underlying detailed reports.
- Intermediate resources (team leaders, branch managers) who receive broad access to the data for their teams or branches.
- Field or floor-level resources (sales representatives, field employees, plant workers) who receive access to the data they require to perform their day-to-day jobs.

In many cases, a company's organisational chart is a useful as a starting point in defining user classes.

The *Impromptu Administrator's Guide* provides an example of such a user class hierarchy. It also describes how to set up:

- security by folder
- security by table or column
- security by value
- report governors

Chapter 3: Database Considerations

Impromptu reporting applications can derive data from

- existing data warehouses or data marts
- legacy systems, where data from one or more heterogeneous RDBMS environments must be integrated
- staging data warehouses that you create as part of your reporting and analysis system

Where the data comes from influences how you should create catalogs for easy deployment. If you're creating a system that you intend to port from one RDBMS environment to another (for example, from Oracle to Sybase SQL Server), you can use advanced Impromptu features to increase the catalog's portability.

For an up-to-date list of environments supported by Cognos products, such as operating systems, browsers, Web servers, directory servers, database servers, OLAP servers and more, please refer to:

- http://support.cognos.com/support/products/software_environments.html

Database Considerations

When implementing any reporting and analysis system, understanding where source data comes from is a key step in providing data access that is

- responsive
- of low impact to currently-running production systems
- portable to other locations and environments as required
- insulated from changes in the underlying database structure

As you design your catalogs and reports, consider the following questions:

- Does the system derive information from a data warehouse/data mart environment, or from a legacy system? The answer affects the complexity of the Impromptu catalogs that you create, and the way in which you allow end users to query the target RDBMS.
- Will you migrate the catalog to support multiple databases? For example, you might create a catalog against an Oracle instance and then port that catalog to another identically-structured Oracle instance. See "Designing Catalogs that Travel Well" (p. 19)
- Will you require a catalog to work with multiple database types? For example, if you're developing an application for wide use, you might require catalogs that work against both Oracle and MS SQL Server versions of a particular database. See "Designing Catalogs that Travel Well" (p. 19)
- Will you be localizing your catalog for a national language? See "National Language Portability Issues" (p. 29)
- Do all affected users have the required database accounts? Or will you set up proxy (generic departmental) accounts that can be used by all of the users in a specific group?

The DBA: A Valuable Resource

The Database Administrator (DBA) is a key resource in setting up data sources for Impromptu. The DBA can provide valuable insights into how the underlying data is stored, and provide guidance about how you should derive data for your data warehouse. The DBA's participation is required to determine an ongoing schedule for data extraction from the OLTP system.

Impromptu Data Sources

Impromptu supports a wide range of data sources, many of which are integral parts of production systems and data warehouse solutions for large corporations. In setting up the Impromptu catalogs for your reporting application, you must decide which of the company's databases are good candidates to provide that data you need to create reports, or whether you need to create a staging data warehouse for Impromptu to read.

Comparing OLTP Systems with BI Systems

Typically, a production Online Transaction Processing (OLTP) system is not a good data source for large-scale BI reporting and analysis systems. OLTP systems are designed and optimized for different purposes than BI applications. The following table lists important differences often cited by BI experts:

OLTP Systems	BI Systems
Application orientated. Tables and views are optimized to make the application run faster.	**Subject orientated**. Tables are modeled on business concepts and designed for usability.
Non integrated. Data for different business applications (like finance versus marketing) is often stored across multiple systems.	**Integrated**. All data relating to a specific subject (like Customers) is stored together.
Volatile. Data is updated each time a transaction occurs. Records are edited in place in the database.	**Non-volatile**. Records are rarely updated or deleted. They are almost always only added.
Little summary data. Data is normalized to optimize for performance. There is no storage of rolled-up values.	**Multiple granularity with summaries**. Data is summarized at various levels of granularity to provide appropriate response times for large volumes of transaction data.
Non-time variant. Holds data that represents the current state of the enterprise.	**Time variant**. Holds data for several time periods so that useful growth comparisons can be made.

OLTP Systems

OLTP systems store transactions for a company's day-to-day business. They are optimized for

- data concurrency (supporting multiple users)
- data integrity
- read/write performance

The RDBMS systems that service these requirements are often complex in terms of their underlying tables, views, indexes, and joins. As a result, extracting highly summarized data from such a system is often impractical and costly in terms of RDBMS and system resources. Also, most large organizations use several OLTP systems. For example, an organization may use an Oracle solution for Finance applications, while it uses a DB2 database with supporting programs for Purchasing applications. This can make it extremely difficult to derive summary data from detailed operational data.

However, for most organizations, OLTP systems are where you'll find the bulk of raw data for important business entities, like customers, orders, products, shipments, and inventory levels—exactly the kind of data that, when viewed in aggregate or over time, provides useful business intelligence. For this reason, the first step towards implementing a BI reporting and analysis system is often the design and implementation of a staging data warehouse.

Business Intelligence Systems

The data that's most valuable to a BI system is not optimized for concurrency or update performance. Instead, BI data sources are intended to store information about the domains of data that business users require. Moreover, the kinds of queries required to feed a BI system generally involve hundreds of thousands or millions (in some cases even billions) of records. Such queries are too costly in terms of their impact on the operational database to be permitted on a wide scale.

Creating a Staging Data Warehouse

Because of the conflicting requirements between OLTP and Business Intelligence systems, Impromptu may require a staging database (separate from the operational database). If a data warehouse is already in place, then setting up a useful Impromptu catalog can be relatively easy, depending on whether the data in the warehouse matches your catalog requirements. In many cases, you may have to add new information (columns or entire tables) to the existing data warehouse.

If no data warehouse exists, you must create a staging data warehouse from which Impromptu can read subject-oriented data. The creation of a data warehouse is beyond the scope of this book. However, you should consider at least the following general concepts before you begin:

- Define the purpose of the data warehouse. In this case, it should be to provide access to the subject-oriented data from which you can build Impromptu catalogs and the required reports.
- Determine what source data the data warehouse requires. Can you extract the data from a single RDBMS? Or will you have to clean, summarize, and merge data from several different systems? If so, then creating the data warehouse can be far more difficult.
- Define how you will set up feeder programs that extract data from your OLTP systems, clean it if necessary, and place it in the data warehouse. This will require you to create scripts, programs, standard queries, and devise a way to automate it all.

Designing Catalogs that Travel Well

Many Impromptu applications are created with the intent that they will be ported to other database environments. This chapter describes Impromptu features that you can use to port Impromptu catalogs from one RDBMS environment to another.

From a user's viewpoint, all databases appear equal in terms of reports that they want to construct. In reality, many factors determine the levels of functionality and performance a reporting environment can achieve.

Impromptu does not constrain itself in a lowest-common-denominator approach in terms of support for RDBMS features. Rather, it exploits vendor-specific functionality across the entire set of supported databases. For example, if a database environment does not support a certain type of outer join, Impromptu extends that RDBMS's capabilities by performing the required join locally.

To construct a generic reporting solution, you must understand the feature set of the target database environments, and then either:

- derive a lowest-common-denominator specification, in which the overall application is limited by the least feature-rich RDBMS environment, or
- create a generic solution that minimizes the modifications required to port the application

If you opt for a generic solution, you must account for feature differences both within a database vendor's product family and in the data access method. For example, you must consider the differences between the Oracle RDBMS environment and the Sybase SQL Server environment if you want to implement a reporting system in one environment and then port it to the other. Similarly, if you port from a native data access environment to an ODBC (or other gateway) data access environment (or vice versa), you must consider mappings of schemas and table qualifications to RDBMS objects, and account for any differences as you port the application.

Understanding Schema and Catalog Qualification Levels

An Impromptu catalog consists of collections of objects which, at the lowest level, represent database tables and their columns. Above these collections can be one or more intermediate collections that represent the name spaces supported by a given RDBMS environment. For example, the Oracle environment includes the schema level between the database root and the table level. The root owner of these collections is known as the database object.

Each Impromptu catalog contains a database object which can be structured in four ways:

- Database object containing Table objects (no qualification)
- Database object containing SchemaLevel objects that contain Table objects (one level of qualification)
- Database object containing CatalogLevel objects that contain Table objects (one level of qualification)
- Database object containing CatalogLevel objects that contain SchemaLevel objects that contain Table object (two levels of qualification)

For a Database object with no qualification levels, Impromptu can access the Tables in the collection by name because all Table object names are unique. For Database objects with qualification levels, the Table object names are not necessarily unique, as the same table name may appear in multiple schemas. In such cases, an error can occur if you try to access a Table object by name from the Database object. The same is true for aliases that you create for the Table object and SchemaLevel objects where the Database object contains two levels of qualification.

