

INDEX

Chapter-1: Introduction	1.
Chapter-2: Using a Designer	15.
Chapter-3: Working with sources	17.
Chapter-4: Working with targets	19.
Chapter-5: Working with transformations	21.
Chapter-6: Aggregator	25.
Chapter-7: Expression	33.
Chapter-8: Filter	36.
Chapter-9: Joiner	39.
Chapter-10: Lookup	53.
Chapter-11: Rank	75.
Chapter-12: Router	80.
Chapter-13: Sequence Generator	87.
Chapter-14: Sorter	96.
Chapter-15: Source Qualifier	101.
Chapter-16: Stored Procedure	120.
Chapter-17: Union	137.
Chapter-18: Update Strategy	140.
Chapter-19: XML	145.
Chapter-20: Mapplets	146.
Chapter-21: Mapping Parameters & Variables	155.
Chapter-22: Workflow Manager	164.
Chapter-23: Workflows and worklets	167.
Chapter-24: Sessions	176.
Chapter-25: Tasks	178.

CHAPTER 1

Product Overview

This chapter includes the following topics:

- ϕ Introduction
- ϕ PowerCenter Domain
- ϕ PowerCenter Repository
- ϕ Administration Console
- ϕ Domain Configuration
- ϕ PowerCenter Client
- ϕ Repository Service
- ϕ Integration Service

Introduction

PowerCenter provides an environment that allows you to load data into a centralized location, such as a data warehouse or operational data store (ODS). You can extract data from multiple sources, transform the data according to business logic you build in the client application, and load the transformed data into file and relational targets.

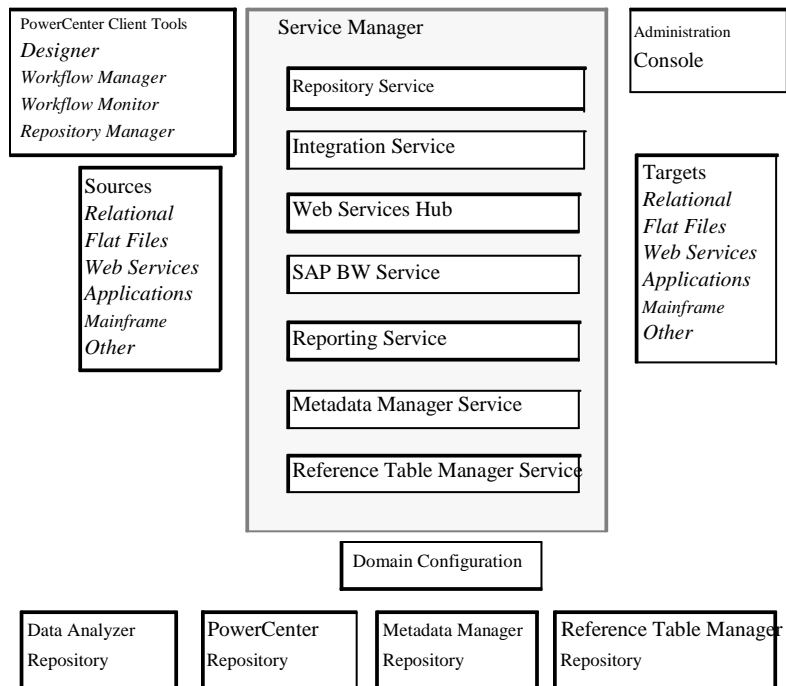
PowerCenter also provides the ability to view and analyze business information and browse and analyze metadata from disparate metadata repositories.

PowerCenter includes the following components:

- ϕ **PowerCenter domain.** The Power Center domain is the primary unit for management and administration within PowerCenter. The Service Manager runs on a PowerCenter domain. The Service Manager supports the domain and the application services. Application services represent server-based functionality and include the Repository Service, Integration Service, Web Services Hub, and SAP BW Service.
- ϕ **PowerCenter repository.** The PowerCenter repository resides in a relational database. The repository database tables contain the instructions required to extract, transform, and load data
- ϕ **Administration Console.** The Administration Console is a web application that you use to administer the PowerCenter domain and PowerCenter security. For more information, see “Administration Console” on page 6.
- ϕ **Domain configuration.** The domain configuration is a set of relational database tables that stores the configuration information for the domain. The Service Manager on the master gateway node manages the domain configuration. The domain configuration is accessible to all gateway nodes in the domain.
- ϕ **PowerCenter Client.** The PowerCenter Client is an application used to define sources and targets, build mappings and mapplets with the transformation logic, and create workflows to run the mapping logic. The PowerCenter Client connects to the repository through the Repository Service to modify repository metadata. It connects to the Integration Service to start workflows.
- ϕ **Repository Service.** The Repository Service accepts requests from the PowerCenter Client to create and modify repository metadata and accepts requests from the Integration Service for metadata when a workflow runs.

- φ **Integration Service.** The Integration Service extracts data from sources and loads data to targets.
- φ **Web Services Hub.** Web Services Hub is a gateway that exposes PowerCenter functionality to external clients through web services.
- φ **SAP BW Service.** The SAP BW Service extracts data from and loads data to SAP NetWeaver BI. If you use PowerExchange for SAP NetWeaver BI, you must create and enable an SAP BW Service in the PowerCenter domain. For more information, see the *PowerCenter Administrator Guide* and the *PowerExchange for SAP NetWeaver User Guide*.
- φ **Reporting Service.** The Reporting Service runs the Data Analyzer web application. Data Analyzer provides a framework for creating and running custom reports and dashboards. You can use Data Analyzer to run the metadata reports provided with PowerCenter, including the PowerCenter Repository Reports and Data Profiling Reports. Data Analyzer stores the data source schemas and report metadata in the Data Analyzer repository.
- φ **Metadata Manager Service.** The Metadata Manager Service runs the Metadata Manager web application. You can use Metadata Manager to browse and analyze metadata from disparate metadata repositories. Metadata Manager helps you understand and manage how information and processes are derived, how they are related, and how they are used. Metadata Manager stores information about the metadata to be analyzed in the Metadata Manager repository.
- φ **Reference Table Manager Service.** The Reference Table Manager Service runs the Reference Table Manager web application. Use Reference Table Manager to manage reference data such as valid, default, and cross-reference values. Reference Table Manager stores reference tables metadata and the users and connection information in the Reference Table Manager repository. The reference tables are stored in a staging area.

The following figure shows the PowerCenter components:



Sources

PowerCenter accesses the following sources:

- φ **Relational.** Oracle, Sybase ASE, Informix, IBM DB2, Microsoft SQL Server, and Teradata.
- φ **File.** Fixed and delimited flat file, COBOL file, XML file, and web log.
- φ **Application.** You can purchase additional PowerExchange products to access business sources such as Hyperion Essbase, WebSphere MQ, IBM DB2 OLAP Server, JMS, Microsoft Message Queue, PeopleSoft, SAP NetWeaver, SAS, Siebel, TIBCO, and webMethods.
- φ **Mainframe.** You can purchase PowerExchange to access source data from mainframe databases such as Adabas, Datacom, IBM DB2 OS/390, IBM DB2 OS/400, IDMS, IDMS-X, IMS, and VSAM.
- φ **Other.** Microsoft Excel, Microsoft Access, and external web services.

Targets

PowerCenter can load data into the following targets:

- φ **Relational.** Oracle, Sybase ASE, Sybase IQ, Informix, IBM DB2, Microsoft SQL Server, and Teradata.
- φ **File.** Fixed and delimited flat file and XML.
- φ **Application.** You can purchase additional PowerExchange products to load data into business sources such as Hyperion Essbase, WebSphere MQ, IBM DB2 OLAP Server, JMS, Microsoft Message Queue, PeopleSoft EPM, SAP NetWeaver, SAP NetWeaver BI, SAS, Siebel, TIBCO, and webMethods.
- φ **Mainframe.** You can purchase PowerExchange to load data into mainframe databases such as IBM DB2 for z/OS, IMS, and VSAM.
- φ **Other.** Microsoft Access and external web services.

You can load data into targets using ODBC or native drivers, FTP, or external loaders.

PowerCenter Domain

PowerCenter has a service-oriented architecture that provides the ability to scale services and share resources across multiple machines. PowerCenter provides the PowerCenter domain to support the administration of the PowerCenter services. A domain is the primary unit for management and administration of services in PowerCenter.

A domain contains the following components:

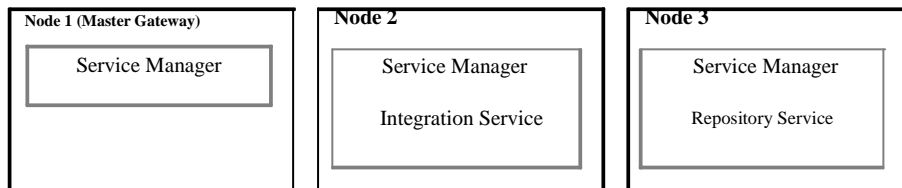
- ϕ **One or more nodes.** A node is the logical representation of a machine in a domain. A domain may contain more than one node. The node that hosts the domain is the master gateway for the domain. You can add other machines as nodes in the domain and configure the nodes to run application services such as the Integration Service or Repository Service. All service requests from other nodes in the domain go through the master gateway.
A nodes runs service processes, which is the runtime representation of an application service running on a node.
- ϕ **Service Manager.** The Service Manager is built into the domain to support the domain and the application services. The Service Manager runs on each node in the domain. The Service Manager starts and runs the application services on a machine.
- ϕ **Application services.** A group of services that represent PowerCenter server-based functionality. The application services that run on each node in the domain depend on the way you configure the node and the application service.

You use the PowerCenter Administration Console to manage the domain.

A domain can be a mixed-version domain or a single-version domain. In a mixed-version domain, you can run multiple versions of services. In a single-version domain, you can run one version of services.

If you have the high availability option, you can scale services and eliminate single points of failure for services. The Service Manager and application services can continue running despite temporary network or hardware failures. High availability includes resilience, failover, and recovery for services and tasks in a domain.

The following figure shows a sample domain with three nodes:



This domain has a master gateway on Node 1. Node 2 runs an Integration Service, and Node 3 runs the Repository Service.

RELATED TOPICS:

- ϕ “Administration Console”

Service Manager

The Service Manager is built in to the domain and supports the domain and the application services. The Service Manager performs the following functions:

- ϕ **Alerts.** Provides notifications about domain and service events.
- ϕ **Authentication.** Authenticates user requests from the Administration Console, PowerCenter Client, Metadata Manager, and Data Analyzer.
- ϕ **Authorization.** Authorizes user requests for domain objects. Requests can come from the Administration Console or from *infacmd*.
- ϕ **Domain configuration.** Manages domain configuration metadata.

- φ **Node configuration.** Manages node configuration metadata.
- φ **Licensing.** Registers license information and verifies license information when you run application services.
- φ **Logging.** Provides accumulated log events from each service in the domain. You can view logs in the Administration Console and Workflow Monitor.
- φ **User management.** Manages users, groups, roles, and privileges.

Application Services

When you install PowerCenter Services, the installation program installs the following application services:

- φ **Repository Service.** Manages connections to the PowerCenter repository.
- φ **Integration Service.** Runs sessions and workflows. For more information, see “Integration Service” on page 14.
- φ **Web Services Hub.** Exposes PowerCenter functionality to external clients through web services.
- φ **SAP BW Service.** Listens for RFC requests from SAP NetWeaver BI and initiates workflows to extract from or load to SAP NetWeaver BI.
- φ **Reporting Service.** Runs the Data Analyzer application.
- φ **Metadata Manager Service.** Runs the Metadata Manager application.
- φ **Reference Table Manager Service.** Runs the Reference Table Manager application.

PowerCenter Repository

The PowerCenter repository resides in a relational database. The repository stores information required to extract, transform, and load data. It also stores administrative information such as permissions and privileges for users and groups that have access to the repository. PowerCenter applications access the PowerCenter repository through the Repository Service.

You administer the repository through the PowerCenter Administration Console and command line programs.

You can develop global and local repositories to share metadata:

- φ **Global repository.** The global repository is the hub of the repository domain. Use the global repository to store common objects that multiple developers can use through shortcuts. These objects may include operational or application source definitions, reusable transformations, mapplets, and mappings.
- φ **Local repositories.** A local repository is any repository within the domain that is not the global repository. Use local repositories for development. From a local repository, you can create shortcuts to objects in shared folders in the global repository. These objects include source definitions, common dimensions and lookups, and enterprise standard transformations. You can also create copies of objects in non-shared folders.

PowerCenter supports versioned repositories. A versioned repository can store multiple versions of an object. PowerCenter version control allows you to efficiently develop, test, and deploy metadata into production.

You can view repository metadata in the Repository Manager. Informatica Metadata Exchange (MX) provides a set of relational views that allow easy SQL access to the PowerCenter metadata repository.

You can also create a Reporting Service in the Administration Console and run the PowerCenter Repository Reports to view repository metadata.

Administration Console

The Administration Console is a web application that you use to administer the PowerCenter domain and PowerCenter security.

Domain Page

You administer the PowerCenter domain on the Domain page of the Administration Console. Domain objects include services, nodes, and licenses.

You can complete the following tasks in the Domain page:

- φ **Manage application services.** Manage all application services in the domain, such as the Integration Service and Repository Service.
- φ **Configure nodes.** Configure node properties, such as the backup directory and resources. You can also shut down and restart nodes.
- φ **Manage domain objects.** Create and manage objects such as services, nodes, licenses, and folders. Folders allow you to organize domain objects and manage security by setting permissions for domain objects.
- φ **View and edit domain object properties.** View and edit properties for all objects in the domain, including the domain object.
- φ **View log events.** Use the Log Viewer to view domain, Integration Service, SAP BW Service, Web Services Hub, and Repository Service log events.
- φ **Generate and upload node diagnostics.** You can generate and upload node diagnostics to the Configuration Support Manager. In the Configuration Support Manager, you can diagnose issues in your Informatica environment and maintain details of your configuration.

Other domain management tasks include applying licenses and managing grids and resources.

The following figure shows the Domain page:



Security Page

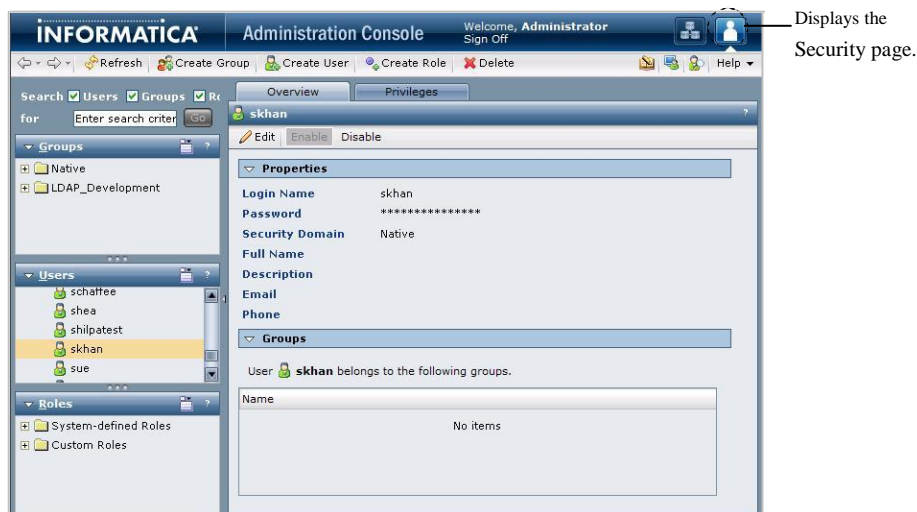
You administer PowerCenter security on the Security page of the Administration Console. You manage users and groups that can log in to the following PowerCenter applications:

- φ Administration Console
- φ PowerCenter Client
- φ Metadata Manager
- φ Data Analyzer

You can complete the following tasks in the Security page:

- φ **Manage native users and groups.** Create, edit, and delete native users and groups.
- φ **Configure LDAP authentication and import LDAP users and groups.** Configure a connection to an LDAP directory service. Import users and groups from the LDAP directory service.
- φ **Manage roles.** Create, edit, and delete roles. Roles are collections of privileges. Privileges determine the actions that users can perform in PowerCenter applications.
- φ **Assign roles and privileges to users and groups.** Assign roles and privileges to users and groups for the domain, Repository Service, Metadata Manager Service, Reporting Service, or Reference Table Manager Service.
- φ **Manage operating system profiles.** Create, edit, and delete operating system profiles. An operating system profile is a level of security that the Integration Services uses to run workflows. The operating system profile contains the operating system user name, service process variables, and environment variables. You can configure the Integration Service to use operating system profiles to run workflows.

The following figure shows the Security page:



Domain Configuration

Configuration information for a PowerCenter domain is stored in a set of relational database tables managed by the Service manager and accessible to all gateway nodes in the domain. The domain configuration database stores the following types of information about the domain:

- φ **Domain configuration.** Domain metadata such as host names and port numbers of nodes in the domain. The domain configuration database also stores information on the master gateway node and all other nodes in the domain.
- φ **Usage.** Includes CPU usage for each application service and the number of Repository Services running in the domain.
- φ **Users and groups.** Information on the native and LDAP users and the relationships between users and groups.
- φ **Privileges and roles.** Information on the privileges and roles assigned to users and groups in the domain.

Each time you make a change to the domain, the Service Manager updates the domain configuration database. For example, when you add a node to the domain, the Service Manager adds the node information to the domain configuration. All gateway nodes connect to the domain configuration database to retrieve domain information and update the domain configuration.

PowerCenter Client

The PowerCenter Client application consists of the following tools that you use to manage the repository, design mappings, mapplets, and create sessions to load the data:

- φ **Designer.** Use the Designer to create mappings that contain transformation instructions for the Integration Service.
- φ **Mapping Architect for Visio.** Use the Mapping Architect for Visio to create mapping templates that can be used to generate multiple mappings.
- φ **Repository Manager.** Use the Repository Manager to assign permissions to users and groups and manage folders.
- φ **Workflow Manager.** Use the Workflow Manager to create, schedule, and run workflows. A workflow is a set of instructions that describes how and when to run tasks related to extracting, transforming, and loading data.
- φ **Workflow Monitor.** Use the Workflow Monitor to monitor scheduled and running workflows for each Integration Service

Install the client application on a Microsoft Windows machine.

PowerCenter Designer

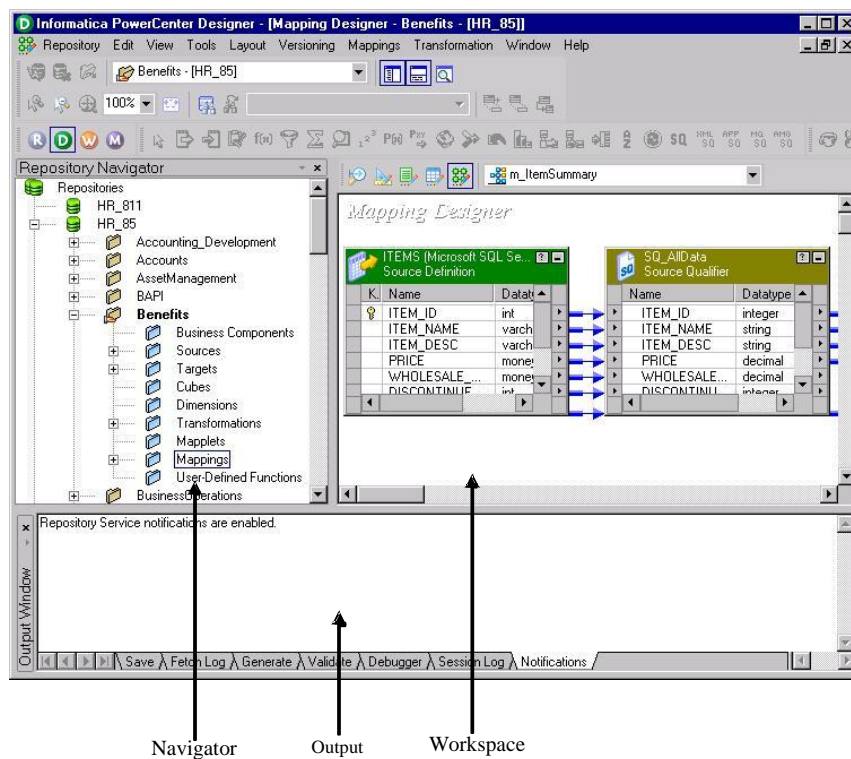
The Designer has the following tools that you use to analyze sources, design target schemas, and build source-to-target mappings:

- φ **Source Analyzer.** Import or create source definitions.
- φ **Target Designer.** Import or create target definitions.
- φ **Transformation Developer.** Develop transformations to use in mappings. You can also develop user-defined functions to use in expressions.
- φ **Mapplet Designer.** Create sets of transformations to use in mappings.
- φ **Mapping Designer.** Create mappings that the Integration Service uses to extract, transform, and load data.

You can display the following windows in the Designer:

- φ **Navigator.** Connect to repositories and open folders within the Navigator. You can also copy objects and create shortcuts within the Navigator.
- φ **Workspace.** Open different tools in this window to create and edit repository objects, such as sources, targets, mapplets, transformations, and mappings.
- φ **Output.** View details about tasks you perform, such as saving your work or validating a mapping.

The following figure shows the default Designer interface:

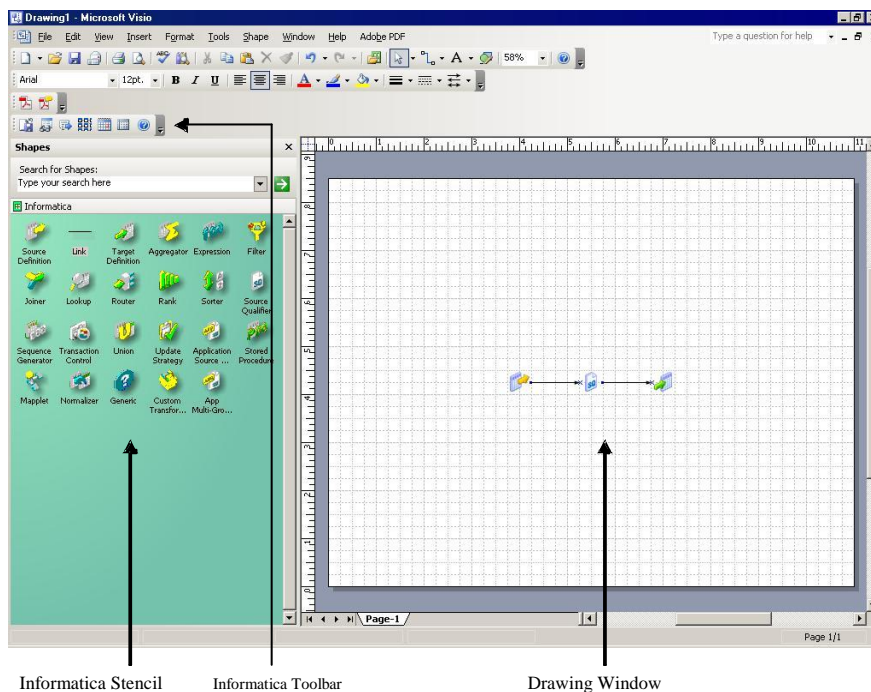


Mapping Architect for Visio

Use Mapping Architect for Visio to create mapping templates using Microsoft Office Visio. When you work with a mapping template, you use the following main areas:

- φ **Informatica stencil.** Displays shapes that represent PowerCenter mapping objects. Drag a shape from the Informatica stencil to the drawing window to add a mapping object to a mapping template.
- φ **Informatica toolbar.** Displays buttons for tasks you can perform on a mapping template. Contains the online help button.
- φ **Drawing window.** Work area for the mapping template. Drag shapes from the Informatica stencil to the drawing window and set up links between the shapes. Set the properties for the mapping objects and the rules for data movement and transformation.

The following figure shows the Mapping Architect for Visio interface:



Repository Manager

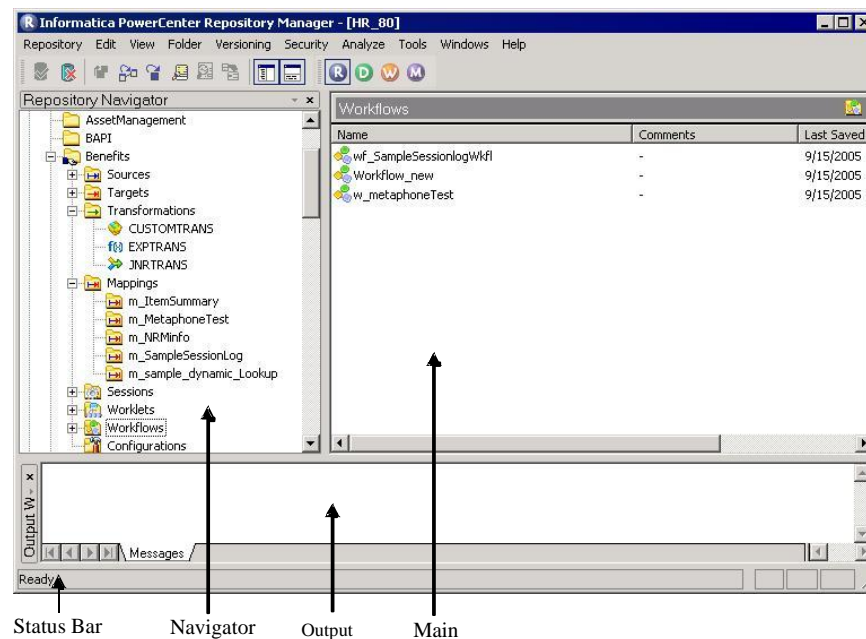
Use the Repository Manager to administer repositories. You can navigate through multiple folders and repositories, and complete the following tasks:

- φ **Manage user and group permissions.** Assign and revoke folder and global object permissions.
- φ **Perform folder functions.** Create, edit, copy, and delete folders. Work you perform in the Designer and Workflow Manager is stored in folders. If you want to share metadata, you can configure a folder to be shared.
- φ **View metadata.** Analyze sources, targets, mappings, and shortcut dependencies, search by keyword, and view the properties of repository objects.

The Repository Manager can display the following windows:

- φ **Navigator.** Displays all objects that you create in the Repository Manager, the Designer, and the Workflow Manager. It is organized first by repository and by folder.
- φ **Main.** Provides properties of the object selected in the Navigator. The columns in this window change depending on the object selected in the Navigator.
- φ **Output.** Provides the output of tasks executed within the Repository Manager.

The following figure shows the Repository Manager interface:



Repository Objects

You create repository objects using the Designer and Workflow Manager client tools. You can view the following objects in the Navigator window of the Repository Manager:

- φ **Source definitions.** Definitions of database objects such as tables, views, synonyms, or files that provide source data.
- φ **Target definitions.** Definitions of database objects or files that contain the target data.
- φ **Mappings.** A set of source and target definitions along with transformations containing business logic that you build into the transformation. These are the instructions that the Integration Service uses to transform and move data.
- φ **Reusable transformations.** Transformations that you use in multiple mappings.
- φ **Mapplets.** A set of transformations that you use in multiple mappings.
- φ **Sessions and workflows.** Sessions and workflows store information about how and when the Integration Service moves data. A workflow is a set of instructions that describes how and when to run tasks related to extracting, transforming, and loading data. A session is a type of task that you can put in a workflow. Each session corresponds to a single mapping.

Workflow Manager

In the Workflow Manager, you define a set of instructions to execute tasks such as sessions, emails, and shell commands. This set of instructions is called a workflow.

The Workflow Manager has the following tools to help you develop a workflow:

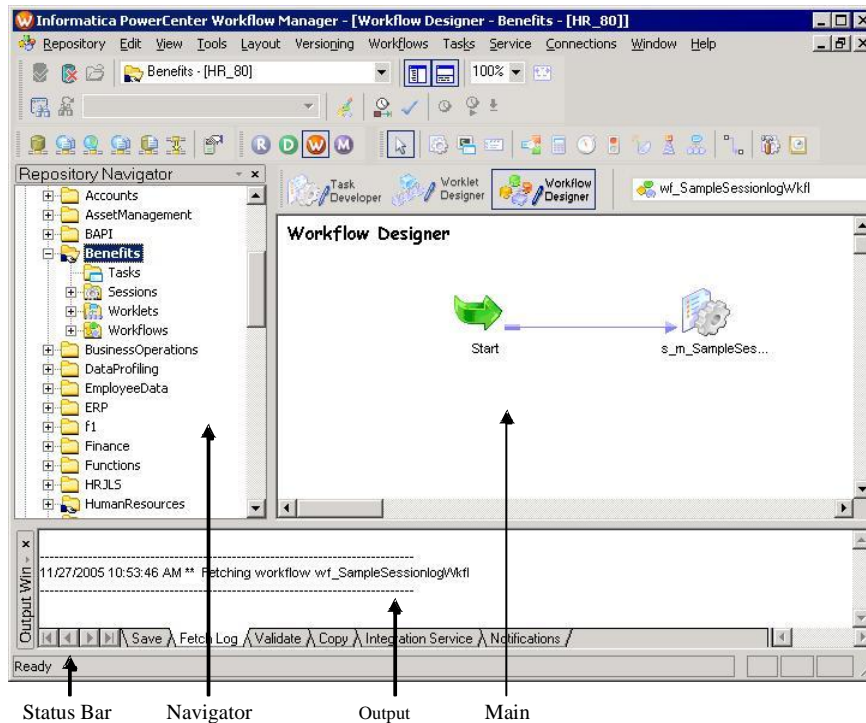
- φ **Task Developer.** Create tasks you want to accomplish in the workflow.
- φ **Worklet Designer.** Create a worklet in the Worklet Designer. A worklet is an object that groups a set of tasks. A worklet is similar to a workflow, but without scheduling information. You can nest worklets inside a workflow.
- φ **Workflow Designer.** Create a workflow by connecting tasks with links in the Workflow Designer. You can also create tasks in the Workflow Designer as you develop the workflow.

When you create a workflow in the Workflow Designer, you add tasks to the workflow. The Workflow Manager includes tasks, such as the Session task, the Command task, and the Email task so you can design a workflow. The Session task is based on a mapping you build in the Designer.

You then connect tasks with links to specify the order of execution for the tasks you created. Use conditional links and workflow variables to create branches in the workflow.

When the workflow start time arrives, the Integration Service retrieves the metadata from the repository to execute the tasks in the workflow. You can monitor the workflow status in the Workflow Monitor.

The following figure shows the Workflow Manager interface:



Workflow Monitor

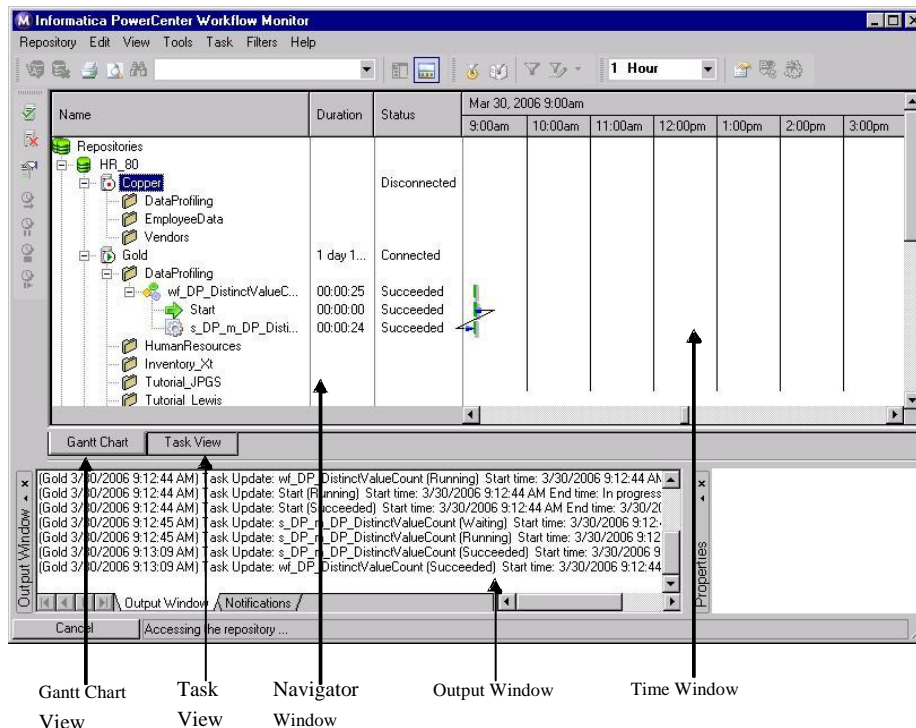
You can monitor workflows and tasks in the Workflow Monitor. You can view details about a workflow or task in Gantt Chart view or Task view. You can run, stop, abort, and resume workflows from the Workflow Monitor. You can view sessions and workflow log events in the Workflow Monitor Log Viewer.

The Workflow Monitor displays workflows that have run at least once. The Workflow Monitor continuously receives information from the Integration Service and Repository Service. It also fetches information from the repository to display historic information.

The Workflow Monitor consists of the following windows:

- φ **Navigators window.** Displays monitored repositories, servers, and repositories objects.
- φ **Output window.** Displays messages from the Integration Service and Repository Service.
- φ **Time window.** Displays progress of workflow runs.
- φ **Gantt Chart view.** Displays details about workflow runs in chronological format.
- φ **Task view.** Displays details about workflow runs in a report format.

The following figure shows the Workflow Monitor interface:



Repository Service

The Repository Service manages connections to the PowerCenter repository from repository clients. A repository client is any PowerCenter component that connects to the repository. The Repository Service is a separate, multi-threaded process that retrieves, inserts, and updates metadata in the repository database tables. The Repository Service ensures the consistency of metadata in the repository.

The Repository Service accepts connection requests from the following PowerCenter components:

- φ **PowerCenter Client.** Use the Designer and Workflow Manager to create and store mapping metadata and connection object information in the repository. Use the Workflow Monitor to retrieve workflow run status information and session logs written by the Integration Service. Use the Repository Manager to organize and secure metadata by creating folders and assigning permissions to users and groups.
- φ **Command line programs.** Use command line programs to perform repository metadata administration tasks and service-related functions.
- φ **Integration Service.** When you start the Integration Service, it connects to the repository to schedule workflows. When you run a workflow, the Integration Service retrieves workflow task and mapping metadata from the repository. The Integration Service writes workflow status to the repository.
- φ **Web Services Hub.** When you start the Web Services Hub, it connects to the repository to access web-enabled workflows. The Web Services Hub retrieves workflow task and mapping metadata from the repository and writes workflow status to the repository.
- φ **SAP BW Service.** Listens for RFC requests from SAP NetWeaver BI and initiates workflows to extract from or load to SAP NetWeaver BI.

You install the Repository Service when you install PowerCenter Services. After you install the PowerCenter Services, you can use the Administration Console to manage the Repository Service.

Integration Service

The Integration Service reads workflow information from the repository. The Integration Service connects to the repository through the Repository Service to fetch metadata from the repository.

A workflow is a set of instructions that describes how and when to run tasks related to extracting, transforming, and loading data. The Integration Service runs workflow tasks. A session is a type of workflow task. A session is a set of instructions that describes how to move data from sources to targets using a mapping.

A session extracts data from the mapping sources and stores the data in memory while it applies the transformation rules that you configure in the mapping. The Integration Service loads the transformed data into the mapping targets.

Other workflow tasks include commands, decisions, timers, pre-session SQL commands, post-session SQL commands, and email notification.

The Integration Service can combine data from different platforms and source types. For example, you can join data from a flat file and an Oracle source. The Integration Service can also load data to different platforms and target types.

You install the Integration Service when you install PowerCenter Services. After you install the PowerCenter Services, you can use the Administration Console to manage the Integration Service.

CHAPTER 2

Using the Designer

This chapter includes the following topics:

- φ Overview
- φ Configuring Designer Options

Overview

The Designer has tools to help you build mappings and maplets so you can specify how to move and transform data between sources and targets. The Designer helps you create source definitions, target definitions, and transformations to build the mappings.

The Designer lets you work with multiple tools at one time and to work in multiple folders and repositories at the same time. It also includes windows so you can view folders, repository objects, and tasks.

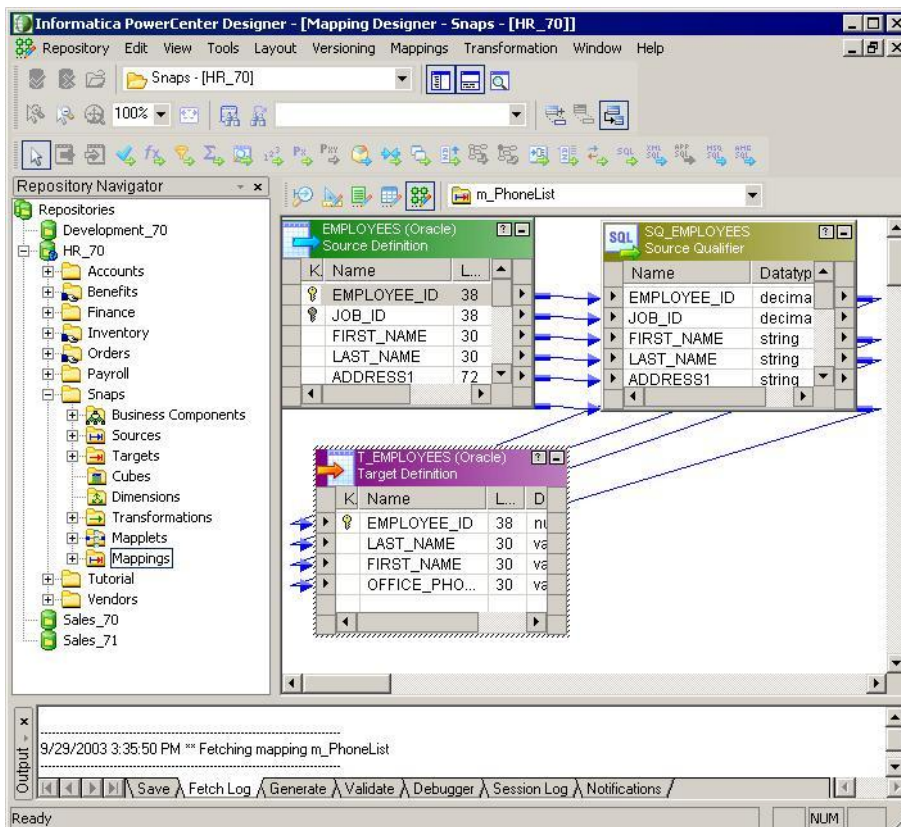
You can configure general Designer settings, such as background color and font. You can also configure specific tool settings for each Designer tool.