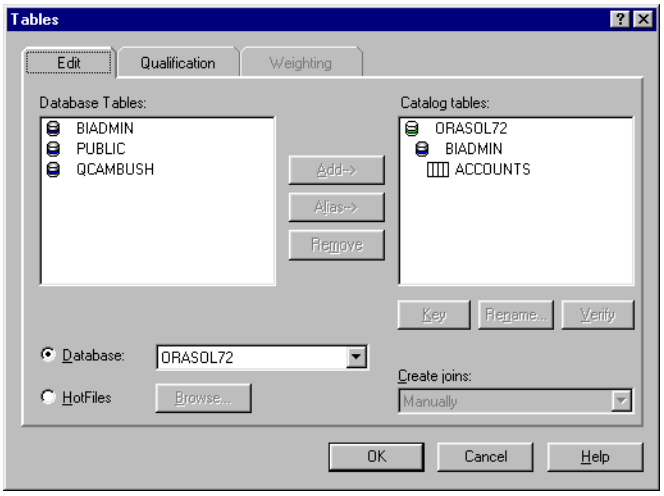
The following illustration shows level differences between a Sybase SQL Server database and an ORACLE database when mapped into an Impromptu catalog:

The Oracle catalog includes:

- a database level (ORASOL72)

- a schema level (BIADMIN)

- a table level (ACCOUNTS)



Database Tables:
BIADMIN
PUBLIC
QCAMBUSH

Catalog tables:
ORASOL72
BIADMIN
ACCOUNTS

(●) Database: ORASOL72
( ) HotFiles

Create joins: Manually

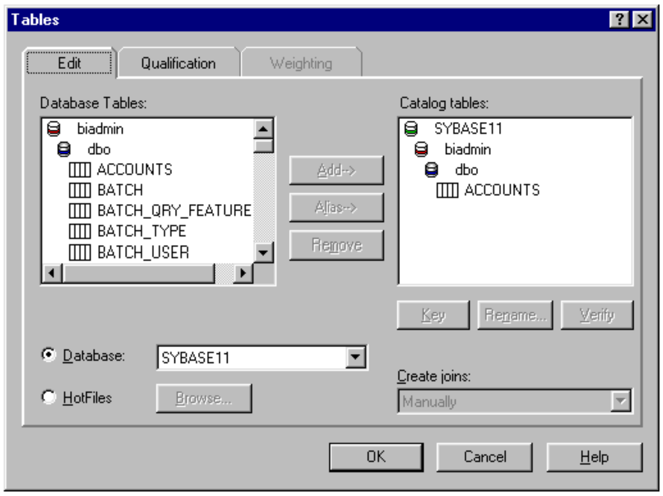
The Sybase catalog includes:

- a "database" level (SYBASE11)

- a catalog level (biadmin)

- a schema level (dbo)

- a Table level (ACCOUNTS)



Database Tables:
biadmin
dbo
ACCOUNTS
BATCH
BATCH_QRY_FEATURE
BATCH_TYPE
BATCH_USER

Catalog tables:
SYBASE11
biadmin
dbo
ACCOUNTS

(●) Database: SYBASE11
( ) HotFiles

Create joins: Manually

The level of qualification of object names varies among RDBMS environments. Impromptu allows you to refine the level of qualification, so that higher levels of qualification are either included or excluded during SQL generation for a query. However, Impromptu does not allow you to change the qualification levels by adding or removing levels, except via OLE automation calls.

In the Tables dialog box, you can

- include or exclude a qualification level
- rename a qualification level

These capabilities are important when you use a catalog developed in one environment against an identically structured set of tables in another environment. You can change the qualification levels and their names to match the new environment. Using the sample database definitions shown on (p. 21), you can port the Sybase catalog to Oracle by excluding one level of qualification, and changing the qualification level names as required.

Note: If you rename ORASOL72, for example to ORASOL72ABCD, Impromptu automatically adds an entry to the Cognos.ini file. The new entry reflects the new logical name and the previous connection information. If you change the database name reference by picking a name from the Database drop-down list, or by manually renaming the entry to a name already mapped in Cognos.ini, Impromptu prompts you with a "connect to database" sequence.

Adding Qualification Levels

Using the example shown on (p. 21), if you were to rename the database from the ORACLE (ORASOL72) database to the Sybase (SYBSOL11) database, the catalog would not function in a Sybase environment. This is because the Sybase environment has an extra level of qualification.

In the Tables dialog box, you cannot add the required qualification level (dbo) to ensure that the catalog functions properly in the Sybase environment. However, you can use CognosScript to write a macro that adds qualification levels to an existing catalog schema levels structure. You can use the InsertQualificationLevel and RemoveQualficationLevel methods to add or remove a specific type of qualification level (either a catalog level or a schema level) for a Database object. In the example shown on (p. 21), you could rename the Database object from ORASOL72 to SYBSOL11, and then use

- the Name property of the SchemaLevel object to rename the existing Oracle schema level "BIADMIN" to "dbo"
- the InsertQualificationLevel method to add a catalog level qualification named "biadmin" to the database

For information about the AddQualificationLevel method, open the CognosScript Macro Help and type the following in the Index search box:

- InsertQualificationLevel

Removing Qualification Levels

When you port from a more qualified environment to a less qualified one (for example, from a Sybase database to an Oracle database), use the Tables dialog box to hide the extra level of qualification and rename the ones that remain as required. You must hide qualification levels if the catalog tables are qualified more than is supported by the database you are querying. For example, a fully qualified Sybase table can be specified as A.B.C, while in ORACLE this could only be expressed as B.C. If you do not hide qualification level A, ORACLE cannot prepare the query.

You cannot change the type of qualification level (schema or catalog) in the Tables dialog box. However, if the object names match and use the same case, Impromptu will continue to run reports in spite of the mismatched types. Mismatched level types prevent the Verify command from finding tables, as it uses the qualification levels along with the displayed names to compare catalog metadata with the RDBMS metadata.

Case Sensitivity

Impromptu preserves the case sensitivity of the names of objects as defined by the RDBMS server. The sample Data Definition Language (DDL) shown in this chapter includes examples of name space and object names in upper, mixed and lowercase. Impromptu preserves the case of the name components during generation of SQL and database requests. As a rule, use upper-case names, and limit character segments to 18 characters. This provides the highest degree of catalog portability.

Database	Example
ORACLE	"BIADMIN"."ACCOUNTS""ACCOUNT_ID"
Sybase SQL Server	"biadmin"."dbo"."ACCOUNTS""Account_Id"
MS/SQL Server	"biadmin"."dbo"."ACCOUNTS""Account_Id"
Informix Online	"bisrv"."accounts""account_id"

You can rename an RDBMS object in Impromptu with the Rename button on the Table or Column objects in the Tables dialog box.

Datatypes

Each table within a database contains one or more columns whose database type maps directly, or is re-mapped to, an Impromptu type.

Impromptu uses the datatype information to present appropriate choices within the Expression Editor, and at runtime when it renders values for display within a report. The supported datatypes and sizes vary across RDBMS vendors, and Impromptu does not always support all possible types exposed by each RDBMS vendor.

For more information about supported datatypes and how they are mapped, visit the Cognos Support Web site (http://support.cognos.com).

Function Portability Considerations

As a catalog or report designer, you can create expressions that reference functions in

- catalog folders
- filters
- join definitions
- reports and templates

Impromptu exposes a large set of functions, many of which are database vendor specific, and many of which are performed by Impromptu. Within Impromptu, these functions are differentiated by the following icons:

- a "local" function that Impromptu executes on the PC.
- a "common" function that Impromptu executes by "pushing" it into the SQL it sends to the RDBMS, where it is executed by an equivalent RDBMS function.
- a "database only" function that can only be performed by the target RDBMS.

Use Local Functions

Only local and common functions are considered portable. Where possible, use functions that Impromptu will push into the SQL it sends to an RDBMS. This can impact how Impromptu decomposes the report specification into SQL statements. In small volume environments this may not be significant. However, for reports against large-scale data stores with many millions of rows, locally executed functions can impact performance negatively.

For example, the first of the following expressions will decompose into a CASE expression when pushed to MS/SQL Server, while the second one will not:

```
if(Account Id = 1010) then ('Yes') else ('No')

if(Account Id = char_length('a') + 1010) then ('Yes') else ('No')
```

The second expression uses a local Impromptu function (char_length) to determine the length of the character expression. If you change the expression to use the octet_length function (an equivalent RDBMS function) instead of char_length, you alter the statement back to using a case expression:

```
select T1."Account_Id" c1, case  when T1."Account_Id"=datalength('a')+1010 then 'Yes'
else 'No' end  c2 from "biadmin"."dbo"."ACCOUNTS" T1
```

An extreme example of this is a simple reflexive join specification that uses a local function as follows:

```
ACCOUNTS."Account_Id" + char_length ('a') = A2 (ACCOUNTS)."Account_Id"
```

The resultant SQL is a cartesian join:

```
select T2."Account_Id", T1."Account_Id"
 from "biadmin"."dbo"."ACCOUNTS" T1, "biadmin"."dbo"."ACCOUNTS" T2
```

Don’t Nest Local Functions in RDBMS Functions

You cannot build an expression in which you nest a local function within a function that is normally pushed to the RDBMS. However, you can nest database functions within Impromptu local functions.