Designer Tools

The Designer provides the following tools:

- φ **Source Analyzer.** Use to import or create source definitions for flat file, XML, COBOL, Application, and relational sources.
- φ **Target Designer.** Use to import or create target definitions.
- φ **Transformation Developer.** Use to create reusable transformations.
- φ **Maplet Designer.** Use to create maplets.
- φ **Mapping Designer.** Use to create mappings.

Figure 1-1 shows the Designer windows:



Designer Windows

The Designer consists of the following windows:

- φ **Navigator.** Use to connect to and work in multiple repositories and folders. You can also copy and delete objects and create shortcuts using the Navigator.
- φ **Workspace.** Use to view or edit sources, targets, mapplets, transformations, and mappings. You work with a single tool at a time in the workspace, which has two formats: default and workbook. You can view multiple versions of an object in the workspace.
- φ **Status bar.** Displays the status of the operation you perform.
- φ **Output.** Provides details when you perform certain tasks, such as saving work or validating a mapping. Right-click the Output window to access window options, such as printing output text, saving text to file, and changing the font size.
- φ **Overview.** An optional window to simplify viewing workbooks containing large mappings or a large number of objects. Outlines the visible area in the workspace and highlights selected objects in color. To open the Overview window, click View > Overview Window.
- φ **Instance Data.** View transformation data while you run the Debugger to debug a mapping.
- φ **Target Data.** View target data while you run the Debugger to debug a mapping.

You can view a list of open windows and switch from one window to another in the Designer. To view the list of open windows, click Window > Windows.

CHAPTER 3

Working with Sources

This chapter includes the following topics:

- φ Overview
- φ Working with Relational Sources

Overview

To extract data from a source, first define sources in the repository. You can import or create the following types of source definitions in the Source Analyzer:

- φ Relational tables, views, and synonyms
- φ Fixed-width and delimited flat files that do not contain binary data.
- φ COBOL files
- φ XML files
- φ Web Services Description Language (WSDL)
- φ Data models using certain data modeling tools through Metadata Exchange for Data Models (an add-on product)

Note: Because source definitions must match the source, you should import definitions instead of creating them manually.

PowerCenter lets you work with sources that use multibyte character sets. Source code pages must be a superset of the target code pages.

Updating Source Definitions

When you update a source definition, the Designer propagates the changes to all mappings using that source. Some changes to source definitions can invalidate mappings.

Creating Sessions

When you create a session, you can specify a source location different from the location you use when you import the source definition. If the source is a file, you can override some of the file properties when you create a session.

Working with Relational Sources

You can add and maintain relational source definitions in the following ways:

- φ **Import source definitions.** Import source definitions into the Source Analyzer.
- φ **Update source definitions.** Update source definitions either manually or by reimporting the definition.

Importing a Relational Source Definition

You can import relational source definitions from database tables, views, and synonyms. When you import a source definition, you import the following source metadata:

- φ Source name
- φ Database location
- φ Column names
- φ Datatypes
- φ Key constraints

To import a relational source definition:

1. In the Source Analyzer, click Sources > Import from Database.
2. Select the ODBC data source used to connect to the source database.

If you need to create or modify an ODBC data source, click the Browse button to open the ODBC Administrator. Create the data source, and click OK. Select the new ODBC data source.
3. Enter a database user name and password to connect to the database.

Note: The user name must have the appropriate database permissions to view the object.

You may need to specify the owner name for database objects you want to use as sources.
4. Click Connect.

If no table names appear, or if the table you want to import does not appear, click All.
5. Scroll down through the list of sources to find the source you want to import. Select the relational object or objects you want to import.

You can hold down the Shift key to select a block of sources within one folder or hold down the Ctrl key to make non-consecutive selections within a folder. You can also select all tables within a folder by selecting the folder and clicking Select All. Use the Select None button to clear all highlighted selections.
6. Click OK.

The source definition appears in the Source Analyzer. In the Navigator, the new source definition appears in the Sources node of the active repository folder, under the source database name.

Editing Relational Source Definitions

You might want to manually edit a source definition to record properties that you cannot import from the source. You can edit a relational source definition to create key columns and key relationships. These relationships can be logical relationships. They do not have to exist in the database.

You can add descriptions or specify links to business documentation for source definitions at any time. Adding descriptions or business documentation links to source definitions is an easy way to document the purpose of a source definition. You can add or modify descriptions to any existing source definition.

CHAPTER 4

Working with Targets

This chapter includes the following topics:

- φ Overview
- φ Importing a Target Definition
- φ Creating a Target Definition from a Source Definition
- φ Creating a Target Definition from a Transformation
- φ Manually Creating a Target Definition

Overview

Before you create a mapping, you must define targets in the repository. Use the Target Designer to import and design target definitions or to create and maintain target definitions. Target definitions include properties such as column names and data types.

Creating Target Definitions

You can create the following types of target definitions in the Target Designer:

- φ **Relational.** Create a relational target for a particular database platform. Create a relational target definition when you want to use an external loader to the target database.
- φ **Flat file.** Create fixed-width and delimited flat file target definitions.

Importing a Target Definition

You can import the following target definitions:

- φ **Flat file.** The Target Designer uses the Flat File Wizard to import a target definition from a flat file that matches the structure of the flat file.
- φ **Relational table.** You can import a relational table to create a target definition that matches the structure of the relational table.
- φ **XML file.** You can import an XML target definition from an XML, DTD, or XML schema file.

Use a target definition in a mapping after you have added it to the repository.

Importing a Relational Target Definition

When you import a target definition from a relational table, the Designer imports the following target details:

- φ **Target name.** The name of the target.
- φ **Database location.** You specify the database location when you import a relational source. You can specify a different location when you edit the target definition in the Target Designer and when you configure a session.
- φ **Column names.** The names of the columns.
- φ **Datatypes.** The Designer imports the native datatype for each column.
- φ **Key constraints.** The constraints in the target definition can be critical, since they may prevent you from moving data into the target if the Integration Service violates a constraint during a workflow. For example, if a column contains the NOT NULL constraint and you do not map data to this column, the Integration Service cannot insert new records into the target table.
- φ **Key relationships.** You can customize the Target Designer to create primary key-foreign key relationships. Click Tools > Options and select the Format tab. Select Import Primary and Foreign Keys.

You can also create logical relationships in the repository. Key relationships do not have to exist in the

database.

When you import target definitions, the Designer does not import the target indexes. You can change this default setting in powrmart.ini. You can find powrmart.ini in the root directory of the PowerCenter Client installation.

To import target definition indexes, add the following text to powrmart.ini [Main] section:

```
ImportIndexes=Yes
```

To import a relational target definition, you must be able to connect to the database from the client machine using a properly configured ODBC data source or gateway. You may also require read permission on the database object.

Note: Because views can include columns from more than one table, the Integration Service might encounter database errors when trying to insert, update, or delete data. If you import a target view, make sure that it is a view of a single table.

To import a relational target definition:

1. In the Target Designer, click Targets > Import from Database.
2. Select the ODBC data source used to connect to the target database.

If you need to create or modify an ODBC data source first, click the Browse button to open the ODBC Administrator. After creating or modifying the ODBC source, continue with the following steps.

3. Enter the user name and password needed to open a connection to the database, and click Connect.

If you are not the owner of the table you want to use as a target, specify the owner name.

4. Drill down through the list of database objects to view the available tables as targets.
5. Select the relational table or tables to import the definitions into the repository.

You can hold down the Shift key to select a block of tables, or hold down the Ctrl key to make non-contiguous selections. You can also use the Select All and Select None buttons to select or clear all available targets.

6. Click OK.

The selected target definitions now appear in the Navigator under the Targets icon.

Manually Creating a Target Definition

You can manually create a target definition rather than importing it or creating it from a source definition.

To manually create a target definition:

1. In the Target Designer, click Targets > Create.
2. Enter a name for the target and select the target type.
If you create a relational definition, follow database-specific naming conventions.
3. Click Create.
4. An empty definition appears in the workspace. It may be covered by the dialog box. The new target definition also appears within the Navigator window.
4. If you want to create another target definition, enter a new target name and target type and click Create. Repeat this step for each target you want to create.
5. Click Done when you finish creating target definitions.
6. Configure the target definition.

Note: You cannot manually create a target definition for XML files.

CHAPTER 5

Working with Transformations

This chapter includes the following topics:

- φ Overview
- φ Creating a Transformation

Overview

A transformation is a repository object that generates, modifies, or passes data. The Designer provides a set of transformations that perform specific functions. For example, an Aggregator transformation performs calculations on groups of data.

Transformations in a mapping represent the operations the Integration Service performs on the data. Data passes through transformation ports that you link in a mapping or mapplet.

Transformations can be active or passive. An active transformation can change the number of rows that pass through it, such as a Filter transformation that removes rows that do not meet the filter condition. A passive transformation does not change the number of rows that pass through it, such as an Expression transformation that performs a calculation on data and passes all rows through the transformation.

Transformations can be connected to the data flow, or they can be unconnected. An unconnected transformation is not connected to other transformations in the mapping. An unconnected transformation is called within another transformation, and returns a value to that transformation.

Table 1-1 provides a brief description of each transformation:

Table 1-1. Transformation Descriptions

Transformation	Type	Description
Aggregator	Active/ Connected	Performs aggregate calculations.
Application Source Qualifier	Active/ Connected	Represents the rows that the Integration Service reads from an application, such as an ERP source, when it runs a session.
Custom	Active or Passive/ Connected	Calls a procedure in a shared library or DLL.
Expression	Passive/ Connected	Calculates a value.
External Procedure	Passive/ Connected or Unconnected	Calls a procedure in a shared library or in the COM layer of Windows.
Filter	Active/ Connected	Filters data.
HTTP	Passive/ Connected	Connects to an HTTP server to read or update data.
Input	Passive/ Connected	Defines maplet input rows. Available in the Maplet Designer.
Java	Active or Passive/ Connected	Executes user logic coded in Java. The byte code for the user logic is stored in the repository.
Joiner	Active/ Connected	Joins data from different databases or flat file systems.
Lookup	Passive/ Connected or Unconnected	Looks up values.
Normalizer	Active/ Connected	Source qualifier for COBOL sources. Can also use in the pipeline to normalize data from relational or flat file sources.
Output	Passive/ Connected	Defines maplet output rows. Available in the Maplet Designer.
Rank	Active/ Connected	Limits records to a top or bottom range.
Router	Active/ Connected	Routes data into multiple transformations based on group conditions.
Sequence Generator	Passive/ Connected	Generates primary keys.
Sorter	Active/ Connected	Sorts data based on a sort key.
Source Qualifier	Active/ Connected	Represents the rows that the Integration Service reads from a relational or flat file source when it runs a session.
SQL	Active or Passive/ Connected	Executes SQL queries against a database.
Stored Procedure	Passive/ Connected or Unconnected	Calls a stored procedure.

Table 1-1. Transformation Descriptions

Transformation	Type	Description
Transaction Control	Active/ Connected	Defines commit and rollback transactions.
Union	Active/ Connected	Merges data from different databases or flat file systems.
Unstructured Data	Active or Passive/ Connected	Transforms data in unstructured and semi-structured formats.
Update Strategy	Active/ Connected	Determines whether to insert, delete, update, or reject rows.
XML Generator	Active/ Connected	Reads data from one or more input ports and outputs XML through a single output port.
XML Parser	Active/ Connected	Reads XML from one input port and outputs data to one or more output ports.
XML Source Qualifier	Active/ Connected	Represents the rows that the Integration Service reads from an XML source when it runs a session.

When you build a mapping, you add transformations and configure them to handle data according to a business purpose. Complete the following tasks to incorporate a transformation into a mapping:

1. **Create the transformation.** Create it in the Mapping Designer as part of a mapping, in the Mapplet Designer as part of a mapplet, or in the Transformation Developer as a reusable transformation.
2. **Configure the transformation.** Each type of transformation has a unique set of options that you can configure.
3. **Link the transformation to other transformations and target definitions.** Drag one port to another to link them in the mapping or mapplet.

Creating a Transformation

You can create transformations using the following Designer tools:

- φ **Mapping Designer.** Create transformations that connect sources to targets. Transformations in a mapping cannot be used in other mappings unless you configure them to be reusable.
- φ **Transformation Developer.** Create individual transformations, called reusable transformations, that use in multiple mappings.
- φ **Mapplet Designer.** Create and configure a set of transformations, called mapplets, that you use in multiple mappings.

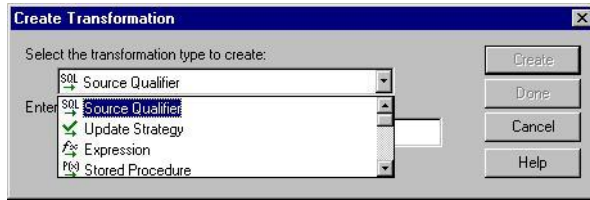
Use the same process to create a transformation in the Mapping Designer, Transformation Developer, and Mapplet Designer.

To create a transformation:

1. Open the appropriate Designer tool.
2. In the Mapping Designer, open or create a Mapping. In the Mapplet Designer, open or create a Mapplet.
3. On the Transformations toolbar, click the button corresponding to the transformation you want to create.

-or-

Click Transformation > Create and select the type of transformation you want to create.



4. Drag across the portion of the mapping where you want to place the transformation.

The new transformation appears in the workspace. Next, you need to configure the transformation by adding any new ports to it and setting other properties.

CHAPTER 6

Aggregator Transformation

This chapter includes the following topics:

- φ Overview
- φ Components of the Aggregator Transformation
- φ Configuring Aggregate Caches
- φ Aggregate Expressions
- φ Group By Ports
- φ Using Sorted Input
- φ Creating an Aggregator Transformation

Overview

Transformation type:

Active
Connected

The Aggregator transformation performs aggregate calculations, such as averages and sums. The Integration Service performs aggregate calculations as it reads and stores data group and row data in an aggregate cache. The Aggregator transformation is unlike the Expression transformation, in that you use the Aggregator transformation to perform calculations on groups. The Expression transformation permits you to perform calculations on a row-by-row basis only.

When you use the transformation language to create aggregate expressions, you can use conditional clauses to filter rows, providing more flexibility than SQL language.

After you create a session that includes an Aggregator transformation, you can enable the session option, Incremental Aggregation. When the Integration Service performs incremental aggregation, it passes new source data through the mapping and uses historical cache data to perform new aggregation calculations incrementally.

Components of the Aggregator Transformation

The Aggregator is an active transformation, changing the number of rows in the pipeline. The Aggregator transformation has the following components and options:

- ϕ **Aggregate cache.** The Integration Service stores data in the aggregate cache until it completes aggregate calculations. It stores group values in an index cache and row data in the data cache.
- ϕ **Aggregate expression.** Enter an expression in an output port. The expression can include non-aggregate expressions and conditional clauses.
- ϕ **Group by port.** Indicate how to create groups. The port can be any input, input/output, output, or variable port. When grouping data, the Aggregator transformation outputs the last row of each group unless otherwise specified.
- ϕ **Sorted input.** Select this option to improve session performance. To use sorted input, you must pass data to the Aggregator transformation sorted by group by port, in ascending or descending order.

You can configure the Aggregator transformation components and options on the Properties and Ports tab.

Configuring Aggregator Transformation Properties

Modify the Aggregator Transformation properties on the Properties tab.

Configure the following options:

Aggregator Setting	Description
Cache Directory	Local directory where the Integration Service creates the index and data cache files. By default, the Integration Service uses the directory entered in the Workflow
Tracing Level	Manager for the process variable SPMCacheDir. If you enter a new directory, make sure the directory exists and contains enough disk space for the aggregate caches. If you have enabled incremental aggregation, the Integration Service creates a backup of the files each time you run the session. The cache directory must contain enough disk space for two sets of the files. Amount of detail displayed in the session log for this transformation.
Sorted Input	Indicates input data is presorted by groups. Select this option only if the mapping passes sorted data to the Aggregator transformation.
Aggregator Data Cache Size	Data cache size for the transformation. Default cache size is 2,000,000 bytes. If the total configured session cache size is 2 GB (2,147,483,648 bytes) or greater, you must run the session on a 64-bit Integration Service. You can configure the Integration Service to determine the cache size at run time, or you can configure a numeric value. If you configure the Integration Service to determine the cache size, you can also configure a maximum amount of memory for the Integration Service to allocate to the cache.
Aggregator Index Cache Size	Index cache size for the transformation. Default cache size is 1,000,000 bytes. If the total configured session cache size is 2 GB (2,147,483,648 bytes) or greater, you must run the session on a 64-bit Integration Service. You can configure the Integration Service to determine the cache size at run time, or you can configure a numeric value. If you configure the Integration Service to determine the cache size, you can also configure a maximum amount of memory for the Integration Service to allocate to the cache.
Transformation Scope	Specifies how the Integration Service applies the transformation logic to incoming data: - Transaction. Applies the transformation logic to all rows in a transaction. Choose Transaction when a row of data depends on all rows in the same transaction, but does not depend on rows in other transactions. - All Input. Applies the transformation logic on all incoming data. When you choose All Input, the PowerCenter drops incoming transaction boundaries. Choose All Input when a row of data depends on all rows in the source.

Configuring Aggregator Transformation Ports

To configure ports in the Aggregator transformation, complete the following tasks:

- ϕ Enter an expression in any output port, using conditional clauses or non-aggregate functions in the port.
- ϕ Create multiple aggregate output ports.
- ϕ Configure any input, input/output, output, or variable port as a group by port.
- ϕ Improve performance by connecting only the necessary input/output ports to subsequent transformations, reducing the size of the data cache.
- ϕ Use variable ports for local variables.
- ϕ Create connections to other transformations as you enter an expression.

Configuring Aggregate Caches

When you run a session that uses an Aggregator transformation, the Integration Service creates index and data caches in memory to process the transformation. If the Integration Service requires more space, it stores overflow values in cache files.

You can configure the index and data caches in the Aggregator transformation or in the session properties. Or, you can configure the Integration Service to determine the cache size at run time.

Note: The Integration Service uses memory to process an Aggregator transformation with sorted ports. It does not use cache memory. You do not need to configure cache memory for Aggregator transformations that use sorted ports.

Aggregate Expressions

The Designer allows aggregate expressions only in the Aggregator transformation. An aggregate expression can include conditional clauses and non-aggregate functions. It can also include one aggregate function nested within another aggregate function, such as:

```
MAX ( COUNT ( ITEM ) )
```

The result of an aggregate expression varies depending on the group by ports used in the transformation. For example, when the Integration Service calculates the following aggregate expression with no group by ports defined, it finds the total quantity of items sold:

```
SUM ( QUANTITY )
```

However, if you use the same expression, and you group by the ITEM port, the Integration Service returns the total quantity of items sold, by item.

You can create an aggregate expression in any output port and use multiple aggregate ports in a transformation.

RELATED TOPICS:

- ϕ “Working with Expressions”

Aggregate Functions

Use the following aggregate functions within an Aggregator transformation. You can nest one aggregate function within another aggregate function.

The transformation language includes the following aggregate functions:

φ AVG
φ COUNT
φ FIRST
φ LAST
φ MAX
φ MEDIAN
φ MIN
φ PERCENTILE
φ STDDEV
φ SUM
φ VARIANCE

When you use any of these functions, you must use them in an expression within an Aggregator transformation.

Nested Aggregate Functions

You can include multiple single-level or multiple nested functions in different output ports in an Aggregator transformation. However, you cannot include both single-level and nested functions in an Aggregator transformation. Therefore, if an Aggregator transformation contains a single-level function in any output port, you cannot use a nested function in any other port in that transformation. When you include single-level and nested functions in the same Aggregator transformation, the Designer marks the mapping or mapplet invalid. If you need to create both single-level and nested functions, create separate Aggregator transformations.

Conditional Clauses

Use conditional clauses in the aggregate expression to reduce the number of rows used in the aggregation. The conditional clause can be any clause that evaluates to TRUE or FALSE.

For example, use the following expression to calculate the total commissions of employees who exceeded their quarterly quota:

```
SUM( COMMISSION, COMMISSION > QUOTA )
```

Non-Aggregate Functions

You can also use non-aggregate functions in the aggregate expression.

The following expression returns the highest number of items sold for each item (grouped by item). If no items were sold, the expression returns 0.

```
IIF( MAX( QUANTITY ) > 0, MAX( QUANTITY ), 0 )
```

Null Values in Aggregate Functions

When you configure the Integration Service, you can choose how you want the Integration Service to handle null values in aggregate functions. You can choose to treat null values in aggregate functions as NULL or zero. By default, the Integration Service treats null values as NULL in aggregate functions.

Group By Ports

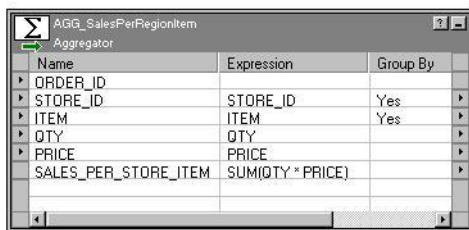
The Aggregator transformation lets you define groups for aggregations, rather than performing the aggregation across all input data. For example, rather than finding the total company sales, you can find the total sales grouped by region.

To define a group for the aggregate expression, select the appropriate input, input/output, output, and variable ports in the Aggregator transformation. You can select multiple group by ports to create a new group for each unique combination. The Integration Service then performs the defined aggregation for each group.

When you group values, the Integration Service produces one row for each group. If you do not group values, the Integration Service returns one row for all input rows. The Integration Service typically returns the last row of each group (or the last row received) with the result of the aggregation. However, if you specify a particular row to be returned (for example, by using the FIRST function), the Integration Service then returns the specified row.

When selecting multiple group by ports in the Aggregator transformation, the Integration Service uses port order to determine the order by which it groups. Since group order can affect the results, order group by ports to ensure the appropriate grouping. For example, the results of grouping by ITEM_ID then QUANTITY can vary from grouping by QUANTITY then ITEM_ID, because the numeric values for quantity are not necessarily unique.

The following Aggregator transformation groups first by STORE_ID and then by ITEM:



Name	Expression	Group By
ORDER_ID		
STORE_ID	STORE_ID	Yes
ITEM	ITEM	Yes
QTY	QTY	
PRICE	PRICE	
SALES_PER_STORE_ITEM	SUM(QTY * PRICE)	

If you send the following data through this Aggregator transformation:

STORE_ID	ITEM	QTY	PRICE
101	'battery'	3	2.99
101	'battery'	1	3.19
101	'battery'	2	2.59
101	'AAA '	2	2.45
201	'battery'	1	1.99
201	'battery'	4	1.59
301	'battery'	1	2.45

The Integration Service performs the aggregate calculation on the following unique groups:

STORE_ID	ITEM
101	'battery'
101	'AAA '
201	'battery'
301	'battery'

The Integration Service then passes the last row received, along with the results of the aggregation, as follows:

STORE_ID	ITEM	QTY	PRICE	SALES_PER_STORE
101	'battery'	2	2.59	17.34

STORE_ID	ITEM	QTY	PRICE	SALES_PER_STORE
101	'AAA'	2	2.45	4.90
201	'battery'	4	1.59	8.35
301	'battery'	1	2.45	2.45

Non-Aggregate Expressions

Use non-aggregate expressions in group by ports to modify or replace groups. For example, if you want to replace 'AAA battery' before grouping, you can create a new group by output port, named CORRECTED_ITEM, using the following expression:

```
IIF( ITEM = 'AAA battery', battery, ITEM )
```

Default Values

Define a default value for each port in the group to replace null input values. This allows the Integration Service to include null item groups in the aggregation.

RELATED TOPICS:

φ “Using Default Values for Ports”

Using Sorted Input

You can improve Aggregator transformation performance by using the sorted input option. When you use sorted input, the Integration Service assumes all data is sorted by group and it performs aggregate calculations as it reads rows for a group. When necessary, it stores group information in memory. To use the Sorted Input option, you must pass sorted data to the Aggregator transformation. You can gain performance with sorted ports when you configure the session with multiple partitions.

When you do not use sorted input, the Integration Service performs aggregate calculations as it reads. Since the data is not sorted, the Integration Service stores data for each group until it reads the entire source to ensure all aggregate calculations are accurate.

For example, one Aggregator transformation has the STORE_ID and ITEM group by ports, with the sorted input option selected. When you pass the following data through the Aggregator, the Integration Service performs an aggregation for the three rows in the 101/battery group as soon as it finds the new group, 201/battery:

STORE_ID	ITEM	QTY	PRICE
101	'battery'	3	2.99
101	'battery'	1	3.19
101	'battery'	2	2.59
201	'battery'	4	1.59
201	'battery'	1	1.99

If you use sorted input and do not presort data correctly, you receive unexpected results.

Sorted Input Conditions

Do not use sorted input if either of the following conditions are true:

- φ The aggregate expression uses nested aggregate functions.
- φ The session uses incremental aggregation.

If you use sorted input and do not sort data correctly, the session fails.

Sorting Data

To use sorted input, you pass sorted data through the Aggregator.

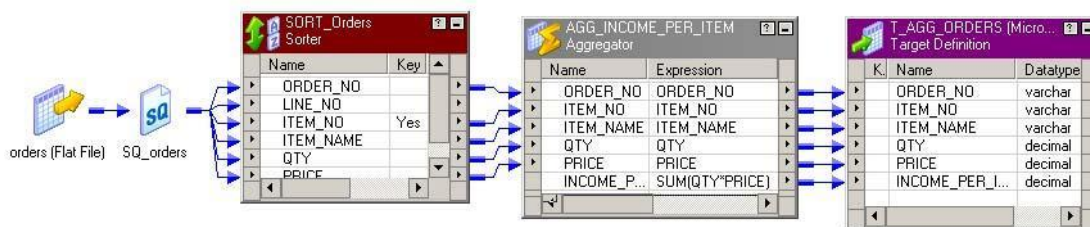
Data must be sorted in the following ways:

- φ By the Aggregator group by ports, in the order they appear in the Aggregator transformation.
- φ Using the same sort order configured for the session. If data is not in strict ascending or descending order based on the session sort order, the Integration Service fails the session. For example, if you configure a session to use a French sort order, data passing into the Aggregator transformation must be sorted using the French sort order.

For relational and file sources, use the Sorter transformation to sort data in the mapping before passing it to the Aggregator transformation. You can place the Sorter transformation anywhere in the mapping prior to the Aggregator if no transformation changes the order of the sorted data. Group by columns in the Aggregator transformation must be in the same order as they appear in the Sorter transformation.

If the session uses relational sources, you can also use the Number of Sorted Ports option in the Source Qualifier transformation to sort group by columns in the source database. Group by columns must be in the same order in both the Aggregator and Source Qualifier transformations.

The following mapping shows a Sorter transformation configured to sort the source data in ascending order by ITEM_NO:



The Sorter transformation sorts the data as follows:

ITEM_NO	ITEM_NAME	QTY	PRICE
345	Soup	4	2.95
345	Soup	1	2.95
345	Soup	2	3.25
546	Cereal	1	4.49
546	Cereal	2	5.25

With sorted input, the Aggregator transformation returns the following results:

ITEM_NAME	QTY	PRICE	INCOME_PER_ITEM
Cereal	2	5.25	14.99
Soup	2	3.25	21.25

Creating an Aggregator Transformation

To use an Aggregator transformation in a mapping, add the Aggregator transformation to the mapping. Then configure the transformation with an aggregate expression and group by ports.

To create an Aggregator transformation:

1. In the Mapping Designer, click Transformation > Create. Select the Aggregator transformation.
2. Enter a name for the Aggregator, click Create. Then click Done.

The Designer creates the Aggregator transformation.

3. Drag the ports to the Aggregator transformation.
The Designer creates input/output ports for each port you include.
4. Double-click the title bar of the transformation to open the Edit Transformations dialog box.

5. Select the Ports tab.

6. Click the group by option for each column you want the Aggregator to use in creating groups.
Optionally, enter a default value to replace null groups.

7. Click Add to add an expression port.

The expression port must be an output port. Make the port an output port by clearing Input (I).

8. Optionally, add default values for specific ports.

If the target database does not handle null values and certain ports are likely to contain null values, specify a default value.

9. Configure properties on the Properties tab.

CHAPTER 7

Expression Transformation

This chapter includes the following topics:

- φ Overview
- φ Expression Transformation Components
- φ Configuring Ports
- φ Creating an Expression Transformation

Overview

Transformation type:

Passive
Connected

Use the Expression transformation to calculate values in a single row. For example, you might need to adjust employee salaries, concatenate first and last names, or convert strings to numbers. You can also use the Expression transformation to test conditional statements before you pass the results to a target or other transformations.

Use the Expression transformation to perform non-aggregate calculations. To perform calculations involving multiple rows, such as sums or averages, use the Aggregator transformation.

Figure 5-1 shows a simple mapping with an Expression transformation used to concatenate the first and last names of employees from the EMPLOYEES table:

Figure 5-1. Sample Mapping with an Expression Transformation



Expression Transformation Components

You can create an Expression transformation in the Transformation Developer or the Mapping Designer.

An Expression transformation contains the following tabs:

- φ **Transformation.** Enter the name and description of the transformation. The naming convention for an Expression transformation is `EXP_TransformationName`. You can also make the transformation reusable.
- φ **Ports.** Create and configure ports.
- φ **Properties.** Configure the tracing level to determine the amount of transaction detail reported in the session log file.
- φ **Metadata Extensions.** Specify the extension name, datatype, precision, and value. You can also create reusable metadata extensions.

Configuring Ports

You can create and modify ports on the Ports tab.

Configure the following components on the Ports tab:

- φ **Port name.** Name of the port.
- φ **Datatype, precision, and scale.** Configure the datatype and set the precision and scale for each port.
- φ **Port type.** A port can be input, output, input/output, or variable. The input ports receive data and output ports pass data. The input/output ports pass data unchanged. Variable ports store data temporarily and can store values across the rows.
- φ **Expression.** Use the Expression Editor to enter expressions. Expressions use the transformation language, which includes SQL-like functions, to perform calculations.
- φ **Default values and description.** Set default value for ports and add description.

Calculating Values

To calculate values for a single row using the Expression transformation, you must include the following ports:

- φ **Input or input/output ports.** Provides values used in a calculation. For example, if you need to calculate the total price for an order, create two input or input/output ports. One port provides the unit price and the other provides the quantity ordered.
- φ **Output ports.** Provides the return value of the expression. You enter the expression as a configuration option for the output port. You can also configure a default value for each port.

You can enter multiple expressions in a single Expression transformation by creating an expression for each output port. For example, you might want to calculate different types of withholding taxes from each employee paycheck, such as local and federal income tax, Social Security and Medicare. Since all of these calculations require the employee salary, the withholding category, and may require the corresponding tax rate, you can create input/output ports for the salary and withholding category and a separate output port for each calculation.

Creating an Expression Transformation

Use the following procedure to create an Expression transformation.

To create an Expression transformation:

1. In the Mapping Designer, open a mapping.
2. Click Transformation > Create. Select Expression transformation.
3. Enter a name and click Done.
4. Select and drag the ports from the source qualifier or other transformations to add to the Expression transformation.

You can also open the transformation and create ports manually.

5. Double-click on the title bar and click on Ports tab. You can create output and variable ports within the transformation.
6. In the Expression section of an output or variable port, open the Expression Editor.
7. Enter an expression. Click Validate to verify the expression syntax.
8. Click OK.
9. Assign the port datatype, precision, and scale to match the expression return value.
10. Create reusable transformations on the Transformation tab.
Note: After you make the transformation reusable, you cannot copy ports from the source qualifier or other transformations. You can create ports manually within the transformation.
11. Configure the tracing level on the Properties tab.
12. Add metadata extensions on the Metadata Extensions tab.
13. Click OK.
14. Connect the output ports to a downstream transformation or target.

CHAPTER 8

Filter Transformation

This chapter includes the following topics:

- φ Overview
- φ Filter Transformation Components
- φ Filter Condition
- φ Steps to Create a Filter Transformation

Overview

Transformation type:

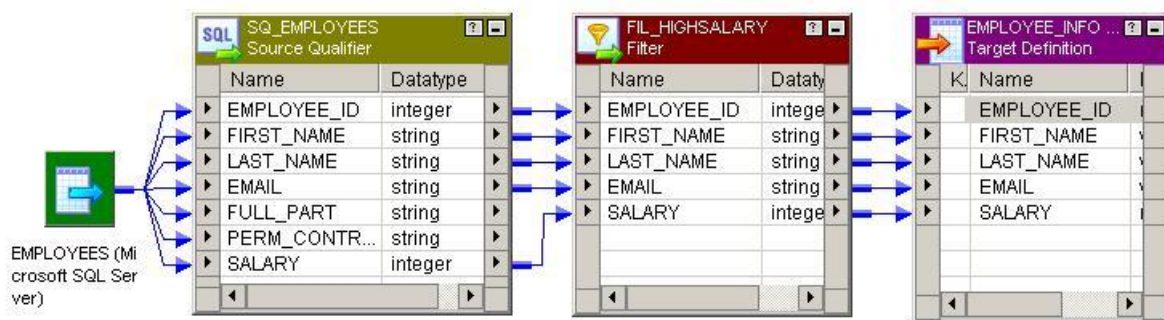
Active
Connected

Use the Filter transformation to filter out rows in a mapping. As an active transformation, the Filter transformation may change the number of rows passed through it. The Filter transformation allows rows that meet the specified filter condition to pass through. It drops rows that do not meet the condition. You can filter data based on one or more conditions.

A filter condition returns TRUE or FALSE for each row that the Integration Service evaluates, depending on whether a row meets the specified condition. For each row that returns TRUE, the Integration Services pass through the transformation. For each row that returns FALSE, the Integration Service drops and writes a message to the session log.

The following mapping passes the rows from a human resources table that contains employee data through a Filter transformation. The filter allows rows through for employees that make salaries of \$30,000 or higher.

Figure 7-1. Sample Mapping with a Filter Transformation



You cannot concatenate ports from more than one transformation into the Filter transformation. The input ports for the filter must come from a single transformation.

Tip: Place the Filter transformation as close to the sources in the mapping as possible to maximize session performance. Rather than passing rows you plan to discard through the mapping, you can filter out unwanted data early in the flow of data from sources to targets.

Filter Transformation Components

You can create a Filter transformation in the Transformation Developer or the Mapping Designer. A Filter transformation contains the following tabs:

- φ **Transformation.** Enter the name and description of the transformation. The naming convention for a Filter transformation is *FIL_TransformationName*. You can also make the transformation reusable.
- φ **Ports.** Create and configure ports.
- φ **Properties.** Configure the filter condition to filter rows. Use the Expression Editor to enter the filter condition. You can also configure the tracing level to determine the amount of transaction detail reported in the session log file.
- φ **Metadata Extensions.** Create a non-reusable metadata extension to extend the metadata of the transformation transformation. Configure the extension name, datatype, precision, and value. You can also promote metadata extensions to reusable extensions if you want to make it available to all transformation transformations.

Configuring Filter Transformation Ports

You can create and modify ports on the Ports tab.

You can configure the following properties on the Ports tab:

- φ **Port name.** Name of the port.
- φ **Datatype, precision, and scale.** Configure the datatype and set the precision and scale for each port.
- φ **Port type.** All ports are input/output ports. The input ports receive data and output ports pass data.
- φ **Default values and description.** Set default value for ports and add description.

Filter Condition

The filter condition is an expression that returns TRUE or FALSE. Enter conditions using the Expression Editor available on the Properties tab.

Any expression that returns a single value can be used as a filter. For example, if you want to filter *out* rows for employees whose salary is less than \$30,000, you enter the following condition:

```
SALARY > 30000
```

You can specify multiple components of the condition, using the AND and OR logical operators. If you want to filter out employees who make less than \$30,000 and more than \$100,000, you enter the following condition:

```
SALARY > 30000 AND SALARY < 100000
```

You can also enter a constant for the filter condition. The numeric equivalent of FALSE is zero (0). Any non-zero value is the equivalent of TRUE. For example, the transformation contains a port named NUMBER_OF_UNITS with a numeric datatype. You configure a filter condition to return FALSE if the value of NUMBER_OF_UNITS equals zero. Otherwise, the condition returns TRUE.

You do not need to specify TRUE or FALSE as values in the expression. TRUE and FALSE are implicit return values from any condition you set. If the filter condition evaluates to NULL, the row is treated as FALSE.

Note: The filter condition is case sensitive.

Filtering Rows with Null Values

To filter rows containing null values or spaces, use the ISNULL and IS_SPACES functions to test the value of the port. For example, if you want to filter out rows that contain NULL value in the FIRST_NAME port, use the following condition:

```
IIF (ISNULL (FIRST_NAME) , FALSE, TRUE)
```

This condition states that if the FIRST_NAME port is NULL, the return value is FALSE and the row should be discarded. Otherwise, the row passes through to the next transformation.

Steps to Create a Filter Transformation

Use the following procedure to create a Filter transformation.

To create a Filter transformation:

1. In the Mapping Designer, open a mapping.
2. Click Transformation > Create. Select Filter transformation.
3. Enter a name for the transformation. Click Create and then click Done.
4. Select and drag all the ports from a source qualifier or other transformation to add them to the Filter transformation.
5. Double-click on the title bar and click on Ports tab. You can also manually create ports within the transformation.
6. Click the Properties tab to configure the filter condition and tracing level.
7. In the Value section of the filter condition, open the Expression Editor.
8. Enter the filter condition you want to apply. The default condition returns TRUE.

Use values from one of the input ports in the transformation as part of this condition. However, you can also use values from output ports in other transformations.

9. Enter an expression. Click Validate to verify the syntax of the conditions you entered.
10. Select the tracing level.
11. Add metadata extensions on the Metadata Extensions tab.

CHAPTER 9

Joiner Transformation

This chapter includes the following topics:

- ϕ Overview
- ϕ Joiner Transformation Properties
- ϕ Defining a Join Condition
- ϕ Defining the Join Type
- ϕ Using Sorted Input
- ϕ Joining Data from a Single Source
- ϕ Blocking the Source Pipelines
- ϕ Working with Transactions
- ϕ Creating a Joiner Transformation

Overview

Transformation type:

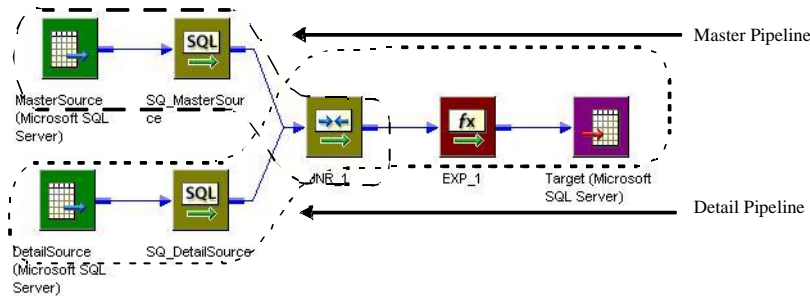
Active
Connected

Use the Joiner transformation to join source data from two related heterogeneous sources residing in different locations or file systems. You can also join data from the same source. The Joiner transformation joins sources with at least one matching column. The Joiner transformation uses a condition that matches one or more pairs of columns between the two sources.