Joins

Impromptu supports several versions of table join specifications in a catalog. At execution time, Impromptu determines how to decompose a report specification into one or more SQL statements. In general, the more an RDBMS conforms to the SQL-92 specifications for joins, the more likely it is that Impromptu will push the query through to the RDBMS.

Where Impromptu knows that an RDBMS cannot perform specific join processing (such as certain types of outer joins, with or without restrictions), it supplements the RDBMS with its own internal relational engine processing. The report will continue to return the expected data, but may require more client-side processing than in other instances.

The need for outer joins as an example is driven by the underlying data model and client reporting requirements. If the data model allows for optional relationships, but users have no need to preserve data where a join would fail, there is no need to define an outer join.

Transactions

Database transactions are associated with a user profile. Within a reporting environment, it makes little sense to apply high levels of isolation that cause readers to block writers. In the data warehouse or data mart environment, this is likely not a consideration, as the environment is designed for read access.

Not all RDBMS engines support the same levels of isolation. Impromptu's data access layer automatically escalates a requested isolation to the next available higher level as required.

For complete details about transactions and database isolation levels supported for each RDBMS, see *Appendix A: Supported Databases* in the *Impromptu Administrator's Guide*.

Sample RDBMS to Impromptu Mappings

This section provides some Data Definition Language (DDL) excerpts for Impromptu supported databases, and lists the Impromptu mappings that result from the DDL. You can use this information to understand the schema levels you must consider as you port a catalog from one RDBMS environment to another.

Oracle Tables and Impromptu Mappings

The following table shows Oracle DDL statements that create a table named ACCOUNTS using the login BIADMIN, together with the equivalent mapping in Impromptu.

Oracle DDL Statements	Impromptu Mapping	
`CREATE TABLE ACCOUNTS`		
`(Account_IdNUMERIC(6),`	`ACCOUNT_ID`	`Integer`
`Name    CHAR(50),`	`NAME`	`Char`
`Lock_AttrCHAR(1),`	`LOCK_ATTR`	`Char`
`Address CHAR(40),`	`ADDRESS`	`Char`
`City    CHAR(15),`	`CITY`	`Char`
`ProvinceCHAR(15),`	`PROVINCE`	`Char`
`Zip     CHAR(6),`	`ZIP`	`Char`
`Area_PhoneNUMERIC(3),`	`AREA_PHONE`	`Small Int`
`Phone   CHAR(9),`	`PHONE`	`Char`
`Area_FaxNUMERIC(3),`	`AREA_FAX`	`Small Int`
`Fax     CHAR(9),`	`FAX`	`Char`
`Credit_LmtNUMERIC(9,2),`	`CREDIT_LMT`	`Integer`
`CurrencyNUMERIC(2),`	`CURRENCY`	`Small Int`
`Group_AcctCHAR(6));`	`GROUP_ACCT`	`Char`

The Catalog overview for the above table looks like this in Impromptu:

```
Database Information

   Logical Name: ORASOL72
   Physical Name:ORACLE@%s@bison02/%s
   Type: OR

Database Structure

   MetaSchema : BIADMIN
       Table : ACCOUNTS
           Column : ACCOUNT_ID
           Column : NAME
           Column : LOCK_ATTR
           Column : ADDRESS
           Column : CITY
           Column : PROVINCE
           Column : ZIP
           Column : AREA_PHONE
           Column : PHONE
           Column : AREA_FAX
           Column : FAX
           Column : CREDIT_LMT
           Column : CURRENCY
           Column : GROUP_ACCT
Catalog Folders

   Folder : Accounts
       Column : Account Id  ("BIADMIN"."ACCOUNTS""ACCOUNT_ID")
       Column : Name  ("BIADMIN"."ACCOUNTS""NAME")
       Column : Lock Attr  ("BIADMIN"."ACCOUNTS""LOCK_ATTR")
       Column : Address  ("BIADMIN"."ACCOUNTS""ADDRESS")
       Column : City  ("BIADMIN"."ACCOUNTS""CITY")
       Column : Province  ("BIADMIN"."ACCOUNTS""PROVINCE")
       Column : Zip  ("BIADMIN"."ACCOUNTS""ZIP")
       Column : Area Phone  ("BIADMIN"."ACCOUNTS""AREA_PHONE")
       Column : Phone  ("BIADMIN"."ACCOUNTS""PHONE")
       Column : Area Fax  ("BIADMIN"."ACCOUNTS""AREA_FAX")
       Column : Fax  ("BIADMIN"."ACCOUNTS""FAX")
       Column : Credit Lmt  ("BIADMIN"."ACCOUNTS""CREDIT_LMT")
       Column : Currency  ("BIADMIN"."ACCOUNTS""CURRENCY")
```

Sybase Tables and Impromptu Mappings

The following table shows Sybase DDL statements that create a table named ACCOUNTS using the login BIADMIN, together with the equivalent mapping in Impromptu.

Sybase DDL Statements	Impromptu Mapping	
CREATE TABLE ACCOUNTS		
(Account_Idint not null,	Account_Id	Integer
Name char(50) ,	Name	Char
Lock_Attrchar(1) null,	Lock_Attr	Char
Address char(40) null,	Address	Char
City char(15) null,	City	Char
Provincechar(15) null,	Province	Char
Zip char(6) null,	Zip	Char
Area_Phonesmallint null,	Area_Phone	Integer
Phone char(9) null,	Phone	Char
Area_Faxsmallint null,	Area_Fax	Integer
Fax char(9) null,	Fax	Char
Credit_Lmtmoney null,	Credit_Lmt	Quad
Currencysmallint null,	Currency	Integer
Group_Acctchar(6) null)	Group_Acct	Char

The Catalog overview for the above table looks like this in Impromptu:

```
Database Information

   Logical Name: SYBASE11
   Physical Name:SYBMAIN2|biadmin@%s/%s@0/0@512@ASYNC=1@POLL=100@APPNAME=Impromptu
   Type:      CT

Database Structure

   MetaCatalog : biadmin
      MetaSchema : dbo
         Table : ACCOUNTS
            Column : Account_Id
            Column : Name
            Column : Lock_Attr
            Column : Address
            Column : City
            Column : Province
            Column : Zip
            Column : Area_Phone
            Column : Phone
            Column : Area_Fax
            Column : Fax
            Column : Credit_Lmt
            Column : Currency
            Column : Group_Acct

Catalog Folders

   Folder : Accounts
      Column : Account Id  ("biadmin"."dbo"."ACCOUNTS""Account_Id")
      Column : Name  ("biadmin"."dbo"."ACCOUNTS""Name")
      Column : Lock Attr  ("biadmin"."dbo"."ACCOUNTS""Lock_Attr")
      Column : Address  ("biadmin"."dbo"."ACCOUNTS""Address")
      Column : City  ("biadmin"."dbo"."ACCOUNTS""City")
      Column : Province  ("biadmin"."dbo"."ACCOUNTS""Province")
      Column : Zip  ("biadmin"."dbo"."ACCOUNTS""Zip")
      Column : Area Phone  ("biadmin"."dbo"."ACCOUNTS""Area_Phone")
      Column : Phone  ("biadmin"."dbo"."ACCOUNTS""Phone")
      Column : Area Fax  ("biadmin"."dbo"."ACCOUNTS""Area_Fax")
      Column : Fax  ("biadmin"."dbo"."ACCOUNTS""Fax")
      Column : Credit Lmt  ("biadmin"."dbo"."ACCOUNTS""Credit_Lmt")
      Column : Currency  ("biadmin"."dbo"."ACCOUNTS""Currency")
      Column : Group Acct  ("biadmin"."dbo"."ACCOUNTS""Group_Acct"
```

Microsoft SQL Server Tables and Impromptu Mappings

The following table shows Microsoft SQL Server DDL statements that create a table named ACCOUNTS using the login BIADMIN, together with the equivalent mapping in Impromptu.