The two input pipelines include a master pipeline and a detail pipeline or a master and a detail branch. The master pipeline ends at the Joiner transformation, while the detail pipeline continues to the target.

Figure 13-1 shows the master and detail pipelines in a mapping with a Joiner transformation:

Figure 13-1. Mapping with Master and Detail Pipelines



To join more than two sources in a mapping, join the output from the Joiner transformation with another source pipeline. Add Joiner transformations to the mapping until you have joined all the source pipelines.

The Joiner transformation accepts input from most transformations. However, consider the following limitations on the pipelines you connect to the Joiner transformation:

- ϕ You cannot use a Joiner transformation when either input pipeline contains an Update Strategy transformation.
- ϕ You cannot use a Joiner transformation if you connect a Sequence Generator transformation directly before the Joiner transformation.

Working with the Joiner Transformation

When you work with the Joiner transformation, you must configure the transformation properties, join type, and join condition. You can configure the Joiner transformation for sorted input to improve Integration Service performance. You can also configure the transformation scope to control how the Integration Service applies transformation logic. To work with the Joiner transformation, complete the following tasks:

- ϕ **Configure the Joiner transformation properties.** Properties for the Joiner transformation identify the location of the cache directory, how the Integration Service processes the transformation, and how the Integration Service handles caching. For more information, see “Joiner Transformation Properties” on page 185.
- ϕ **Configure the join condition.** The join condition contains ports from both input sources that must match for the Integration Service to join two rows. Depending on the type of join selected, the Integration Service either adds the row to the result set or discards the row.
- ϕ **Configure the join type.** A join is a relational operator that combines data from multiple tables in different databases or flat files into a single result set. You can configure the Joiner transformation to use a Normal, Master Outer, Detail Outer, or Full Outer join type. For more information, see “Defining the Join Type” on page 186.
- ϕ **Configure the session for sorted or unsorted input.** You can improve session performance by configuring the Joiner transformation to use sorted input. To configure a mapping to use sorted data, you establish and maintain a sort order in the mapping so that the Integration Service can use the sorted data when it processes the Joiner transformation.
- ϕ **Configure the transaction scope.** When the Integration Service processes a Joiner transformation, it can apply transformation logic to all data in a transaction, all incoming data, or one row of data at a time.

If you have the partitioning option in PowerCenter, you can increase the number of partitions in a pipeline to improve session performance.

Joiner Transformation Properties

Properties for the Joiner transformation identify the location of the cache directory, how the Integration Service processes the transformation, and how the Integration Service handles caching. The properties also determine how the Integration Service joins tables and files.

When you create a mapping, you specify the properties for each Joiner transformation. When you create a session, you can override some properties, such as the index and data cache size for each transformation.

The following table describes the Joiner transformation properties:

Option	Description
Case-Sensitive String Comparison	If selected, the Integration Service uses case-sensitive string comparisons when performing joins on string columns.
Cache Directory	Specifies the directory used to cache master or detail rows and the index to these rows. By default, the cache files are created in a directory specified by the process variable \$PMCacheDir. If you override the directory, make sure the directory exists and contains enough disk space for the cache files. The directory can be a mapped or mounted drive.
Join Type	Specifies the type of join: Normal, Master Outer, Detail Outer, or Full Outer.
Null Ordering in Master	Not applicable for this transformation type.
Null Ordering in Detail	Not applicable for this transformation type.
Tracing Level	Amount of detail displayed in the session log for this transformation. The options are Terse, Normal, Verbose Data, and Verbose Initialization.
Joiner Data Cache Size	Data cache size for the transformation. Default cache size is 2,000,000 bytes. If the total configured cache size is 2 GB or more, you must run the session on a 64-bit Integration Service. You can configure a numeric value, or you can configure the Integration Service to determine the cache size at runtime. If you configure the Integration Service to determine the cache size, you can also configure a maximum amount of memory for the Integration Service to allocate to the cache.
Joiner Index Cache Size	Index cache size for the transformation. Default cache size is 1,000,000 bytes. If the total configured cache size is 2 GB or more, you must run the session on a 64-bit Integration Service. You can configure a numeric value, or you can configure the Integration Service to determine the cache size at runtime. If you configure the Integration Service to determine the cache size, you can also configure a maximum amount of memory for the Integration Service to allocate to the cache.
Sorted Input	Specifies that data is sorted. Choose Sorted Input to join sorted data. Using sorted input can improve performance.
Master Sort Order	Specifies the sort order of the master source data. Choose Ascending if the master source data is in ascending order. If you choose Ascending, also enable sorted input. Default is Auto.
Transformation Scope	Specifies how the Integration Service applies the transformation logic to incoming data. You can choose Transaction, All Input, or Row.

Defining a Join Condition

The join condition contains ports from both input sources that must match for the Integration Service to join two rows. Depending on the type of join selected, the Integration Service either adds the row to the result set or discards the row. The Joiner transformation produces result sets based on the join type, condition, and input data sources.

Before you define a join condition, verify that the master and detail sources are configured for optimal performance. During a session, the Integration Service compares each row of the master source against the detail source. To improve performance for an unsorted Joiner transformation, use the source with fewer rows as the master source. To improve performance for a sorted Joiner transformation, use the source with fewer duplicate key values as the master.

By default, when you add ports to a Joiner transformation, the ports from the first source pipeline display as detail sources. Adding the ports from the second source pipeline sets them as master sources. To change these settings, click the M column on the Ports tab for the ports you want to set as the master source. This sets ports from this source as master ports and ports from the other source as detail ports.

You define one or more conditions based on equality between the specified master and detail sources. For example, if two sources with tables called EMPLOYEE_AGE and EMPLOYEE_POSITION both contain employee ID numbers, the following condition matches rows with employees listed in both sources:

```
EMP_ID1 = EMP_ID2
```

Use one or more ports from the input sources of a Joiner transformation in the join condition. Additional ports increase the time necessary to join two sources. The order of the ports in the condition can impact the performance of the Joiner transformation. If you use multiple ports in the join condition, the Integration Service compares the ports in the order you specify.

The Designer validates datatypes in a condition. Both ports in a condition must have the same datatype. If you need to use two ports in the condition with non-matching datatypes, convert the datatypes so they match.

If you join Char and Varchar datatypes, the Integration Service counts any spaces that pad Char values as part of the string:

```
Char(40) = "abcd"  
Varchar(40) = "abcd"
```

The Char value is "abcd" padded with 36 blank spaces, and the Integration Service does not join the two fields because the Char field contains trailing spaces.

Note: The Joiner transformation does not match null values. For example, if both EMP_ID1 and EMP_ID2 contain a row with a null value, the Integration Service does not consider them a match and does not join the two rows. To join rows with null values, replace null input with default values, and then join on the default values.

Defining the Join Type

In SQL, a join is a relational operator that combines data from multiple tables into a single result set. The Joiner transformation is similar to an SQL join except that data can originate from different types of sources.

You define the join type on the Properties tab in the transformation. The Joiner transformation supports the following types of joins:

- ϕ Normal
- ϕ Master Outer
- ϕ Detail Outer
- ϕ Full Outer

Note: A normal or master outer join performs faster than a full outer or detail outer join.

If a result set includes fields that do not contain data in either of the sources, the Joiner transformation populates the empty fields with null values. If you know that a field will return a NULL and you do not want to insert NULLs in the target, you can set a default value on the Ports tab for the corresponding port.

Normal Join

With a normal join, the Integration Service discards all rows of data from the master and detail source that do not match, based on the condition.

For example, you might have two sources of data for auto parts called PARTS_SIZE and PARTS_COLOR with the following data:

PARTS_SIZE (master source)

PART_ID1	DESCRIPTION	SIZE
1	Seat Cover	Large
2	Ash Tray	Small
3	Floor Mat	Medium

PARTS_COLOR (detail source)

PART_ID2	DESCRIPTION	COLOR
1	Seat Cover	Blue
3	Floor Mat	Black
4	Fuzzy Dice	Yellow

To join the two tables by matching the PART_IDs in both sources, you set the condition as follows:

```
PART_ID1 = PART_ID2
```

When you join these tables with a normal join, the result set includes the following data:

PART_ID	DESCRIPTION	SIZE	COLOR
1	Seat Cover	Large	Blue
3	Floor Mat	Medium	Black

The following example shows the equivalent SQL statement:

```
SELECT * FROM PARTS_SIZE, PARTS_COLOR WHERE PARTS_SIZE.PART_ID1 = PARTS_COLOR.PART_ID2
```

Master Outer Join

A master outer join keeps all rows of data from the detail source and the matching rows from the master source. It discards the unmatched rows from the master source.

When you join the sample tables with a master outer join and the same condition, the result set includes the following data:

PART_ID	DESCRIPTION	SIZE	COLOR
1	Seat Cover	Large	Blue
3	Floor Mat	Medium	Black
4	Fuzzy Dice	NULL	Yellow

Because no size is specified for the Fuzzy Dice, the Integration Service populates the field with a NULL.

The following example shows the equivalent SQL statement:

```
SELECT * FROM PARTS_SIZE RIGHT OUTER JOIN PARTS_COLOR ON (PARTS_COLOR.PART_ID2 = PARTS_SIZE.PART_ID1)
```

Detail Outer Join

A detail outer join keeps all rows of data from the master source and the matching rows from the detail source. It discards the unmatched rows from the detail source.

When you join the sample tables with a detail outer join and the same condition, the result set includes the following data:

PART_ID	DESCRIPTION	SIZE	COLOR
1	Seat Cover	Large	Blue
2	Ash Tray	Small	NULL
3	Floor Mat	Medium	Black

Because no color is specified for the Ash Tray, the Integration Service populates the field with a NULL.

The following example shows the equivalent SQL statement:

```
SELECT * FROM PARTS_SIZE LEFT OUTER JOIN PARTS_COLOR ON (PARTS_SIZE.PART_ID1 =
PARTS_COLOR.PART_ID2)
```

Full Outer Join

A full outer join keeps all rows of data from both the master and detail sources.

When you join the sample tables with a full outer join and the same condition, the result set includes:

PART_ID	DESCRIPTION	SIZE	Color
1	Seat Cover	Large	Blue
2	Ash Tray	Small	NULL
3	Floor Mat	Medium	Black
4	Fuzzy Dice	NULL	Yellow

Because no color is specified for the Ash Tray and no size is specified for the Fuzzy Dice, the Integration Service populates the fields with NULL.

The following example shows the equivalent SQL statement:

```
SELECT * FROM PARTS_SIZE FULL OUTER JOIN PARTS_COLOR ON (PARTS_SIZE.PART_ID1 =
PARTS_COLOR.PART_ID2)
```

Using Sorted Input

You can improve session performance by configuring the Joiner transformation to use sorted input. When you configure the Joiner transformation to use sorted data, the Integration Service improves performance by minimizing disk input and output. You see the greatest performance improvement when you work with large data sets.

To configure a mapping to use sorted data, you establish and maintain a sort order in the mapping so the Integration Service can use the sorted data when it processes the Joiner transformation. Complete the following tasks to configure the mapping:

- ❖ **Configure the sort order.** Configure the sort order of the data you want to join. You can join sorted flat files, or you can sort relational data using a Source Qualifier transformation. You can also use a Sorter transformation.
- ❖ **Add transformations.** Use transformations that maintain the order of the sorted data.
- ❖ **Configure the Joiner transformation.** Configure the Joiner transformation to use sorted data and configure the join condition to use the sort origin ports. The sort origin represents the source of the sorted data.

When you configure the sort order in a session, you can select a sort order associated with the Integration Service code page. When you run the Integration Service in Unicode mode, it uses the selected session sort order to sort character data. When you run the Integration Service in ASCII mode, it sorts all character data

using a binary sort order. To ensure that data is sorted as the Integration Service requires, the database sort order must be the same as the user-defined session sort order.

When you join sorted data from partitioned pipelines, you must configure the partitions to maintain the order of sorted data.

Configuring the Sort Order

You must configure the sort order to ensure that the Integration Service passes sorted data to the Joiner transformation.

Configure the sort order using one of the following methods:

- ϕ **Use sorted flat files.** When the flat files contain sorted data, verify that the order of the sort columns match in each source file.
- ϕ **Use sorted relational data.** Use sorted ports in the Source Qualifier transformation to sort columns from the source database. Configure the order of the sorted ports the same in each Source Qualifier transformation.
- ϕ **Use Sorter transformations.** Use a Sorter transformation to sort relational or flat file data. Place a Sorter transformation in the master and detail pipelines. Configure each Sorter transformation to use the same order of the sort key ports and the sort order direction.

If you pass unsorted or incorrectly sorted data to a Joiner transformation configured to use sorted data, the session fails and the Integration Service logs the error in the session log file.

Adding Transformations to the Mapping

When you add transformations between the sort origin and the Joiner transformation, use the following guidelines to maintain sorted data:

- ϕ Do not place any of the following transformations between the sort origin and the Joiner transformation:
 - ☐ Custom
 - ☐ Unsorted Aggregator
 - ☐ Normalizer
 - ☐ Rank
 - ☐ Union transformation
 - ☐ XML Parser transformation
 - ☐ XML Generator transformation
 - ☐ Maplet, if it contains one of the above transformations
- ϕ You can place a sorted Aggregator transformation between the sort origin and the Joiner transformation if you use the following guidelines:
 - ☐ Configure the Aggregator transformation for sorted input using the guidelines in “Using Sorted Input” on page 28.
 - ☐ Use the same ports for the group by columns in the Aggregator transformation as the ports at the sort origin.
 - ☐ The group by ports must be in the same order as the ports at the sort origin.
- ϕ When you join the result set of a Joiner transformation with another pipeline, verify that the data output from the first Joiner transformation is sorted.

Tip: You can place the Joiner transformation directly after the sort origin to maintain sorted data.

Configuring the Joiner Transformation

To configure the Joiner transformation, complete the following tasks:

- ϕ Enable Sorted Input on the Properties tab.

- φ Define the join condition to receive sorted data in the same order as the sort origin.

Defining the Join Condition

Configure the join condition to maintain the sort order established at the sort origin: the sorted flat file, the Source Qualifier transformation, or the Sorter transformation. If you use a sorted Aggregator transformation between the sort origin and the Joiner transformation, treat the sorted Aggregator transformation as the sort origin when you define the join condition. Use the following guidelines when you define join conditions:

- φ The ports you use in the join condition must match the ports at the sort origin.
- φ When you configure multiple join conditions, the ports in the first join condition must match the first ports at the sort origin.
- φ When you configure multiple conditions, the order of the conditions must match the order of the ports at the sort origin, and you must not skip any ports.
- φ The number of sorted ports in the sort origin can be greater than or equal to the number of ports at the join condition.

Example of a Join Condition

For example, you configure Sorter transformations in the master and detail pipelines with the following sorted ports:

1. ITEM_NO
2. ITEM_NAME
3. PRICE

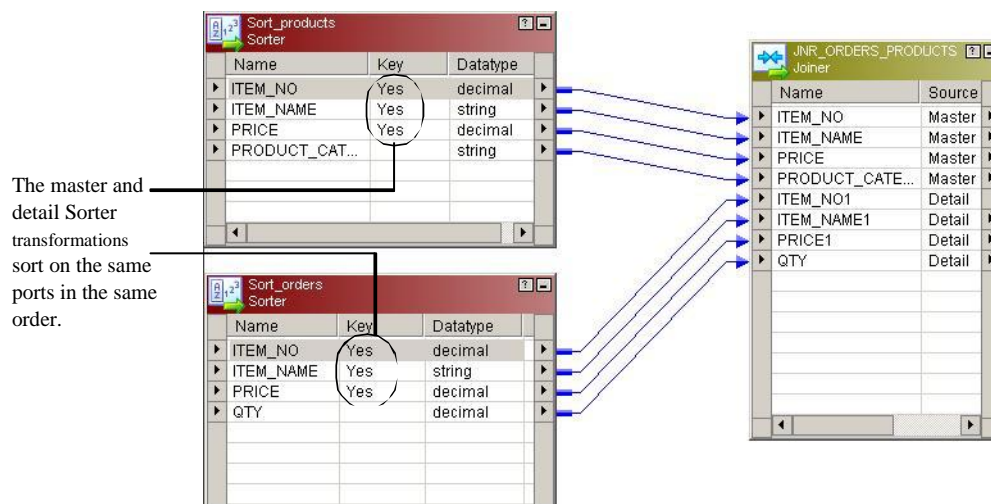
When you configure the join condition, use the following guidelines to maintain sort order:

- φ You must use ITEM_NO in the first join condition.
- φ If you add a second join condition, you must use ITEM_NAME.
- φ If you want to use PRICE in a join condition, you must also use ITEM_NAME in the second join condition.

If you skip ITEM_NAME and join on ITEM_NO and PRICE, you lose the sort order and the Integration Service fails the session.

Figure 13-2 shows a mapping configured to sort and join on the ports ITEM_NO, ITEM_NAME, and PRICE:

Figure 13-2. Mapping Configured to Join Data from Two Pipelines



When you use the Joiner transformation to join the master and detail pipelines, you can configure any one of the following join conditions:

```

ITEM_NO = ITEM_NO
or
ITEM_NO = ITEM_NO1
ITEM_NAME = ITEM_NAME1
or
ITEM_NO = ITEM_NO1
ITEM_NAME = ITEM_NAME1
PRICE = PRICE1

```

Joining Data from a Single Source

You may want to join data from the same source if you want to perform a calculation on part of the data and join the transformed data with the original data. When you join the data using this method, you can maintain the original data and transform parts of that data within one mapping. You can join data from the same source in the following ways:

- ϕ Join two branches of the same pipeline.
- ϕ Join two instances of the same source.

Joining Two Branches of the Same Pipeline

When you join data from the same source, you can create two branches of the pipeline. When you branch a pipeline, you must add a transformation between the source qualifier and the Joiner transformation in at least one branch of the pipeline. You must join sorted data and configure the Joiner transformation for sorted input.

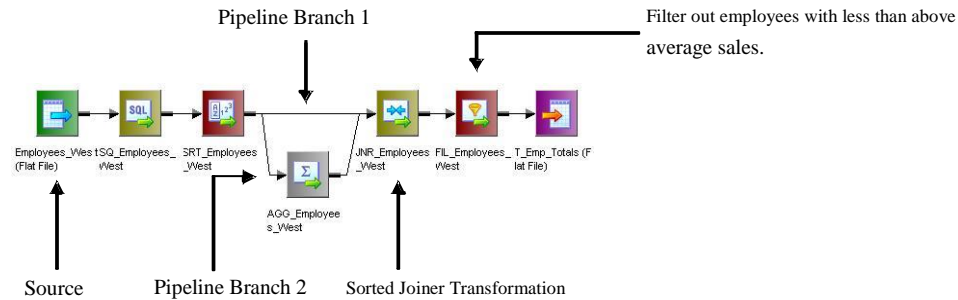
For example, you have a source with the following ports:

- ϕ Employee
- ϕ Department
- ϕ Total Sales

In the target, you want to view the employees who generated sales that were greater than the average sales for their departments. To do this, you create a mapping with the following transformations:

- ϕ **Sorter transformation.** Sorts the data.
- ϕ **Sorted Aggregator transformation.** Averages the sales data and group by department. When you perform this aggregation, you lose the data for individual employees. To maintain employee data, you must pass a branch of the pipeline to the Aggregator transformation and pass a branch with the same data to the Joiner transformation to maintain the original data. When you join both branches of the pipeline, you join the aggregated data with the original data.
- ϕ **Sorted Joiner transformation.** Uses a sorted Joiner transformation to join the sorted aggregated data with the original data.
- ϕ **Filter transformation.** Compares the average sales data against sales data for each employee and filter out employees with less than above average sales.

The following figure shows a mapping that joins two branches of the same pipeline:



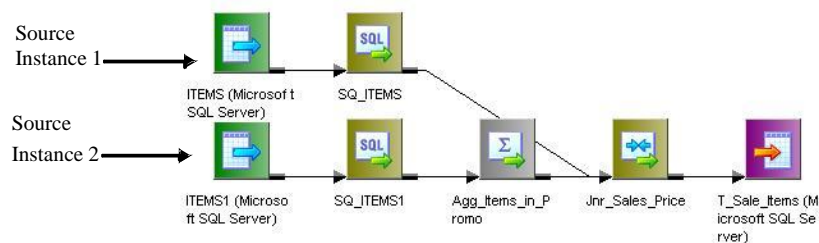
Note: You can also join data from output groups of the same transformation, such as the Custom transformation or XML Source Qualifier transformation. Place a Sorter transformation between each output group and the Joiner transformation and configure the Joiner transformation to receive sorted input.

Joining two branches might impact performance if the Joiner transformation receives data from one branch much later than the other branch. The Joiner transformation caches all the data from the first branch, and writes the cache to disk if the cache fills. The Joiner transformation must then read the data from disk when it receives the data from the second branch. This can slow processing.

Joining Two Instances of the Same Source

You can also join same source data by creating a second instance of the source. After you create the second source instance, you can join the pipelines from the two source instances. If you want to join unsorted data, you must create two instances of the same source and join the pipelines.

The following figure shows two instances of the same source joined with a Joiner transformation:



Note: When you join data using this method, the Integration Service reads the source data for each source instance, so performance can be slower than joining two branches of a pipeline.

Guidelines

Use the following guidelines when deciding whether to join branches of a pipeline or join two instances of a source:

- φ Join two branches of a pipeline when you have a large source or if you can read the source data only once. For example, you can only read source data from a message queue once.
- φ Join two branches of a pipeline when you use sorted data. If the source data is unsorted and you use a Sorter transformation to sort the data, branch the pipeline after you sort the data.
- φ Join two instances of a source when you need to add a blocking transformation to the pipeline between the source and the Joiner transformation.
- φ Join two instances of a source if one pipeline may process slower than the other pipeline.
- φ Join two instances of a source if you need to join unsorted data.

Blocking the Source Pipelines

When you run a session with a Joiner transformation, the Integration Service blocks and unblocks the source data, based on the mapping configuration and whether you configure the Joiner transformation for sorted input.

Unsorted Joiner Transformation

When the Integration Service processes an unsorted Joiner transformation, it reads all master rows before it reads the detail rows. To ensure it reads all master rows before the detail rows, the Integration Service blocks the detail source while it caches rows from the master source. Once the Integration Service reads and caches all master rows, it unblocks the detail source and reads the detail rows. Some mappings with unsorted Joiner transformations violate data flow validation.

Sorted Joiner Transformation

When the Integration Service processes a sorted Joiner transformation, it blocks data based on the mapping configuration. Blocking logic is possible if master and detail input to the Joiner transformation originate from different sources.

The Integration Service uses blocking logic to process the Joiner transformation if it can do so without blocking all sources in a target load order group simultaneously. Otherwise, it does not use blocking logic. Instead, it stores more rows in the cache.

When the Integration Service can use blocking logic to process the Joiner transformation, it stores fewer rows in the cache, increasing performance.

Caching Master Rows

When the Integration Service processes a Joiner transformation, it reads rows from both sources concurrently and builds the index and data cache based on the master rows. The Integration Service then performs the join based on the detail source data and the cache data. The number of rows the Integration Service stores in the cache depends on the partitioning scheme, the source data, and whether you configure the Joiner transformation for sorted input. To improve performance for an unsorted Joiner transformation, use the source with fewer rows as the master source. To improve performance for a sorted Joiner transformation, use the source with fewer duplicate key values as the master.

Working with Transactions

When the Integration Service processes a Joiner transformation, it can apply transformation logic to all data in a transaction, all incoming data, or one row of data at a time. The Integration Service can drop or preserve transaction boundaries depending on the mapping configuration and the transformation scope. You configure how the Integration Service applies transformation logic and handles transaction boundaries using the transformation scope property.

You configure transformation scope values based on the mapping configuration and whether you want to preserve or drop transaction boundaries.

You can preserve transaction boundaries when you join the following sources:

- ϕ **You join two branches of the same source pipeline.** Use the Transaction transformation scope to preserve transaction boundaries.
- ϕ **You join two sources, and you want to preserve transaction boundaries for the detail source.** Use the Row transformation scope to preserve transaction boundaries in the detail pipeline.

You can drop transaction boundaries when you join the following sources:

- **You join two sources or two branches and you want to drop transaction boundaries.** Use the All Input transformation scope to apply the transformation logic to all incoming data and drop transaction boundaries for both pipelines.

Table 13-1 summarizes how to preserve transaction boundaries using transformation scopes with the Joiner transformation:

Table 13-1. Integration Service Behavior with Transformation Scopes for the Joiner Transformation

Transformation Scope	Input Type	Integration Service Behavior
Row	Unsorted	Preserves transaction boundaries in the detail pipeline.
	Sorted	Session fails.
*Transaction	Sorted	Preserves transaction boundaries when master and detail originate from the same transaction generator. Session fails when master and detail do not originate from the same transaction generator
	Unsorted	Session fails.
*All Input	Sorted, Unsorted	Drops transaction boundaries.
<i>*Sessions fail if you use real-time data with All Input or Transaction transformation scopes.</i>		

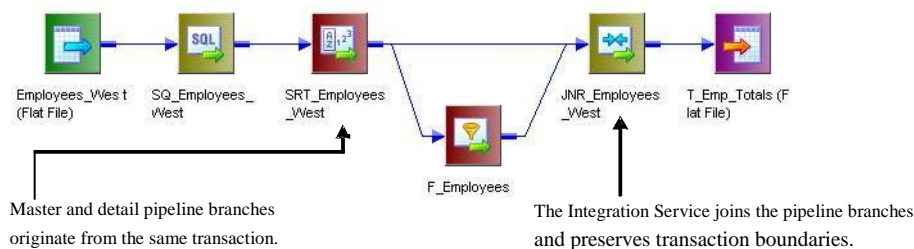
Preserving Transaction Boundaries for a Single Pipeline

When you join data from the same source, use the Transaction transformation scope to preserve incoming transaction boundaries for a single pipeline. Use the Transaction transformation scope when the Joiner transformation joins data from the same source, either two branches of the same pipeline or two output groups of one transaction generator. Use this transformation scope with sorted data and any join type.

When you use the Transaction transformation scope, verify that master and detail pipelines originate from the same transaction control point and that you use sorted input. For example, in Figure 13-3 the Sorter transformation is the transaction control point. You cannot place another transaction control point between the Sorter transformation and the Joiner transformation.

Figure 13-3 shows a mapping that joins two branches of a pipeline and preserves transaction boundaries:

Figure 13-3. Preserving Transaction Boundaries when You Join Two Pipeline Branches



Preserving Transaction Boundaries in the Detail Pipeline

When you want to preserve the transaction boundaries in the detail pipeline, choose the Row transformation scope. The Row transformation scope allows the Integration Service to process data one row at a time. The Integration Service caches the master data and matches the detail data with the cached master data.

When the source data originates from a real-time source, such as IBM MQ Series, the Integration Service matches the cached master data with each message as it is read from the detail source.

Use the Row transformation scope with Normal and Master Outer join types that use unsorted data.

Dropping Transaction Boundaries for Two Pipelines

When you want to join data from two sources or two branches and you do not need to preserve transaction boundaries, use the All Input transformation scope. When you use All Input, the Integration Service drops incoming transaction boundaries for both pipelines and outputs all rows from the transformation as an open transaction. At the Joiner transformation, the data from the master pipeline can be cached or joined concurrently, depending on how you configure the sort order. Use this transformation scope with sorted and unsorted data and any join type.

Creating a Joiner Transformation

To use a Joiner transformation, add a Joiner transformation to the mapping, set up the input sources, and configure the transformation with a condition and join type and sort type.

To create a Joiner transformation:

1. In the Mapping Designer, click Transformation > Create. Select the Joiner transformation. Enter a name, and click OK.

The naming convention for Joiner transformations is `JNR_TransformationName`. Enter a description for the transformation.

The Designer creates the Joiner transformation.

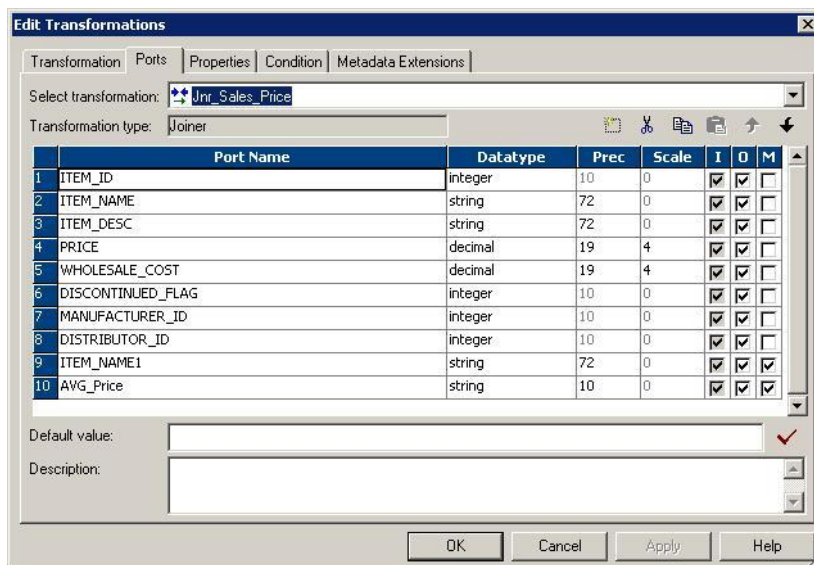
2. Drag all the input/output ports from the first source into the Joiner transformation.

The Designer creates input/output ports for the source fields in the Joiner transformation as detail fields by default. You can edit this property later.

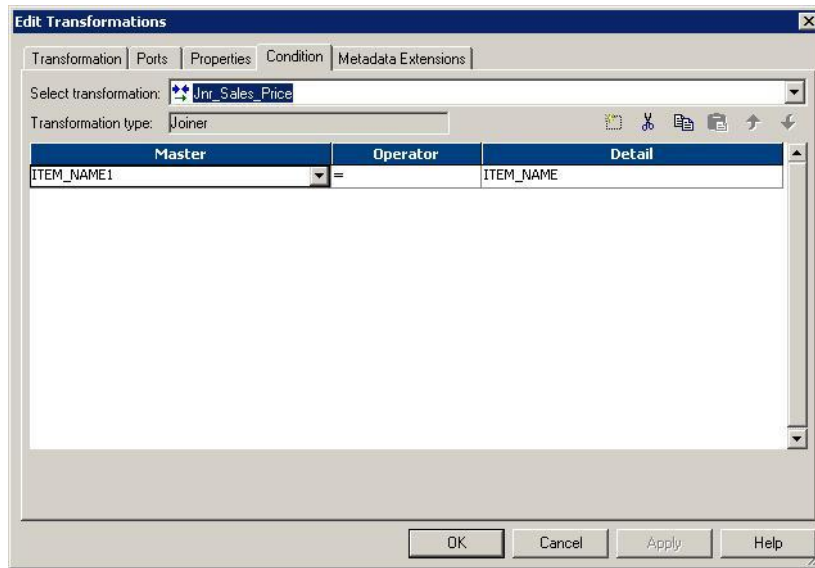
3. Select and drag all the input/output ports from the second source into the Joiner transformation.

The Designer configures the second set of source fields and master fields by default.

4. Double-click the title bar of the Joiner transformation to open the transformation.



5. Click the Ports tab.



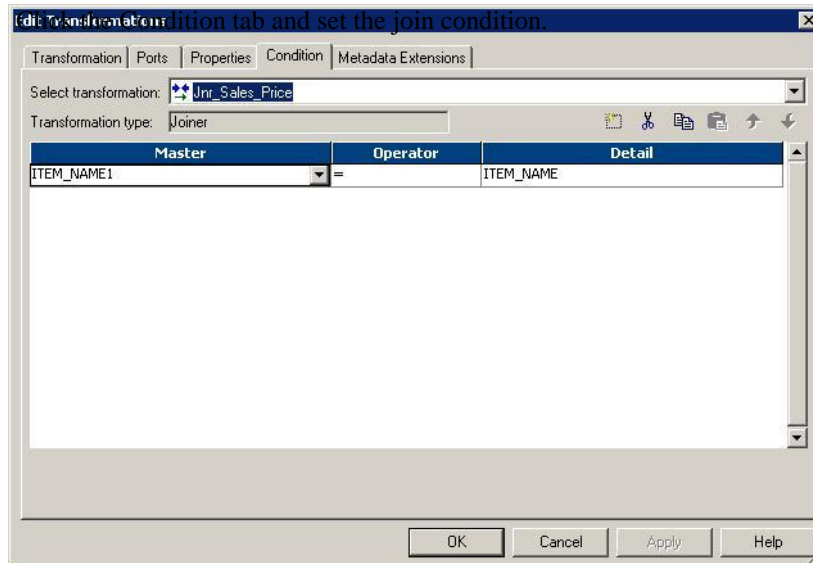
6. Click any box in the M column to switch the master/detail relationship for the sources.

Tip: To improve performance for an unsorted Joiner transformation, use the source with fewer rows as the master source. To improve performance for a sorted Joiner transformation, use the source with fewer duplicate key values as the master.

7. Add default values for specific ports.

Some ports are likely to contain null values, since the fields in one of the sources may be empty. You can specify a default value if the target database does not handle NULLs.

8. Click the Condition tab and set the join condition.



9. Click the Add button to add a condition. You can add multiple conditions.

The master and detail ports must have matching datatypes. The Joiner transformation only supports equivalent (=) joins.

10. Click the Properties tab and configure properties for the transformation.

CHAPTER 10

Lookup Transformation

This chapter includes the following topics:

- φ Overview
- φ Lookup Source Types
- φ Connected and Unconnected Lookups
- φ Lookup Components
- φ Lookup Properties
- φ Lookup Query
- φ Lookup Condition
- φ Lookup Caches
- φ Configuring Unconnected Lookup Transformations
- φ Creating a Lookup Transformation

Overview

Transformation type:

Passive
Connected/Unconnected

Use a Lookup transformation in a mapping to look up data in a flat file, relational table, view, or synonym. You can import a lookup definition from any flat file or relational database to which both the PowerCenter Client and Integration Service can connect. You can also create a lookup definition from a source qualifier. You can use multiple Lookup transformations in a mapping.

The Integration Service queries the lookup source based on the lookup ports in the transformation and a lookup condition. The Lookup transformation returns the result of the lookup to the target or another transformation.

Perform the following tasks with a Lookup transformation:

- φ **Get a related value.** Retrieve a value from the lookup table based on a value in the source. For example, the source has an employee ID. Retrieve the employee name from the lookup table.
- φ **Perform a calculation.** Retrieve a value from a lookup table and use it in a calculation. For example, retrieve a sales tax percentage, calculate a tax, and return the tax to a target.
- φ **Update slowly changing dimension tables.** Determine whether rows exist in a target.

Configure the Lookup transformation to perform the following types of lookups:

- ϕ **Relational or flat file lookup.** Perform a lookup on a flat file or a relational table. When you create a Lookup transformation using a relational table as the lookup source, you can connect to the lookup source using ODBC and import the table definition as the structure for the Lookup transformation. When you create a Lookup transformation using a flat file as a lookup source, the Designer invokes the Flat File Wizard.
- ϕ **Pipeline lookup.** Perform a lookup on application sources such as a JMS, MSMQ, or SAP. Drag the source into the mapping and associate the Lookup transformation with the source qualifier. Configure partitions to improve performance when the Integration Service retrieves source data for the lookup cache.
- ϕ **Connected or unconnected lookup.** A connected Lookup transformation receives source data, performs a lookup, and returns data to the pipeline. An unconnected Lookup transformation is not connected to a source or target. A transformation in the pipeline calls the Lookup transformation with a :LKP expression. The unconnected Lookup transformation returns one column to the calling transformation.
- ϕ **Cached or uncached lookup.** Cache the lookup source to improve performance. If you cache the lookup source, you can use a dynamic or static cache. By default, the lookup cache remains static and does not change during the session. With a dynamic cache, the Integration Service inserts or updates rows in the cache. When you cache the target table as the lookup source, you can look up values in the cache to determine if the values exist in the target. The Lookup transformation marks rows to insert or update the target.

Lookup Source Types

When you create a Lookup transformation, you can choose a relational table, flat file, or a source qualifier as the lookup source.

Relational Lookups

When you create a Lookup transformation using a relational table as a lookup source, you can connect to the lookup source using ODBC and import the table definition as the structure for the Lookup transformation.

Use the following options with relational lookups:

- ϕ Override the default SQL statement to add a WHERE clause or to query multiple tables.
- ϕ Sort null data high or low, based on database support.
- ϕ Perform case-sensitive comparisons based on the database support.

Flat File Lookups

When you create a Lookup transformation using a flat file as a lookup source, select a flat file definition in the repository or import the source when you create the transformation. When you import a flat file lookup source, the Designer invokes the Flat File Wizard.

Use the following options with flat file lookups:

- ϕ Use indirect files as lookup sources by configuring a file list as the lookup file name.
- ϕ Use sorted input for the lookup.
- ϕ Sort null data high or low.
- ϕ Use case-sensitive string comparison with flat file lookups.

Using Sorted Input

When you configure a flat file Lookup transformation for sorted input, the condition columns must be grouped. If the condition columns are not grouped, the Integration Service cannot cache the lookup and fails the session. For optimal caching performance, sort the condition columns.

For example, a Lookup transformation has the following condition:

```
OrderID = OrderID1
CustID = CustID1
```

In the following flat file lookup source, the keys are grouped, but not sorted. The Integration Service can cache the data, but performance may not be optimal.

OrderID	CustID	ItemNo.	ItemDesc	Comments
1001	CA502	F895S	Flashlight	Key data is grouped, but not sorted. CustID is out of order within OrderID.
1001	CA501	C530S	Compass	
1001	CA501	T552T	Tent	
1005	OK503	S104E	Safety Knife	Key data is grouped, but not sorted. OrderID is out of order.
1003	CA500	F304T	First Aid Kit	
1003	TN601	R938M	Regulator System	

The keys are not grouped in the following flat file lookup source. The Integration Service cannot cache the data and fails the session.