MS SQL Server DDL Statements	Impromptu Mapping	
CREATE TABLE ACCOUNTS		
(Account_Idint not null,	Account_Id	Integer
Name char(50) ,	Name	Char
Lock_Attrchar(1) null,	Lock_Attr	Var Char
Address char(40) null,	Address	Var Char
City char(15) null,	City	Var Char
Provincechar(15) null,	Province	Var Char
Zip char(6) null,	Zip	Var Char
Area_Phonesmallint null,	Area_Phone	Small Int
Phone char(9) null,	Phone	Var Char
Area_Faxsmallint null,	Area_Fax	Small Int
Fax char(9) null,	Fax	Var Char
Credit_Lmtmoney null,	Credit_Lmt	Quad
Currencysmallint null,	Currency	Small Int
Group_Acctchar(6) null)	Group_Acct	Var Char

The Catalog overview for the above table looks like this in Impromptu:

```
Database Information

    Logical Name: MSSQL
    Physical Name:SRVR0003|biadmin@%s/%s@APPNAME=Impromptu - Steve@MSVER=65
    Type:       MS

Database Structure

    MetaCatalog : biadmin
        MetaSchema : dbo
            Table : ACCOUNTS
                Column : Account_Id
                Column : Name
                Column : Lock_Attr
                Column : Address
                Column : City
                Column : Province
                Column : Zip
                Column : Area_Phone
                Column : Phone
                Column : Area_Fax
                Column : Fax
                Column : Credit_Lmt
                Column : Currency
                Column : Group_Acct

Catalog Folders

    Folder : Accounts
        Column : Account Id  ("biadmin"."dbo"."ACCOUNTS""Account_Id")
        Column : Name  ("biadmin"."dbo"."ACCOUNTS""Name")
        Column : Lock Attr  ("biadmin"."dbo"."ACCOUNTS""Lock_Attr")
        Column : Address  ("biadmin"."dbo"."ACCOUNTS""Address")
        Column : City  ("biadmin"."dbo"."ACCOUNTS""City")
        Column : Province  ("biadmin"."dbo"."ACCOUNTS""Province")
        Column : Zip  ("biadmin"."dbo"."ACCOUNTS""Zip")
        Column : Area Phone  ("biadmin"."dbo"."ACCOUNTS""Area_Phone")
        Column : Phone  ("biadmin"."dbo"."ACCOUNTS""Phone")
        Column : Area Fax  ("biadmin"."dbo"."ACCOUNTS""Area_Fax")
        Column : Fax  ("biadmin"."dbo"."ACCOUNTS""Fax")
        Column : Credit Lmt  ("biadmin"."dbo"."ACCOUNTS""Credit_Lmt")
        Column : Currency  ("biadmin"."dbo"."ACCOUNTS""Currency")
        Column : Group Acct  ("biadmin"."dbo"."ACCOUNTS""Group_Acct")
```

Informix Tables and Impromptu Mappings

The following table shows Informix DDL statements that create a table named ACCOUNTS using the login bisrv, together with the equivalent mapping in Impromptu

Informix DDL Statements	Impromptu Mapping	
`CREATE TABLE ACCOUNTS`		
`(Account_IdINT,`	`account_id`	`Integer`
`Name    CHAR(50),`	`name`	`Char`
`Lock_AttrCHAR(1),`	`lock_attr`	`Char`
`Address CHAR(40),`	`address`	`Char`
`City    CHAR(15),`	`city`	`Char`
`ProvinceCHAR(15),`	`province`	`Char`
`Zip     CHAR(6),`	`zip`	`Char`
`Area_PhoneSMALLINT,`	`area_phone`	`Integer`
`Phone   CHAR(9),`	`phone`	`Char`
`Area_FaxSMALLINT,`	`area_fax`	`Integer`
`Fax     CHAR(9),`	`fax`	`Char`
`Credit_LmtNUMERIC(9,2),`	`credit_lmt`	`Decimal`
`CurrencyNUMERIC(2),`	`currency`	`Decimal`
`Group_AcctCHAR(6));`	`group_acct`	`Char`

The Catalog overview for the above table looks like this in Impromptu:

```
Database Information

   Logical Name: INF
   Physical Name://srv01/biadminy|sqlinformix@%s/%s
   Type:      IF

Database Structure

   MetaSchema : bisrv
      Table : accounts
         Column : account_id
         Column : name
         Column : lock_attr
         Column : address
         Column : city
         Column : province
         Column : zip
         Column : area_phone
         Column : phone
         Column : area_fax
         Column : fax
         Column : credit_lmt
         Column : currency
         Column : group_acct

Catalog Folders

   Folder : Accounts
      Column : Account Id ("bisrv"."accounts""account_id")
      Column : Name ("bisrv"."accounts""name")
      Column : Lock Attr ("bisrv"."accounts""lock_attr")
      Column : Address ("bisrv"."accounts""address")
      Column : City  ("bisrv"."accounts""city")
      Column : Province  ("bisrv"."accounts""province")
      Column : Zip  ("bisrv"."accounts""zip")
      Column : Area Phone  ("bisrv"."accounts""area_phone")
      Column : Phone  ("bisrv"."accounts""phone")
      Column : Area Fax  ("bisrv"."accounts""area_fax")
      Column : Fax  ("bisrv"."accounts""fax")
      Column : Credit Lmt  ("bisrv"."accounts""credit_lmt")
      Column : Currency  ("bisrv"."accounts""currency")
      Column : Group Acct  ("bisrv"."accounts""group_acct")
```

National Language Portability Issues

Many Impromptu applications are designed in one language and then ported to another language environment. Although Impromptu is not currently a globalized product in the sense that you can develop a single application that supports all languages, there are guidelines that you can follow to minimize the amount of work required.

If in addition to (or instead of) porting from one RDBMS environment to another, you must localize and translate an English catalog to some other national language, you must consider:

- the version of Impromptu that you're using. Impromptu is available in several national language versions. It's possible to use the English version of Impromptu with a locale other than English. For more information about configuring locales, see the installation guide for your version of Impromptu.
- the environment to which you're porting the application. Is the RDBMS available in the target language? Are there NLS versions of integrated applications? Or will you use the English version with the appropriate locale settings for the target language?
- the names that you've assigned to the objects within your catalogs. For example, folder names, calculations, prompts, and descriptions will have to be localized for the particular national language. Even the names of styles that you've created to use in templates and reports may have to be localized.
- colors. Certain colors that are acceptable or have specific connotations in one language have other (possibly negative) connotations in other national languages.
- graphics. Logos that appear in reports must be localized for the particular national language environment.
- field and column widths. Languages such as German and French require more space than their English equivalents. Also, standard paper sizes (for report printing) vary from one country to the next.
- macros. If you have developed macros that prompt for information or display dialogs, you must localize these prompts and dialogs.

Chapter 4: Environmental Considerations

Now that you've designed and built your catalogs, templates, reports, and integrated macros, you can proceed to the deployment phase of the project. During this stage, you'll require a clear understanding of

- where the application files are to be stored
- how you can distribute and secure the application's required files
- which files and application objects you must deploy as part of the application
- how you'll roll the development application into production

By planning for these important factors before you begin report distribution, you make the deployment and ongoing maintenance of your Impromptu reporting system easier.

Impromptu Applications

When you roll out an Impromptu application out to a broad group of users, there are several details to consider to make the ongoing deployment and support as easy as possible. The sections that follow provide guidelines for streamlining the deployment process.

The next sections describe all the pieces that are potentially required in an Impromptu reporting application. The simplest possible reporting application includes:

- a catalog
- a set of reports
- an Impromptu.ini file that contains Impromptu configuration information
- the database connectivity drivers that Impromptu needs to connect to the target RDB
- MS

Most enterprise reporting applications are more complex than this. In addition to the four items listed above, they likely contain:

- hotfiles that store commonly used data
- local and remote snapshots that store query results
- Scheduler files that automate report generation and updating
- graphics used as logos in report headings or as object images in reports
- macros that extend and automate Impromptu, integrating it with other applications
- external applications that use the data Impromptu returns

For truly large installations, things can become even more complex. Impromptu is often used in conjunction with PowerPlay, the Cognos OLAP client. In such cases, you must plan to deploy PowerPlay cubes and reports in addition to your Impromptu reports. Moreover, corporations with hundreds of users often encounter issues with fonts, inconsistent monitor resolutions, inconsistent LAN share mappings, printer drivers, and so on. The sections that follow discuss Impromptu applications in terms of their component parts.

Where Should You Store Application Files?

We recommend that you create standard locations for reports and templates, and that you create FastFind directory locations so users can access their reports quickly and easily. Wherever possible, all files required for an application should be stored in a single folder.

This may be impractical in situations where

- you want to share reports or templates between different Impromptu reporting applications
- you want to use your LAN security to secure access to catalogs, templates, and reports

In such cases, do not hard code paths (drive letters) in any of the required application components (catalogs, templates, reports, macros, or other required files). Instead, place all shared files in a common location and then:

- use UNCs (Universal Naming Conventions) to specify the locations of these objects. When you save reports or catalogs in a LAN environment, Impromptu automatically does this for you. However, you should make it a habit to use UNCs when saving application files.
- grant all affected users access to the shared location, and to the locations of any related or dependent files
- ensure that all affected users map the same drive letters to the same LAN shares

If you move an Impromptu application, ensure that you maintain a consistent folder hierarchy for any dependent files in the new location. When an Impromptu file (catalog, report, or template) is moved to a new location, it will attempt to open dependent files by building new relative paths based on the known paths before the move.

Note: At runtime, Impromptu's OLE API does not use the same search algorithm when looking for dependent files as do interactive Impromptu catalogs and reports. If you create a macro that depends on a hard-coded location and then change that location, you must update the macro accordingly.

For example, suppose that your Impromptu application is currently stored in a folder named `Finance` in a LAN share named `\\SRV01\Reports\`. Assume also that you've stored the application's common files for different groups as follows:

Dependent Files	Location	Relative Path
Graphics for Logos	`\\SRV01\Reports\Graphics`	`..\Graphics`
Common picklist reports	`\\SRV01\Reports\Picklists`	`..\picklists`
Reports for Senior Management	`\\SRV01\Reports\Finance \CORPMGMT`	`./CORPMGMT`
Reports for various cost centers	\\SRV01\Reports\Finance \*<cost center name>* where <cost center name> is a set of folders that contain reports for specific cost centers, named according cost center.	`./<cost center name>`

If you move the entire application to a new location such as `\\SRV09\Daily\Finance Reports`, then Impromptu will attempt to open the dependent files using paths relative to the new location. So, at runtime, Impromptu would look for dependent files as follows:

Dependent Files	Location
Graphics for Logos	`\\SRV09\Daily\Graphics`
Common picklist reports	`\\SRV09\Daily\Picklists`
Reports for Senior Management	`\\SRV09\Daily\Finance Reports\CORPMGMT`
Reports for various cost centers	`\\SRV09\Daily\Finance Reports\<cost center name>` where <cost center name> is a set of folders that contain reports for specific cost centers, named according cost center.

How Impromptu Searches for Dependent Files

Impromptu catalogs and reports include information about where they're saved. Whenever you open a catalog or report, Impromptu compares the file's current location with the location stored when the file was last opened. If these are different, Impromptu updates the file storage location information, and uses the new location to build relative paths to dependent files.

When opening "dependent" files at run time, Impromptu looks for related files based on several steps:

1. Use the UNC path that is stored in the dependent file being opened.
2. Use the current directory. Most often, this is the directory of the parent (catalog or report) that is trying to locate the dependent file.
3. Replace the drive letter (or UNC "\\SERVER\SHARE" portion) in the path with the parent's drive. In other words, attempt to use the existing path, but use the new drive or LAN share.
4. Compare the parent's original path and the dependent file's original path, attempt to derive a relative path pointer, and search using only this relative path.
5. Use the path of the parent catalog or report. This may or may not be the same as the path in step 2.
6. Use the path specified for the Workspace location in the Preferences dialog box.
7. If Impromptu is looking for a catalog, use the path specified for the Catalog Location in the Preferences dialog box.
8. If Impromptu is looking for a report, use the path specified for Report location in the Preferences dialog box.
9. If the file is still not found, use the standard Windows file location algorithm, which looks in standard locations such as the current directory, the user's search PATH, the Windows and Windows\System folders, and so on.

Impromptu applies the preceding steps differently for different kinds of reporting objects in different contexts. The following tables list the steps that Impromptu applies when it tries to locate various kinds of dependent files.

How reports and templates locate dependent files

Dependent File Type	Steps
associated catalog, hotfiles	1-6, 7, 9
picklists, dataset filters, drill-through reports, picture frame (either loaded from file or with a prefix), HTML # includes, Auto-open macro	1-6, 8, 9
OLE object, loaded from a file	1

How catalogs locate dependent files

Dependent File Type	Steps
master copy of local distributed catalog	1
hotfiles	1-6, 7, 9

How Impromptu locates other dependent files

Dependent File Type	Steps
application startup macro	1-6, 7, 9
default catalog, last used catalog, files opened via Launch buttons, reports opened via Most recently Used (MRU) list in File menu	1

Search Algorithm Side Effects

Impromptu's sophisticated search algorithm has one potentially important side effect. In certain cases, a user may think he or she is connected to a particular catalog when in fact they are connected to another catalog.

For example, suppose that a user opens a report that is normally associated with the following catalog:

```
\\srv0001\cognos\impromptu\finance\finance.cat
```

As the report opens, if it fails to find the catalog but finds another `finance.cat` catalog using the search algorithm described above, the user may still think he or she is connected to the original catalog. For this reason, it is recommended that catalogs be assigned meaningful descriptions.

Copying Catalogs

In some projects, you might have to physically copy a catalog to a new location, and then have users open their reports against the new copy. This technique is useful when you wish to translate or rename objects in a catalog, as you might when translating a catalog from one language to another. It's also useful if you need to replicate a catalog at a remote location and subsequently open reports against that replicated catalog. The key is to copy the master catalog and then apply the changes to the copy.

When you make a copy of a catalog, and even change the names of the tables or columns, users can open reports against the catalog copy and not raise an error. Within the copied catalog, Impromptu retains the original metadata and object identifiers that were stored in the original catalog. When associating a report with the copied catalog, Impromptu determines whether the parent catalog identifiers it saved in the report match those in the .cat file it has opened. If they match, it binds the report object to the catalog objects using these internal object identifiers.

Associating Reports with Different Catalogs

In most cases, Impromptu reports remain associated with the catalog that you use to create them. When you associate a report with a catalog other than the one the report was created with (or a copy of that catalog), Impromptu displays a message to this effect and attempts to bind the report objects with the catalog. Impromptu uses the TABLE and TABLE_COLUMN names stored in the report to bind report objects to catalog objects. If the catalog does not contain these objects, or if their names are slightly different, Impromptu fails to find a binding.

Note: Table and column names are case-sensitive. Changing the case of a catalog object can cause a name match to fail. To enable portability of reports, always use case-blind names (uppercase names). You can check TABLE and TABLE_COLUMN names using either the Table dialog box or the Folder dialog box, where Impromptu displays qualified names.

When Impromptu cannot bind a report object to a corresponding catalog object, you will encounter an error message like the following:

```
**** ERROR ****

Data item [\Account Id] is invalid because [[ORASOL71.BIADMIN.ACCOUNTS.ACCOUNT_ID]]
could not be found.
```

This name matching does not care if the names in the preceding levels to the TABLE object are different.

How Can You Tell Which Catalog a Template is Associated With?

For large applications, it's possible that you'll have a large number of templates and at some point you may lose track of which templates are associated with which catalogs. In such cases, you can use the Open dialog box to find out which reports are associated with the currently active catalog.

In the Open dialog box, the Filter by Catalog check box controls which reports and templates Impromptu displays. If you enable Filter by Catalog, Impromptu displays only reports and templates that are associated with the active catalog.

Locating Calculations

A report can contain calculated expressions that are local to the report or referenced from the catalog. When Impromptu attempts to resolve by name a calculation from a folder, it uses the folder name and expression name. If this search fails, Impromptu displays one of the following error messages:

```
**** ERROR ****

Data item [\Folder2\CALC1] could not be found.

**** ERROR ****

Data item [\CALC1] is invalid because it contains additional invalid Data Items.
```

When the expression is in a deeply nested folder, the message shows the hierarchy

```
**** ERROR ****
Data item [\T1\Folder1\Folder2\CALC1] could not be found.
```

At this point, you will have to recreate the definition of the calculated column CALC1. If you do not know what the correct definition is, you may have to locate the original catalog in which CALC1 was defined and obtain the definition from that catalog.

If your catalog includes user classes, Impromptu does not consider folder security when re-binding. So, even if a user does not normally have access to a folder that contains a calculation, Impromptu successfully finds and binds to a calculation.

Using FastFind Effectively

Impromptu's FastFind feature is a convenient way to provide user groups with quick entry points to catalogs and reports. You can specify a location, known as the FastFind location, that Impromptu uses to provide quick access

- when users are creating reports
- to reports, templates, catalogs, and shortcuts to such objects when users are opening or saving files

The FastFind location is stored in the Impromptu.ini file. You can change the FastFind location in the File Locations tab (labelled "report directory"). You have the following options when setting up FastFind locations for your Impromptu users:

- set up a single FastFind location for all users of a given application, and include shortcuts to common catalogs, templates, reports, and other files that are part of the application. Also include shortcuts to folders that contain such objects. To tailor the way in which FastFind displays tabs at runtime, use your LAN security to grant or deny access to the folders you've included.

  This option is preferred, as it meets the users' needs without making it more difficult to maintain the Impromptu.ini file. It provides users a common entry point to Impromptu, and ready access to common files. It also gives you a mechanism to provide group-dependent files for individual groups without affecting other groups. If the location of the templates and reports changes, you need only modify the shortcuts in the FastFind folder.
- set up a unique FastFind location for each Impromptu user group

  This option provides the most control in terms of Impromptu's interaction with users as they create, open, and save files. However, because the FastFind location is stored in Impromptu.ini, this approach requires you to deploy multiple Impromptu.ini files (one for each different FastFind location). As you manage changes to other attributes stored in Impromptu.ini, you'll have to update multiple Impromptu.ini files.

Using Workgroup and User Templates Effectively

Impromptu's Workgroup and User template feature allows you to set up templates that are shared among your application's users. You can use this feature to provide users with consistent access to templates across an entire application.

In the File location dialog box, you can specify folders for Workgroup templates and User templates. When users create a new report, Impromptu displays the New dialog box. Impromptu populates the New dialog box with tabs for the FastFind folder, as well as folders in the Workgroup Templates and User Templates locations. However, Impromptu only displays a tab for a Workgroup or user Template folder if the following two conditions are true:

- the folder contains at least one Impromptu template
- the user creating the report has permission to access the folder

These features let you to store common templates for your users centrally, in folders within the Workgroup Templates location. It also allows your users to store their own templates separately, usually in a local drive on their personal computer. Moreover, it lets you selectively control for different groups of users the tabs that Impromptu displays in the New dialog box. If your LAN security denies a specific user access to a folder within the Workgroup locations folder, then the New dialog box does not display a tab for that folder for that user.