OrderID	CustID	ItemNo.	ItemDesc	Comments
1001	CA501	T552T	Tent	Key data for CustID is not grouped.
1001	CA501	C530S	Compass	
1005	OK503	S104E	Safety Knife	
1003	TN601	R938M	Regulator System	
1003	CA500	F304T	First Aid Kit	
1001	CA502	F895S	Flashlight	

If you choose sorted input for indirect files, the range of data must not overlap in the files.

Pipeline Lookups

Create a pipeline Lookup transformation to perform a lookup on an application source that is not a relational table or flat file. A pipeline Lookup transformation has a source qualifier as the lookup source. The source qualifier can represent any type of source definition, including JMS and MSMQ. The source definition cannot have more than one group.

When you configure a pipeline Lookup transformation, the lookup source and source qualifier are in a different pipeline from the Lookup transformation. The source and source qualifier are in a partial pipeline that contains no target. The Integration Service reads the source data in this pipeline and passes the data to the Lookup transformation to create the cache. You can create multiple partitions in the partial pipeline to improve performance.

To improve performance when processing relational or flat file lookup sources, create a pipeline Lookup transformation instead of a relational or flat file Lookup transformation. You can create partitions to process the lookup source and pass it to the Lookup transformation.

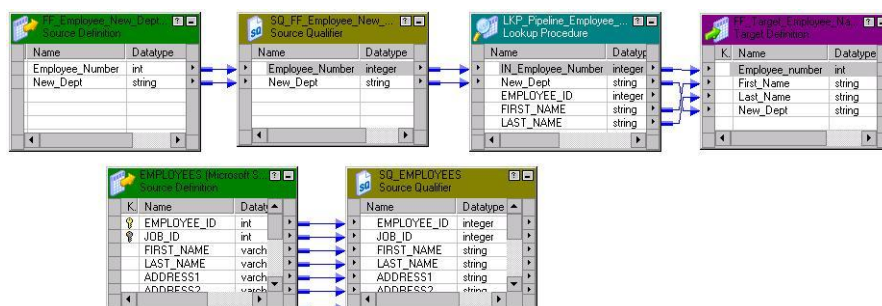
Create a connected or unconnected pipeline Lookup transformation.

Configuring a Pipeline Lookup Transformation in a Mapping

A mapping that contains a pipeline Lookup transformation includes a partial pipeline that contains the lookup source and source qualifier. The partial pipeline does not include a target. The Integration Service retrieves the lookup source data in this pipeline and passes the data to the lookup cache.

The partial pipeline is in a separate target load order group in session properties. You can create multiple partitions in the pipeline to improve performance. You can not configure the target load order with the partial pipeline.

The following mapping shows a mapping that contains a pipeline Lookup transformation and the partial pipeline that processes the lookup source:



The mapping contains the following objects:

- φ The lookup source definition and source qualifier are in a separate pipeline. The Integration Service creates a lookup cache after it processes the lookup source data in the pipeline.
- φ A flat file source contains new department names by employee number.
- φ The pipeline Lookup transformation receives Employee_Number and New_Dept from the source file. The pipeline Lookup performs a lookup on Employee_ID in the lookup cache. It retrieves the employee first and last name from the lookup cache.
- φ A flat file target receives the Employee_ID, First_Name, Last_Name, and New_Dept from the Lookup transformation.

Connected and Unconnected Lookups

You can configure a connected Lookup transformation to receive input directly from the mapping pipeline, or you can configure an unconnected Lookup transformation to receive input from the result of an expression in another transformation.

The following table lists the differences between connected and unconnected lookups:

Connected Lookup	Unconnected Lookup
Receives input values directly from the pipeline.	Receives input values from the result of a :LKP expression in another transformation.
Use a dynamic or static cache.	Use a static cache.
Cache includes the lookup source columns in the lookup condition and the lookup source columns that are output ports.	Cache includes all lookup/output ports in the lookup condition and the lookup/return port.
Can return multiple columns from the same row or insert into the dynamic lookup cache.	Designate one return port (R). Returns one column from each row.

Connected Lookup	Unconnected Lookup
If there is no match for the lookup condition, the Integration Service returns the default value for all output ports. If you configure dynamic caching, the Integration Service inserts rows into the cache or leaves it unchanged.	If there is no match for the lookup condition, the Integration Service returns NULL.
If there is a match for the lookup condition, the Integration Service returns the result of the lookup condition for all lookup/output ports. If you configure dynamic caching, the Integration Service either updates the row in the cache or leaves the row unchanged.	If there is a match for the lookup condition, the Integration Service returns the result of the lookup condition into the return port.
Pass multiple output values to another transformation. Link lookup/output ports to another transformation.	Pass one output value to another transformation. The lookup/output/return port passes the value to the transformation calling :LKP expression.
Supports user-defined default values.	Does not support user-defined default values.

Connected Lookup Transformation

The following steps describe how the Integration Service processes a connected Lookup transformation:

1. A connected Lookup transformation receives input values directly from another transformation in the pipeline.
2. For each input row, the Integration Service queries the lookup source or cache based on the lookup ports and the condition in the transformation.
3. If the transformation is uncached or uses a static cache, the Integration Service returns values from the lookup query.

If the transformation uses a dynamic cache, the Integration Service inserts the row into the cache when it does not find the row in the cache. When the Integration Service finds the row in the cache, it updates the row in the cache or leaves it unchanged. It flags the row as insert, update, or no change.

4. The Integration Service passes return values from the query to the next transformation.

If the transformation uses a dynamic cache, you can pass rows to a Filter or Router transformation to filter new rows to the target.

Note: This chapter discusses connected Lookup transformations unless otherwise specified.

Unconnected Lookup Transformation

An unconnected Lookup transformation receives input values from the result of a :LKP expression in another transformation. You can call the Lookup transformation more than once in a mapping.

A common use for unconnected Lookup transformations is to update slowly changing dimension tables. For more information about slowly changing dimension tables, visit the Informatica Knowledge Base at <http://my.informatica.com>.

The following steps describe the way the Integration Service processes an unconnected Lookup transformation:

1. An unconnected Lookup transformation receives input values from the result of a :LKP expression in another transformation, such as an Update Strategy transformation.
2. The Integration Service queries the lookup source or cache based on the lookup ports and condition in the transformation.
3. The Integration Service returns one value into the return port of the Lookup transformation.
4. The Lookup transformation passes the return value into the :LKP expression.

Lookup Components

Define the following components when you configure a Lookup transformation in a mapping:

- ϕ Lookup source
- ϕ Ports
- ϕ Properties
- ϕ Condition

Lookup Source

Use a flat file, relational table, or source qualifier for a lookup source. When you create a Lookup transformation, you can create the lookup source from the following locations:

- ϕ Relational source or target definition in the repository
- ϕ Flat file source or target definition in the repository
- ϕ Table or file that the Integration Service and PowerCenter Client machine can connect to
- ϕ Source qualifier definition in a mapping

The lookup table can be a single table, or you can join multiple tables in the same database using a lookup SQL override. The Integration Service queries the lookup table or an in-memory cache of the table for all incoming rows into the Lookup transformation.

The Integration Service can connect to a lookup table using ODBC or native drivers. Configure native drivers for optimal performance.

Indexes and a Lookup Table

If you have privileges to modify the database containing a lookup table, you can improve lookup initialization time by adding an index to the lookup table. You can improve performance for very large lookup tables. Since the Integration Service queries, sorts, and compares values in lookup columns, the index needs to include every column in a lookup condition.

You can improve performance by indexing the following types of lookup:

- ϕ **Cached lookups.** You can improve performance by indexing the columns in the lookup ORDER BY. The session log contains the ORDER BY clause.
- ϕ **Uncached lookups.** Because the Integration Service issues a SELECT statement for each row passing into the Lookup transformation, you can improve performance by indexing the columns in the lookup condition.

Lookup Ports

The Ports tab contains input and output ports. The Ports tab also includes lookup ports that represent columns of data to return from the lookup source. An unconnected Lookup transformation returns one column of data to the calling transformation in this port. An unconnected Lookup transformation has one return port.

The following table describes the port types in a Lookup transformation:

Ports	Type of Lookup	Description
I	Connected Unconnected	Input port. Create an input port for each lookup port you want to use in the lookup condition. You must have at least one input or input/output port in each Lookup transformation.
O	Connected Unconnected	Output port. Create an output port for each lookup port you want to link to another transformation. You can designate both input and lookup ports as output ports. For connected lookups, you must have at least one output port. For unconnected lookups, select a lookup port as a return port (R) to pass a return value.

Ports	Type of Lookup	Description
L	Connected Unconnected	Lookup port. The Designer designates each column in the lookup source as a lookup (L) and output port (O).
R	Unconnected	Return port. Use only in unconnected Lookup transformations. Designates the column of data you want to return based on the lookup condition. You can designate one lookup port as the return port.

The Lookup transformation also enables an associated ports property that you configure when you use a dynamic cache. The associated port is the input port that contains the data to update the lookup cache.

Use the following guidelines to configure lookup ports:

- ϕ If you delete lookup ports from a flat file lookup, the session fails.
- ϕ You can delete lookup ports from a relational lookup if the mapping does not use the lookup port. This reduces the amount of memory the Integration Service needs to run the session.

Lookup Properties

On the Properties tab, configure properties such as an SQL override for relational lookups, the lookup source name caching properties.

RELATED TOPICS:

- ϕ “Lookup Properties”

Lookup Condition

On the Condition tab, enter the condition or conditions you want the Integration Service to use to find data in the lookup source.

RELATED TOPICS:

- ϕ “Lookup Condition”

Lookup Properties

Configure the lookup properties such as caching and multiple matches on the Lookup Properties tab. Configure the lookup condition or the SQL statements to query the lookup table. You can also change the Lookup table name.

When you create a mapping, you configure the properties for each Lookup transformation. When you create a session, you can override properties such as the index and data cache size for each transformation.

The following table describes the Lookup transformation properties:

Option	Lookup Type	Description
Lookup SQL Override	Relational	Overrides the default SQL statement to query the lookup table. Specifies the SQL statement you want the Integration Service to use for querying lookup values. Use only with the lookup cache enabled.
Lookup Table Name	Pipeline Relational	The name of the table or source qualifier from which the transformation looks up and caches values. When you create the Lookup transformation, choose a source, target, or source qualifier as the lookup source. You can also import a table, view, or synonym from another database when you create the Lookup transformation. If you enter a lookup SQL override, you do not need to enter the Lookup Table Name.
Lookup Source Filter	Relational	Restricts the lookups the Integration Service performs based on the value of data in Lookup transformation input ports.
Lookup Caching Enabled	Flat File Pipeline Relational	Indicates whether the Integration Service caches lookup values during the session. When you enable lookup caching, the Integration Service queries the lookup source once, caches the values, and looks up values in the cache during the session. Caching the lookup values can improve session performance. When you disable caching, each time a row passes into the transformation, the Integration Service issues a select statement to the lookup source for lookup values. Note: The Integration Service always caches flat file and pipeline lookups.
Lookup Policy on Multiple Match	Flat File Pipeline Relational	Determines which rows that the Lookup transformation returns when it finds multiple rows that match the lookup condition. You can select the first or last row returned from the cache or lookup source, or report an error. Or, you can allow the Lookup transformation to use any value. When you configure the Lookup transformation to return any matching value, the transformation returns the first value that matches the lookup condition. It creates an index based on the key ports instead of all Lookup transformation ports. If you do not enable the Output Old Value On Update option, the Lookup Policy On Multiple Match option is set to Report Error for dynamic lookups.
Lookup Condition	Flat File Pipeline Relational	Displays the lookup condition you set in the Condition tab.
Connection Information	Relational	Specifies the database containing the lookup table. You can define the database in the mapping, session, or parameter file: - Mapping. Select the connection object. You can also specify the database connection type. Type <i>Relational</i> : before the connection name if it is a relational connection. Type <i>Application</i> : before the connection name if it is an application connection. - Session. Use the \$Source or \$Target connection variable. If you use one of these variables, the lookup table must reside in the source or target database. Specify the database connection in the session properties for each variable. - Parameter file. Use the session parameter \$DBConnectionName or \$AppConnectionName, and define it in the parameter file. By default, the Designer specifies \$Source if you choose an existing source table and \$Target if you choose an existing target table when you create the Lookup transformation. You can override these values in the session properties. The Integration Service fails the session if it cannot determine the type of database connection.

Option	Lookup Type	Description
Source Type	Flat File Pipeline Relational	Indicates that the Lookup transformation reads values from a relational table, flat file, or source qualifier.
Tracing Level	Flat File Pipeline Relational	Sets the amount of detail included in the session log.
Lookup Cache Directory Name	Flat File Pipeline Relational	Specifies the directory used to build the lookup cache files when you configure the Lookup transformation to cache the lookup source. Also used to save the persistent lookup cache files when you select the Lookup Persistent option. By default, the Integration Service uses the \$PMCacheDir directory configured for the Integration Service.
Lookup Cache Persistent	Flat File Pipeline Relational	Indicates whether the Integration Service uses a persistent lookup cache, which consists of at least two cache files. If a Lookup transformation is configured for a persistent lookup cache and persistent lookup cache files do not exist, the Integration Service creates the files during the session. Use only with the lookup cache enabled.
Lookup Data Cache Size Lookup Index Cache Size	Flat File Pipeline Relational	Default is Auto. Indicates the maximum size the Integration Service allocates to the data cache and the index in memory. You can configure a numeric value, or you can configure the Integration Service to determine the cache size at run-time. If you configure the Integration Service to determine the cache size, you can also configure a maximum amount of memory for the Integration Service to allocate to the cache. If the Integration Service cannot allocate the configured amount of memory when initializing the session, it fails the session. When the Integration Service cannot store all the data cache data in memory, it pages to disk. Use with the lookup cache enabled.
Dynamic Lookup Cache	Flat File Pipeline Relational	Indicates to use a dynamic lookup cache. Inserts or updates rows in the lookup cache as it passes rows to the target table. Use only with the lookup cache enabled.
Output Old Value On Update	Flat File Pipeline Relational	Use with dynamic caching enabled. When you enable this property, the Integration Service outputs old values out of the lookup/output ports. When the Integration Service updates a row in the cache, it outputs the value that existed in the lookup cache before it updated the row based on the input data. When the Integration Service inserts a new row in the cache, it outputs null values. When you disable this property, the Integration Service outputs the same values out of the lookup/output and input/output ports. This property is enabled by default.
Cache File Name Prefix	Flat File Pipeline Relational	Use with persistent lookup cache. Specifies the file name prefix to use with persistent lookup cache files. The Integration Service uses the file name prefix as the file name for the persistent cache files it saves to disk. Only enter the prefix. Do not enter .idx or .dat. You can enter a parameter or variable for the file name prefix. Use any parameter or variable type that you can define in the parameter file. If the named persistent cache files exist, the Integration Service builds the memory cache from the files. If the named persistent cache files do not exist, the Integration Service rebuilds the persistent cache files.
Recache From Lookup Source	Flat File Pipeline Relational	Use with the lookup cache enabled. When selected, the Integration Service rebuilds the lookup cache from the lookup source when it first calls the Lookup transformation instance. If you use a persistent lookup cache, it rebuilds the persistent cache files before using the cache. If you do not use a persistent lookup cache, it rebuilds the lookup cache in memory before using the cache.

Option	Lookup Type	Description
Insert Else Update	Flat File Pipeline Relational	Use with dynamic caching enabled. Applies to rows entering the Lookup transformation with the row type of insert. When enabled, the Integration Service inserts new rows in the cache and updates existing rows. When disabled, the Integration Service does not update existing rows.
Update Else Insert	Flat File Pipeline Relational	Use with dynamic caching enabled. Applies to rows entering the Lookup transformation with the row type of update. When enabled, the Integration Service updates existing rows, and inserts a new row if it is new. When disabled, the Integration Service does not insert new rows.
Datetime Format	Flat File	Click the Open button to select a datetime format. Define the format and field width. Milliseconds, microseconds, or nanoseconds formats have a field width of 29. If you do not select a datetime format for a port, you can enter any datetime format. Default is MM/DD/YYYY HH24:MI:SS. The Datetime format does not change the size of the port.
Thousand Separator	Flat File	If you do not define a thousand separator for a port, the Integration Service uses the properties defined here. You can choose no separator, a comma, or a period. Default is no separator.
Decimal Separator	Flat File	If you do not define a decimal separator for a particular field in the lookup definition or on the Ports tab, the Integration Service uses the properties defined here. You can choose a comma or a period decimal separator. Default is period.
Case-Sensitive String Comparison	Flat File Pipeline	The Integration Service uses case-sensitive string comparisons when performing lookups on string columns. For relational lookups, the case-sensitive comparison is based on the database support.
Null Ordering	Flat File Pipeline	Determines how the Integration Service orders null values. You can choose to sort null values high or low. By default, the Integration Service sorts null values high. This overrides the Integration Service configuration to treat nulls in comparison operators as high, low, or null. For relational lookups, null ordering is based on the database default value.
Sorted Input	Flat File Pipeline	Indicates whether or not the lookup file data is sorted. This increases lookup performance for file lookups. If you enable sorted input, and the condition columns are not grouped, the Integration Service fails the session. If the condition columns are grouped, but not sorted, the Integration Service processes the lookup as if you did not configure sorted input.
Lookup Source is Static	Flat File Pipeline Relational	The lookup source does not change in a session.

Option	Lookup Type	Description
Pre-build Lookup Cache		<p>Allow the Integration Service to build the lookup cache before the Lookup transformation requires the data.</p> <ul style="list-style-type: none"> - Auto. - Always allowed. The Integration Service builds the lookup cache before the Lookup transformation receives the first source row. - Always disallowed. The Integration Service builds the lookup cache when the Lookup transformation receives the source row.
Subsecond Precision	Relational	<p>Specifies the subsecond precision for datetime ports.</p> <p>For relational lookups, you can change the precision for databases that have an editable scale for datetime data. You can change subsecond precision for Oracle Timestamp, Informix Datetime, and Teradata Timestamp datatypes.</p> <p>Enter a positive integer value from 0 to 9. Default is 6 microseconds.</p> <p>If you enable pushdown optimization, the database returns the complete datetime value, regardless of the subsecond precision setting.</p>

Configuring Lookup Properties in a Session

When you configure a session, you can configure lookup properties that are unique to sessions:

- φ **Flat file lookups.** Configure lookup location information, such as the source file directory, file name, and the file type.
- φ **Relational lookups.** You can define \$Source and \$Target variables in the session properties. You can also override connection information to use the \$DBConnectionName or \$AppConnectionName session parameter.
- φ **Pipeline lookups.** Configure the lookup source file properties such as the source file directory, file name, and the file type. If the source is a relational table or application source, configure the connection information.

Configuring Flat File Lookups in a Session

When you configure a flat file lookup in a session, configure the lookup source file properties on the Transformation View of the Mapping tab. Choose the Lookup transformation and configure the flat file properties in the session properties for the transformation.