Notes

- Do not build hierarchical folder structures within Workgroup and User Templates folders. The New dialog box only displays templates in folders immediately below the Workgroup and User Templates folders.
- Before displaying the New dialog box, Impromptu scans folders within the Workgroup and User Template locations to verify that they contain templates. If your application uses a large number of templates and folders, this scanning can delay the appearance of the New dialog box at runtime when users create reports.

Impromptu Application Files

The previous section describes how you should package Impromptu application files for wide deployment. This section describes the kinds of files that you must consider when deploying an entire application.

The Catalog

Most Impromptu applications require a catalog. For an application that uses a shared, secured, or distributed catalog, you should place the catalog centrally on a LAN folder to which all affected users have access. This folder becomes the core folder for the application, and you should strive to store as many of the application's required files in this folder as possible.

Also, you should use Impromptu's FastFind feature to ensure that your users have quick access to the catalogs they require.

See "Where Should You Store Application Files?" (p. 31)

Database Access Drivers

Each user will require a copy of the appropriate database access drivers. For native access to databases such as Oracle and Sybase, this includes native database client applications such as SQL*Net and Open Client. Before you set up users with Impromptu reports, verify that they can access the target RDBMS system using that database's tools. For example, verify that Oracle users can access the target RDBMS using SQL*Plus.

For databases accessed via an ODBC driver, each user will require a supported ODBC driver and associated DNS entry for the database being accessed.

Templates

If users' needs are best met with standard reports and they don't perform ad hoc queries, then you need not implement any templates for them. You only need to create templates for yourself. Conversely, if you plan to make ad hoc reporting widely available to users, you should provide a set of templates that makes report creation easy.

Creating templates for ad hoc reports serves many purposes:

- it provides all users with a common look and feel for their reports
- it allows you to impose corporate standards on all reports that your users generate
- it provides guidance for less advanced users, with built-in layout, formatting characteristics, and placeholders that help users create reports without having to learn advanced Impromptu features

If your templates are driven by placeholders, you lower the risk of users inadvertently adding the wrong information to a report. When you create placeholders, Impromptu displays a Label and a Prompt when the user builds a report. It displays the Label in the report (for example, as a column heading in a list report), and the Prompt as the placeholder name. You could create a placeholder with the Label "Customer ID" and the Prompt "Add CustID from CUST_MSTR". The placeholder itself tells users which data to use to fill the placeholder.

The *Impromptu Administrator's Guide* and the *Mastering Impromptu Reports* online books provide information about how to create templates. This section describes issues to consider as you design templates for wide distribution with your reporting applications.

You must deploy templates with the catalogs that you provide. When you deploy them, you must follow all of the dependency rules that apply to Impromptu catalogs and reports.

There are two common ways to deploy templates. You can:

- Provide templates via email, or put them in a LAN location and have users add the templates they want to their User Templates or FastFind location. Where possible, store all templates and reports in the core application folder that contains the catalog.
- Put templates in a Workgroup Templates location, and replace users' Impromptu.ini files so all these standard templates are available to them. See "Using Workgroup and User Templates Effectively" (p. 36).

When deploying templates, use styles consistently. Take control of styles and don't allow them to proliferate. If you create three or four variants of a style, you may make it difficult to maintain them over time.

Reports

You can deploy reports in several ways. You can:

- Publish reports in central LAN folders that all users access to open reports. To meet the needs of various users, you can publish reports in hierarchical LAN folders, and use LAN security features to grant users access to the folders that contain only their reports and catalogs.
- Use Cognos Scheduler to run reports automatically according to a schedule, and then either distribute them to users or publish them on a LAN.
- Use Portfolio to set up briefing books that let users view reports in a presentation style format.
- Publish reports as HTML (HyperText Markup Language) so users can view them with a Web browser.

You can also print and distribute reports for your end users.

Physical Deployment Considerations

Depending on the nature of your application and environment, you must consider the portability of reports from several points of view:

- Do the reports you've provided travel well and display correctly for your users? Physical constraints, such as fonts, colors, and monitor resolution can affect the way objects are displayed from one personal computer to the next.
- For more complex projects in which you create and distribute multiple catalogs, will users be able to open reports using catalogs other than the one the report was created with?
- Does a given report depend on other reports? In other words, are there certain reports that you must always package and deploy together?

Most of these considerations involve physical constraints that are imposed by your environment. For detailed information about these, see "Impromptu Applications" (p. 31).

Naming Frames

One of the simplest ways you can make templates and reports easier to deploy and maintain in a large-scale reporting environment is by adopting standard naming conventions for Impromptu frames. By default, Impromptu assigns names like "Form Frame" or "Form Frame 1". You can make frames more usable and identifiable by assigning meaningful names to them. Additionally, by naming frames consistently, you'll make it easier to create and maintain macros that refer to individual frames.

Helping Users Find Reports

You can use Impromptu's FastFind feature to present users with common entry points to the reports that you provide them. For more information, see "Using FastFind Effectively" (p. 35).

If you are using secured LAN folders to control which reports and catalogs a user can view, do not hard code paths to the catalog or to any reports. Instead, use UNCs (Universal Naming Conventions) to point to common folders that contain shared report objects.

Hotfiles, Snapshots, and Remote Snapshots

When applications use Impromptu hotfiles and snapshots, these files must be distributed with the catalogs and reports that reference them.

For local snapshots, Impromptu stores report data within the report itself, so the potential deployment issue is report size. Snapshot reports are significantly larger than equivalent reports that do not include a snapshot of the data. As a rule of thumb, you should limit snapshot files to between 5,000 and 8,000 rows of data.

Macros

If possible, locate all macros that an application uses together with the catalog and reports.

When writing macros, do not hard code paths to any of the files that you reference. At runtime, the hard-coded paths may not resolve to the appropriate values for users with different LAN folder mappings. Instead, use UNCs to point to common macros and any external DLL functions that your macros reference. When you make the application available to end users, they will not encounter problems finding macros at runtime, provided they have appropriate LAN access to the share where you've stored macros, and the physical locations have not changed.

Another factor to consider is event-driven macros. When you create macros that are intended to update specific reports or result sets, you should attempt to "trigger" the updates on known events. For example, if a group of your users depends on the data in a specific hotfile, it makes sense to regenerate reports only when that hotfile changes. You can write a macro that checks for changes to the hotfile, and regenerates reports only when its associated date has changed.

External Applications

Where possible, use OLE Automation to launch external applications rather than executing the applications from a launch button. OLE Automation provides greater control over the synchronization of commands and their execution. Simply executing the application passes control of the operating system to that application, with no way to control its operating characteristics.

Also, ensure that all users' personal computers are configured to launch the external application correctly. This means ensuring a common setup for these external applications across the enterprise.

Scheduler Files

Scheduler automates the running of processes locally. You can use Scheduler to:

- allow users to schedule their own local Impromptu reports and other tasks
- set up remote snapshots and notify users when they've been updated

INI Files

There are several INI files that contain Cognos application configuration settings, many of which are Impromptu specific. These include database connection strings so that users can connect to the target database.

Each user must have a Cognos.ini file that contains the appropriate database connection strings. To grant access to a new database, or to associate a catalog with a physical instance of the database other than the one it was developed with, you must deploy the new database connection settings to all affected users.

The Impromptu.ini file contains many settings that affect Impromptu runtime execution. These range from the default directories for storing files to report styles that you've defined and used in your standard reports.

For more information about INI file settings, see the *Impromptu User Reference*.

To ensure consistency within your user community, you should maintain strict control of the Impromptu.ini file. To do so, you can store the most up-to-date copy of the Impromptu.ini file in a LAN location that's accessible by all affected users. You can then implement either a push distribution mechanism (in which you automatically update your users' local Impromptu.ini files) or a pull distribution mechanism (where users copy the current Impromptu.ini file to their personal computers).

User-defined Function Files

Impromptu supports both Database and External User Defined Functions. Implementing either requires you to make changes in the database initialization files supplied with Impromptu. If you extend Impromptu's capabilities with respect to functions, you must plan to deploy any changes that you've made to Impromptu's function initialization files. To avoid having to distribute dll files, use an RDBMS function in favor of a local user-defined function wherever possible.

For details about how to create user-defined functions and the database initialization files affected, see the online book *How to Create User-defined Functions*.

Deploying initialization files in the Windows NT environment requires that you do one of the following:

- If users are running Impromptu from a LAN, then you can replace only the ini files for that Impromptu instance. This gives users access to the user-defined functions that you've created.
- If users are running Impromptu locally on their PCs, ensure that their database initialization files are replaced with the ones that you've updated with UDF information. You can post these files on a central LAN server and have users replace their local copies. You should store these files in the same location as the Impromptu.ini file.

Font Definition Files

If your application uses non-standard TrueType or Postscript fonts, ensure that your users have these fonts installed on their personal computers. If fonts are not found at runtime, then the operating system will substitute fonts, with unpredictable results.

HTML Files

A HyperText Markup Language (HTML) report is a read-only report users can view with a Web browser. With Impromptu, you can save and distribute reports in HTML format. Anyone with a Web browser can view an HTML report. You do not need Impromptu to view an HTML report.

Impromptu creates different file types when you save a report as HTML. The quantity and types of files depend on the size of the report, the number of graphics, and the options that you include. When you distribute an HTML report, you must provide all of its associated files.

Notes:

- To avoid problems with older browsers that do not support background colors, do not use text colors that may be used as Web browser backgrounds, such as gray or white.
- To minimize the size of the HTML files created, make the background pattern of objects transparent wherever possible.

Storing and Distributing HTML Reports

Once you save a report as HTML, you must make it available to report viewers so they can look at it with a Web browser. You must include all files associated with the HTML report. You can distribute HTML reports by

- making them available on an Internet or intranet Web site
- sending them to specific users via email
- making them available on a network

If you are consistently updating a set of standard HTML reports, use your Web Server software to set up aliases for the locations in which you publish reports. This insulates users from future changes in the location of these reports. You can also assign these aliases meaningful names. Check with your webmaster to find out if it's possible to set up aliases for your Impromptu HTML reports.

Enabling Searches for HTML Reports

When you save a report as HTML, Impromptu automatically includes <meta name> tags in the page header of each report page and on each Table of Contents page. These tags let you (and the report users) search for and index reports. The Web master in your organization can provide a search capability that lets users search for all HTML reports with a particular <meta name> string.

The <meta name> tags in the report page headers appear as follows:

```
<meta name= "Generator" content="Cognos Impromptu Report">
<meta name= "Report" content="report file name without a path or extension">
<meta name= "Catalog content="catalog file name without a path or extension">
```

You can enhance Impromptu's automatic <meta name> tags to add metadata information to your HTML report output. Impromptu inserts your supplemental tags in the page header of each report page, and on each Table of Contents page for the published report. These supplemental tags provide you with information you can use to generate links between HTML reports. You can write scripts (using Perl or some other language) that parse the meta name tags, and generate links as required.

The additional tags are:

```
<meta name= "Data content"= "column name, unique catalog ID number, data type">
<td> column value<!--#data item--></td>
```

For example

```
<meta name= "Data content = "
    Country, 147, Character,
    Product Type 86, Character,
    Product Line, 87, Character,
    Total Sales 94, 0, Number,
    Total Sales 95, 0, Number">

<td>Recycled Products<!--Product Line--></td>
```

These tags provide information about which catalog and columns within the catalog were used to generate the report. With these tags in the header of HTML report output, it's possible to search for and link reports based on the context of the source catalog.

To enable enhanced HTML tags, edit the Impromptu.ini file to include the following information in the Startup options.

```
[Startup Options]
Export HTML Metadata=1
```

Using Remote Snapshots

A remote snapshot is a report that has its data stored on a database server such as Informix, Oracle, or Sybase. When users access remote snapshots, they can

- access query results quickly, without executing a complex query
- easily ensure that the remote snapshot contains the most current data by updating the data stored on the server
- perform filter operations on the remote snapshot
- save the remote snapshot as a local snapshot. This allows users to work without a network connection when they are travelling. Users can update the local snapshot with data from the remote snapshot as required.

When users access a remote snapshot, Impromptu runs a query against the remote result set. When accessing a remote snapshot, users do not need to use a catalog. As a result, users can use remote snapshots even if they are not authorized to log on to the catalog.

Notes

- The icon in the Preview box (Open dialog box) and in the lower right corner of the screen indicates the type of snapshot:

  Remote Snapshot

  Local Snapshot
- You cannot save a remote snapshot as a Hotfile.
- You can reconnect a remote snapshot to the database by clicking Database (Access tab, Query dialog box). This action is irreversible.

Going Into Production

Before you deploy your application, we recommend you test the catalogs, templates, reports, and any associated files with selected users. Choose a representative user from each user class, and set that user up with access to the system. Monitor the progress of your test users for a set period of time, collect feedback, and implement any required changes.

Maintaining Catalog and Report Dependencies

When you move your finished reporting application onto a production server, move the entire folder structure as is, and then grant users access as required. This will help minimize the likelihood of a report being unable to locate a dependent report or catalog. For more information, see "How Impromptu Searches for Dependent Files" (p. 33).

Moving to the Production Database

Putting a system into production often means rolling that system from a development environment to a production environment. For Impromptu, this means pointing your catalog to the production database rather than a development environment.

To have an Impromptu catalog reference another database, you can create a new database definition and change the reference in the Catalog Tables dialog box. For example, if you defined your catalog against a Sybase SQL Server database named TEST_SAL, on a server named SRV001, you would see the following entry in the Cognos.ini file:

```
SYSTEM_SALES=^User ID:^?Password:;LOCAL;SY;SRV001|TEST_SAL@%s/%s@APPNAME=Impromptu
```

If you then wanted to change the system so that the catalog pointed to an identically-structured database named SAL_ADM on server SRV020, you'd use Impromptu's Database Definition dialog box to make the appropriate changes. The revised entry in the Cognos.ini file would look like this:

```
SYSTEM_SALES=^User ID:^?Password:;LOCAL;SY;SRV020|SAL_ADM@%s/%s@APPNAME=Impromptu
```

Chapter 5: Support and Maintenance Considerations

This chapter discusses the implications of making ongoing changes to your Impromptu reporting system. It provides guidelines about the effects of making changes to

- the RDBMS tables and columns referenced in a catalog
- the folders, columns, calculations, filters, and prompts in a catalog, and how these changes affect reports
- report and template attributes
- the overall reporting environment
- the version of Impromptu that is installed at your site

Support and Maintenance Considerations

It's a mistake to think that once you've rolled out your finished reporting system to users, it's complete and will require only minor, occasional modifications. Business Intelligence systems evolve constantly, changing as business needs change. Plan for iterations of report development. As users recognize the benefits and power of ad hoc reporting, you'll encounter numerous requests for increasingly more complex reports. You might also be asked to modify the catalog to meet new requirements.

Consider the following questions:

- How will you deal with changes to the structure or content of the system's Impromptu catalogs? How will you analyze the impact of such changes?
- How will you deal with changes to the data structures against which you've created Impromptu catalogs? Is it likely that the tables and columns in the target RDBMS will change?
- Will you need to migrate an existing Impromptu catalog to a newer version? If so, you will not be able to open the new catalog with the older version.
- If you've created a staging data warehouse as part of the system, how often will its content need to be refreshed? How long will the update take?

Reporting systems typically work on non-volatile data that is stored in a data warehouse. Updating the information in the data warehouse and reports is a regular activity that you must plan for. If reports must be updated on a daily basis, you must ensure that the update process takes only a few hours. An update process that takes 22 hours is probably unacceptable in such a case, as the data would be out of date before it's released.

Changing Catalog Objects

When you change the name of a catalog object, such as a catalog qualification level or a schema qualification level, Impromptu changes only the external name for the object. Internally, Impromptu identifies the objects by unique object IDs. Once a report is bound to the catalog, the SQL generated by Impromptu automatically reflects the new catalog and/or schema qualification changes. Additionally, once a report is created, that report stores the catalog metadata IDs for the report columns it contains.

For example, suppose that you create a report that references a folder column named Customer Number. If you change the name of the column to Customer ID within the catalog folders, any reports that you've created will continue to run correctly even if they contain references to the old name, Customer Number.

Note: You cannot change the names of tables and their associated columns within a catalog unless you are doing so because they have changed in the database. These must match the tables and columns that Impromptu looks for in the RDBMS.

Deleting and Recreating Catalog Objects

When you create a new object in a catalog, Impromptu assigns it a unique object ID. When you drop an object and later re-create it (using either automation scripts or the Tables dialog box), Impromptu considers the object a new object, and assigns it a new object ID. When binding objects in a report to the new object in the catalog, Impromptu rebinds using the original TABLE and COLUMN names.

See "Associating Reports with Different Catalogs" (p. 34)

When do You Have to Recreate Catalog Objects?

If the database structure underlying a catalog changes, you may have to recreate the catalogs that reference the changed objects. However, if only the names of tables and columns change, you can use the Tables dialog box to rename the catalog's tables accordingly. For example, if the name of a table in the database changes from CUST_MAST to CUSTOMERS, you can rename the old table. Because Impromptu maintains metadata derived from the database, it is incapable of finding tables and columns whose names have changed in the database unless you change them to match within the catalog.

In such cases, you'll have to regenerate the reports you've created against the original catalog.

When do You Have to Regenerate Reports?

If you change the names of folders within a catalog, Impromptu reports you've created will continue to run. Any name changes that you make within folders are merely external; they do not affect the underlying catalog object identifier that the reports use to bind to catalog objects.

However, if you invalidate the catalog columns by changing the names of tables and/or columns within tables in the catalog, reports that you've created will fail if they attempt to reference these objects. Moreover, if you attempt to create new reports that reference the changed columns and tables, Impromptu will issue an error.

If you delete and recreate a catalog calculation, condition, or prompt, any reports that reference the old object names will issue an error message and not execute. You will have to rebuild these reports and use the new names.

Changing Templates and Styles

If you modify the templates that you provide for your users, ensure that you provide adequate information for them to understand the new templates. This is especially true if you replace or modify placeholders in such a way that users might mismatch source columns with their intended placeholders.

If you add new styles or change existing styles in the reports that you distribute, you'll need to deploy a new version of the Impromptu.ini file. You could also copy the style text to a separate "master" style ini file that you create, and then write a macro that reads this file and updates users' impromptu.ini files. You can use the Windows Get and Write profile string routines for this purpose.

Automating Deployment

A big part of most Impromptu applications is the automation of the update and distribution process for:

- catalogs, reports, and templates
- hotfiles and snapshots
- macros
- supporting files, such as the Impromptu.ini file, graphics files, and briefing books

This section describes ways that you can automate these aspects of an Impromptu application.

Creating Macros and Automation Scripts

The CognosScript macro engine provides a mechanism for automating common Impromptu, and Cognos Scheduler tasks.

You can set up scripts that create Impromptu snapshots on the LAN, based on the occurrence of specific events. This approach allows you to further interact with the data after it has been refreshed.

The most common automation tasks you should consider implementing are:

- automatic emailing of notifications for completed Scheduler requests (or URLs, for reports published as HTML)
- automatic mailing of notifications when a master distributed catalog has changed
- automatic notification when a new version of the Impromptu.ini file is made available
- automatic exporting of Impromptu reports when the data returned is required for other systems (Excel spreadsheets, for example)

If you are integrating Impromptu with other applications, you should always use OLE API's instead of launching these applications from a Launch button. See "Macros" (p. 38)

Example

The following Visual Basic script shows a technique to associate .imr files with a catalog. The code sample does not use the OpenReport method to open the Impromptu reports. The OpenReport method cannot tell Impromptu to use the currently open catalog, regardless of the catalog name in the IMR file or whether Impromptu finds the parent catalog. When OpenReport cannot find the catalog, it raises an error when Impromptu is not visible, or displays a file open dialog when Impromptu is visible.

To address these issues, the macro uses the SendKeys statement to issue commands to the currently active Impromptu instance. Unlike OLE automation, SendKeys does not cause the script to block (wait) on a method call if the method causes a modal dialog box to appear. As a result, the SendKeys commands can become desynchronized with displayed dialogs. To overcome this problem, the code includes delay timers that provide sufficient time for Impromptu to display dialogs.

In the context of the script, dialogs appear in two conditions:

- Impromptu has detected that the catalog the report is being connected to is not the same as the original
- Impromptu has failed to successfully rebind the columns using the name matching algorithm.

There is a subtle point associated to the final failure condition. Even though the binding failed, Impromptu considers that the report has changed. When the CloseReport method is called, it will display a Yes, No, Cancel dialog to save the changes. A SendKeys statement forces a "No" response.

The macro also includes some basic logging of which reports failed to convert and a simple summary.

```
Option Explicit
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Private mGoodConverts As Long
Private mBadConverts As Long
Private mImpRep As Object
Private mFileNo As Integer
Private mOpened As Boolean
Private mImpApp As Object

Private Sub ConvertError(InName As String)
    ’ serious problem loading in the database.
    ’ Report the error and terminate
    Const ERRLOG = "C:\TEMP\Conv.Log"
    mBadConverts = mBadConverts + 1
    If Not mOpened Then
        mFileNo = FreeFile
        Open ERRLOG For Append Access Write As #mFileNo
        mOpened = True
    End If
    Print #mFileNo, InName
    If Not mImpRep Is Nothing Then
        Print #mFileNo, mImpApp.GetNextQueryError
    End If
End Sub

Private Sub ConvertSummary()

    Print #mFileNo, "Number of conversions attempted (" & (mBadConverts + mGoodConverts)
& _
        ")" & vbCrLf & " Successful (" & mGoodConverts & ") Failed (" & mBadConverts & ")"
    Close #mFileNo
End Sub

Sub Main()
    Const DIRPREFIX = "D:\Temp\"
    Const CATFILE = DIRPREFIX + "Cat02\Cat02.cat"
    Const REPORTDIR = DIRPREFIX + "Reports\"

    Dim mFile As String
    Dim mConvertFile As String
    On Error GoTo CloseDown

    Set mImpApp = CreateObject("Impromptu.Application")
    mImpApp.OpenCatalog CATFILE, "Creator", , , , 0
    mImpApp.Visible 1
    mFile = Dir(REPORTDIR + "*.imr")
    Do While mFile <> Empty
        mConvertFile = REPORTDIR + mFile
        AppActivate "Impromptu"
        Sleep (2000)
        SendKeys "^O" + mConvertFile + "{Enter}" + CATFILE + "{Enter}", True
        Sleep (2000)
        SendKeys "{Enter}", True
        Sleep (2000)
        SendKeys "{Enter}", True
        Set mImpRep = mImpApp.ActiveDocument
        If Not mImpRep Is Nothing Then
            If "" = mImpApp.GetNextQueryError And mImpApp.geterrornumber = 0 Then
                mGoodConverts = mGoodConverts + 1
                With mImpRep
                    .save
                    .closereport
                End With
            Else
                ConvertError mConvertFile
                mImpRep.closereport
                Sleep (2000)
                SendKeys "%N", True
            End If
        Else
            ConvertError mConvertFile
        End If
        Set mImpRep = Nothing
        mFile = Dir
    Loop
    ConvertSummary

CloseDown:
    If Err.Number Then
        MsgBox Err.Number & " " & Err.Description
    End If
    If Not mImpApp Is Nothing Then
        mImpApp.Quit
    End If
    Set mImpRep = Nothing
    Set mImpApp = Nothing
End Sub
```

Scheduling Report Distribution

The Cognos Scheduler coordinates the execution of automated processes, called tasks, on a set date and time, or at recurring intervals. Scheduler supports tasks that run once and tasks that run repeatedly. Using Scheduler, you submit Impromptu report requests to run locally.

Scheduler can run tasks completely unattended because you can specify security parameters for your databases and catalogs within Impromptu or Scheduler. Scheduler stores passwords in an encrypted format so they cannot be seen when the tasks are running.

You can create tasks that run:

- a CognosScript macro that automatically runs a daily production report or builds a HotFile every night of the week, starting at 2:00 A.M.
- an Impromptu report and save it as a HotFile.
- a CognosScript macro that automatically creates a set of standard Impromptu reports once a month and then emails the results to a group of users.
- Portfolio, to update its OLE links every morning at 7:00 A.M. to ensure that briefing books contain the most recent data.

Tips

- To eliminate the need to regularly modify standard reports, ensure that any dates in the filter expression of a standard report are created with functions rather than hard-coded dates. For example, use (add-years (today()),-1) which changes according to the current system date.

  For information about functions, see the Impromptu online Help. In the index tab, type

  - functions
- Every macro you create for your users implies an additional file (or set of files) that you will have to deploy with the application. The compiled macros that you create must be available and locatable from Impromptu for each user who wants to use the macro. Also, the Impromptu OLE interface is more sensitive to changes in file location than the Impromptu client. Avoid moving files referenced within macros from one location to another.

Using Scheduler to Automate HTML Publishing

You can use Cognos Scheduler to program the generation of HTML report updates for specific or recurring times and dates, or on a recurring basis. Scheduler can open and save a report as HTML at the same time. You can use Scheduler to update all the reports on your Website after a regularly scheduled database update to ensure that the reports on the Website are always current.

Automating Support Tasks

An often-neglected task in application rollout is preparing instructions for your operations group, and ensuring that any supporting automation scripts are in place for them. For example, you should make it clear when your application files are to be backed up, what to do in the event of an update failure, and so on.

Version and Migration Issues

Like most production systems, Impromptu applications must be upgraded to new versions as they become available. This section describes issues you should consider as you upgrade from one version of Impromptu to another.

Impromptu always supports forward migration. For example, you can move Impromptu version 5.0 catalogs and reports forward so that they're readable and writable with version 6.0. When a newer version accesses a catalog or report from an older version, Impromptu asks you how to proceed. You can do either of the following:

- Open the catalog or report without updating it to the format supported by the new version. This allows you to access (view only) catalogs and reports from older versions.
- Open the catalog or report and update it to the new version format. Once you do this, you'll be unable to access the updated catalog or report with the older version.

These options allow a newer version of Impromptu to co-exist with documents created by older versions of Impromptu. However, to maintain catalogs and reports, you must continue to use the version of Impromptu that matches your users.

Notes

- Impromptu version 3.5 is the only version for 16-bit Windows and Windows NT 3.5. All later versions are supported in Windows NT 4.0 only. If you must deploy to a 16-bit environment, you will have to continue to work with Impromptu 3.5.

Index