Table 14-1 describes the session properties you configure for flat file lookups:

Table 14-1. Session Properties for Flat File Lookups

Property	Description
Lookup Source File Directory	<p>Enter the directory name. By default, the Integration Service looks in the process variable directory, \$PMLookupFileDir, for lookup files.</p> <p>You can enter the full path and file name. If you specify both the directory and file name in the Lookup Source Filename field, clear this field. The Integration Service concatenates this field with the Lookup Source Filename field when it runs the session.</p> <p>You can also enter the \$InputFileName session parameter to configure the file name.</p>

Table 14-1. Session Properties for Flat File Lookups

Property	Description
Lookup Source Filename	<p>Name of the lookup file. If you use an indirect file, enter the name of the indirect file you want the Integration Service to read.</p> <p>You can also enter the lookup file parameter, <code>\$LookupFileName</code>, to change the name of the lookup file for the session.</p> <p>If you configure both the directory and file name in the Source File Directory field, clear Lookup Source Filename. The Integration Service concatenates Lookup Source Filename with the Lookup Source File Directory field for the session. For example, the Lookup Source File Directory field contains "C:\lookup_data\" and the Lookup Source Filename field contains "filename.txt." When the Integration Service begins the session, it looks for "C:\lookup_data\filename.txt."</p>
Lookup Source Filetype	<p>Indicates whether the lookup source file contains the source data or a list of files with the same file properties. Choose Direct if the lookup source file contains the source data. Choose Indirect if the lookup source file contains a list of files.</p> <p>When you select Indirect, the Integration Service creates one cache for all files. If you use sorted input with indirect files, verify that the range of data in the files do not overlap. If the range of data overlaps, the Integration Service processes the lookup as an unsorted lookup source.</p>

Configuring Relational Lookups in a Session

When you configure a relational lookup in a session, configure the connection for the lookup database on the Transformation View of the Mapping tab. Choose the Lookup transformation and configure the connection in the session properties for the transformation.

Choose from the following options to configure a connection for a relational Lookup transformation:

- ϕ Choose a relational or application connection.
- ϕ Configure a database connection using the `$Source` or `$Target` connection variable.
- ϕ Configure the session parameter `$DBConnectionName` or `$AppConnectionName`, and define the session parameter in a parameter file.

Configuring Pipeline Lookups in a Session

When you configure a pipeline Lookup in a session, configure the location of lookup source file or the connection for the lookup table on the Sources node of the Mapping tab. Choose the Source Qualifier that represents the lookup source.

Lookup Query

The Integration Service queries the lookup based on the ports and properties you configure in the Lookup transformation. The Integration Service runs a default SQL statement when the first row enters the Lookup transformation. If you use a relational lookup or a pipeline lookup against a relational table, you can customize the default query with the Lookup SQL Override property.

You can restrict the rows that a Lookup transformation retrieves from the source when it builds the lookup cache. Configure the Lookup Source Filter.

Default Lookup Query

The default lookup query contains the following statements:

- φ **SELECT.** The SELECT statement includes all the lookup ports in the mapping. You can view the SELECT statement by generating SQL using the Lookup SQL Override property. Do not add or delete any columns from the default SQL statement.
- φ **ORDER BY.** The ORDER BY clause orders the columns in the same order they appear in the Lookup transformation. The Integration Service generates the ORDER BY clause. You cannot view this when you generate the default SQL using the Lookup SQL Override property.

Overriding the Lookup Query

The lookup SQL override is similar to entering a custom query in a Source Qualifier transformation. You can override the lookup query for a relational lookup. You can enter the entire override, or you can generate and edit the default SQL statement. When the Designer generates the default SQL statement for the lookup SQL override, it includes the lookup/output ports in the lookup condition and the lookup/return port.

Override the lookup query in the following circumstances:

- φ **Override the ORDER BY clause.** Create the ORDER BY clause with fewer columns to increase performance. When you override the ORDER BY clause, you must suppress the generated ORDER BY clause with a comment notation.

Note: If you use pushdown optimization, you cannot override the ORDER BY clause or suppress the generated ORDER BY clause with a comment notation.
- φ **A lookup table name or column names contains a reserved word.** If the table name or any column name in the lookup query contains a reserved word, you must ensure that all reserved words are enclosed in quotes.
- φ **Use parameters and variables.** Use parameters and variables when you enter a lookup SQL override. Use any parameter or variable type that you can define in the parameter file. You can enter a parameter or variable within the SQL statement, or you can use a parameter or variable as the SQL query. For example, you can use a session parameter, \$ParamMyLkpOverride, as the lookup SQL query, and set \$ParamMyLkpOverride to the SQL statement in a parameter file.

The Designer cannot expand parameters and variables in the query override and does not validate it when you use a parameter or variable. The Integration Service expands the parameters and variables when you run the session.

- φ **A lookup column name contains a slash (/) character.** When generating the default lookup query, the Designer and Integration Service replace any slash character (/) in the lookup column name with an underscore character. To query lookup column names containing the slash character, override the default lookup query, replace the underscore characters with the slash character, and enclose the column name in double quotes.
- φ **Add a WHERE clause.** Use a lookup SQL override to add a WHERE clause to the default SQL statement. You might want to use the WHERE clause to reduce the number of rows included in the cache. When you add a WHERE clause to a Lookup transformation using a dynamic cache, use a Filter transformation before the Lookup transformation to pass rows into the dynamic cache that match the WHERE clause.

Note: The session fails if you include large object ports in a WHERE clause.
- φ **Other.** Use a lookup SQL override if you want to query lookup data from multiple lookups or if you want to modify the data queried from the lookup table before the Integration Service caches the lookup rows. For example, use TO_CHAR to convert dates to strings.

Overriding the ORDER BY Clause

By default, the Integration Service generates an ORDER BY clause for a cached lookup. The ORDER BY clause contains all lookup ports. To increase performance, you can suppress the default ORDER BY clause and enter an override ORDER BY with fewer columns.

Note: If you use pushdown optimization, you cannot override the ORDER BY clause or suppress the generated ORDER BY clause with a comment notation.

The Integration Service always generates an ORDER BY clause, even if you enter one in the override. Place two dashes '--' after the ORDER BY override to suppress the generated ORDER BY clause. For example, a Lookup transformation uses the following lookup condition:

```
ITEM_ID = IN_ITEM_ID
PRICE <= IN_PRICE
```

The Lookup transformation includes three lookup ports used in the mapping, ITEM_ID, ITEM_NAME, and PRICE. When you enter the ORDER BY clause, enter the columns in the same order as the ports in the lookup condition. You must also enclose all database reserved words in quotes. Enter the following lookup query in the lookup SQL override:

```
SELECT ITEMS_DIM.ITEM_NAME, ITEMS_DIM.PRICE, ITEMS_DIM.ITEM_ID FROM ITEMS_DIM ORDER BY
ITEMS_DIM.ITEM_ID, ITEMS_DIM.PRICE --
```

To override the default ORDER BY clause for a relational lookup, complete the following steps:

1. Generate the lookup query in the Lookup transformation.
2. Enter an ORDER BY clause that contains the condition ports in the same order they appear in the Lookup condition.
3. Place two dashes '--' as a comment notation after the ORDER BY clause to suppress the ORDER BY clause that the Integration Service generates.

If you override the lookup query with an ORDER BY clause without adding comment notation, the lookup fails.

Note: Sybase has a 16 column ORDER BY limitation. If the Lookup transformation has more than 16 lookup/output ports including the ports in the lookup condition, override the ORDER BY clause or use multiple Lookup transformations to query the lookup table.

Reserved Words

If any lookup name or column name contains a database reserved word, such as MONTH or YEAR, the session fails with database errors when the Integration Service executes SQL against the database. You can create and maintain a reserved words file, reswords.txt, in the Integration Service installation directory. When the Integration Service initializes a session, it searches for reswords.txt. If the file exists, the Integration Service places quotes around matching reserved words when it executes SQL against the database.

You may need to enable some databases, such as Microsoft SQL Server and Sybase, to use SQL-92 standards regarding quoted identifiers. Use connection environment SQL to issue the command. For example, with Microsoft SQL Server, use the following command:

```
SET QUOTED_IDENTIFIER ON
```

Note: The reserved words file, reswords.txt, is a file that you create and maintain in the Integration Service installation directory. The Integration Service searches this file and places quotes around reserved words when it executes SQL against source, target, and lookup databases.

Guidelines to Overriding the Lookup Query

Use the following guidelines when you override the lookup SQL query:

- ϕ You can only override the lookup SQL query for relational lookups.
- ϕ Configure the Lookup transformation for caching. If you do not enable caching, the Integration Service does not recognize the override.
- ϕ Generate the default query, and then configure the override. This helps ensure that all the lookup/output ports are included in the query. If you add or subtract ports from the SELECT statement, the session fails.
- ϕ Add a source lookup filter to filter the rows that are added to the lookup cache. This ensures the Integration Service only inserts rows in the dynamic cache and target table that match the WHERE clause.
- ϕ To share the cache, use the same lookup SQL override for each Lookup transformation.

- ϕ If you override the ORDER BY clause, the session fails if the ORDER BY clause does not contain the condition ports in the same order they appear in the Lookup condition or if you do not suppress the generated ORDER BY clause with the comment notation.
- ϕ If you use pushdown optimization, you cannot override the ORDER BY clause or suppress the generated ORDER BY clause with comment notation.
- ϕ If the table name or any column name in the lookup query contains a reserved word, you must enclose all reserved words in quotes.

Steps to Overriding the Lookup Query

Use the following steps to override the default lookup SQL query.

To override the default lookup query:

1. On the Properties tab, open the SQL Editor from within the Lookup SQL Override field.
2. Click Generate SQL to generate the default SELECT statement. Enter the lookup SQL override.
3. Connect to a database, and then click Validate to test the lookup SQL override.
4. Click OK to return to the Properties tab.

Filtering Lookup Source Rows

You can limit the number of lookups that the Integration Service performs on a lookup source table. Add a Lookup Source Filter statement to the Lookup transformation properties. You can configure a Lookup Source Filter for relational Lookup transformations.

When you configure a Lookup Source Filter, the Integration Service performs lookups based on the results of the filter statement.

For example, you might need to retrieve the last name of every employee with an ID greater than 510. Configure a lookup source filter on the EmployeeID column:

```
EmployeeID >= 510
```

EmployeeID is an input port in the Lookup transformation. When the Integration Service reads the source row, it performs a lookup on the cache when EmployeeID has a value greater than 510. When EmployeeID is less than or equal to 510, the Lookup transformation does not retrieve the last name.

When you add a lookup source filter to the Lookup query for a session configured for pushdown optimization, the Integration Service creates a view to represent the SQL override. It runs an SQL query against this view to push the transformation logic to the database.

To configure a lookup source filter:

1. In the Mapping Designer or Transformation Developer, open the Lookup transformation.
2. Select the Properties tab.
3. Click the Open button in the Lookup Source Filter field.
4. In the SQL Editor, select the input ports to filter and enter a filter conditions.

Do not include the keyword WHERE in the filter condition. Enclose string mapping parameters and variables in string identifiers.
5. Select the ODBC data source containing the source included in the query.
6. Enter the user name and password to connect to this database.
7. Click Validate.

The Designer runs the query and reports whether its syntax is correct.

Lookup Condition

The Integration Service finds data in the lookup source with a lookup condition. The lookup condition is similar to the WHERE clause in an SQL query. When you configure a lookup condition in a Lookup transformation, you compare the value of one or more columns in the source data with values in the lookup source or cache.

For example, the source data contains an employee_number. The lookup source table contains employee_ID, first_name, and last_name. You configure the following lookup condition:

```
employee_ID = employee_number
```

For each employee_number, the Integration Service returns the employee_ID, last_name, and first_name column from the lookup source.

The Integration Service can return more than one row from the lookup source. You configure the following lookup condition:

```
employee_ID > employee_number
```

The Integration Service returns rows for all employee_ID numbers greater than the source employee number.

Use the following guidelines when you enter a condition for a Lookup transformation:

- ϕ The datatypes for the columns in a lookup condition must match.
- ϕ You must enter a lookup condition in all Lookup transformations.
- ϕ Use one input port for each lookup port in the lookup condition. Use the same input port in more than one condition in a transformation.
- ϕ When you enter multiple conditions, the Integration Service evaluates each condition as an AND, not an OR. The Integration Service returns rows that match all the conditions you configure.
- ϕ If you include multiple conditions, enter the conditions in the following order to optimize lookup performance:
 - ☐ Equal to (=)
 - ☐ Less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=)
 - ☐ Not equal to (!=)
- ϕ The Integration Service matches null values. For example, if an input lookup condition column is NULL, the Integration Service evaluates the NULL equal to a NULL in the lookup.
- ϕ If you configure a flat file lookup for sorted input, the Integration Service fails the session if the condition columns are not grouped. If the columns are grouped, but not sorted, the Integration Service processes the lookup as if you did not configure sorted input.

The Integration Service processes lookup matches differently depending on whether you configure the transformation for a dynamic cache or an uncached or static cache.

Uncached or Static Cache

Use the following guidelines when you configure a Lookup transformation that has a static lookup cache or an uncached lookup source:

- ϕ Use the following operators when you create the lookup condition:

```
=, >, <, >=, <=, !=
```

If you include more than one lookup condition, place the conditions in the following order to optimize lookup performance:

- ☐ Equal to (=)
- ☐ Less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=)
- ☐ Not equal to (!=)

For example, create the following lookup condition:

```
ITEM_ID = IN_ITEM_ID
PRICE <= IN_PRICE
```

φ The input value must meet all conditions for the lookup to return a value.

The condition can match equivalent values or supply a threshold condition. For example, you might look for customers who do not live in California, or employees whose salary is greater than \$30,000. Depending on the nature of the source and condition, the lookup might return multiple values.

Dynamic Cache

If you configure a Lookup transformation to use a dynamic cache, you can use only the equality operator (=) in the lookup condition.

Handling Multiple Matches

Lookups find a value based on the conditions you set in the Lookup transformation. If the lookup condition is not based on a unique key, or if the lookup source is denormalized, the Integration Service might find multiple matches in the lookup source or cache.

You can configure a Lookup transformation to handle multiple matches in the following ways:

φ **Return the first matching value, or return the last matching value.** You can configure the transformation to return the first matching value or the last matching value. The first and last values are the first value and last value found in the lookup cache that match the lookup condition. When you cache the lookup source, the Integration Service generates an ORDER BY clause for each column in the lookup cache to determine the first and last row in the cache. The Integration Service then sorts each lookup source column in ascending order.

The Integration Service sorts numeric columns in ascending numeric order (such as 0 to 10), date/time columns from January to December and from the first of the month to the end of the month, and string columns based on the sort order configured for the session.

φ **Return any matching value.** You can configure the Lookup transformation to return any value that matches the lookup condition. When you configure the Lookup transformation to return any matching value, the transformation returns the first value that matches the lookup condition. It creates an index based on the key ports rather than all Lookup transformation ports. When you use any matching value, performance can improve because the process of indexing rows is simplified.

φ **Return an error.** When the Lookup transformation uses a static cache or no cache, the Integration Service marks the row as an error, writes the row to the session log by default, and increases the error count by one. When the Lookup transformation uses a dynamic cache, the Integration Service fails the session when it encounters multiple matches either while caching the lookup table or looking up values in the cache that contain duplicate keys. Also, if you configure the Lookup transformation to output old values on updates, the Lookup transformation returns an error when it encounters multiple matches.

Lookup Caches

You can configure a Lookup transformation to cache the lookup file or table. The Integration Service builds a cache in memory when it processes the first row of data in a cached Lookup transformation. It allocates memory for the cache based on the amount you configure in the transformation or session properties. The Integration Service stores condition values in the index cache and output values in the data cache. The Integration Service queries the cache for each row that enters the transformation.

The Integration Service also creates cache files by default in the \$PMCacheDir. If the data does not fit in the memory cache, the Integration Service stores the overflow values in the cache files. When the session completes, the Integration Service releases cache memory and deletes the cache files unless you configure the Lookup transformation to use a persistent cache.

When configuring a lookup cache, you can configure the following options:

- ϕ Persistent cache
- ϕ Recache from lookup source
- ϕ Static cache
- ϕ Dynamic cache
- ϕ Shared cache
- ϕ Pre-build lookup cache

Note: You can use a dynamic cache for relational or flat file lookups.

Configuring Unconnected Lookup Transformations

An unconnected Lookup transformation is a Lookup transformation that is not connected to a source or target. Call the lookup from another transformation with a :LKP expression.

You can perform the following tasks when you call a lookup from an expression:

- ϕ Test the results of a lookup in an expression.
- ϕ Filter rows based on the lookup results.
- ϕ Mark rows for update based on the result of a lookup and update slowly changing dimension tables.
- ϕ Call the same lookup multiple times in one mapping.

Complete the following steps to configure an unconnected Lookup transformation:

1. Add input ports to the Lookup transformation.
2. Add the lookup condition to the Lookup transformation.
3. Designate a return value.
4. Configure the lookup expression in another transformation.

Step 1. Add Input Ports

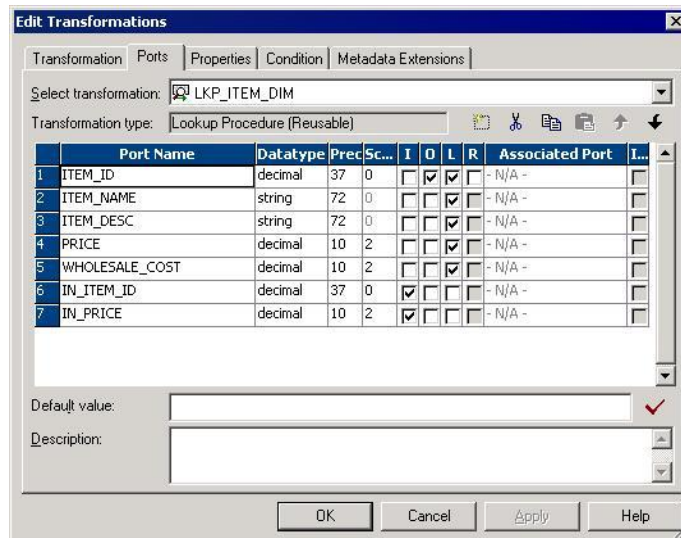
Create an input port in the Lookup transformation for each argument in the :LKP expression.

For each lookup condition you plan to create, you need to add an input port to the Lookup transformation. You can create a different port for each condition, or use the same input port in more than one condition.

For example, a retail store increased prices across all departments during the last month. The accounting department only wants to load rows into the target for items with increased prices.

To accomplish this, complete the following tasks:

- φ Create a lookup condition that compares the ITEM_ID in the source with the ITEM_ID in the target.
- φ Compare the PRICE for each item in the source with the price in the target table.
 - If the item exists in the target table and the item price in the source is less than or equal to the price in the target table, you want to delete the row.
 - If the price in the source is greater than the item price in the target table, you want to update the row.
- φ Create an input port (IN_ITEM_ID) with datatype Decimal (37,0) to match the ITEM_ID and an IN_PRICE input port with Decimal (10,2) to match the PRICE lookup port.



Step 2. Add the Lookup Condition

After you configure the ports, define a lookup condition to compare transformation input values with values in the lookup source or cache. To increase performance, add conditions with an equal sign first.

In this case, add the following lookup condition:

```
ITEM_ID = IN_ITEM_ID
PRICE <= IN_PRICE
```

If the item exists in the mapping source and lookup source and the mapping source price is less than or equal to the lookup price, the condition is true and the lookup returns the values designated by the Return port. If the lookup condition is false, the lookup returns NULL. When you write the update strategy expression, use ISNULL nested in an IIF function to test for null values.

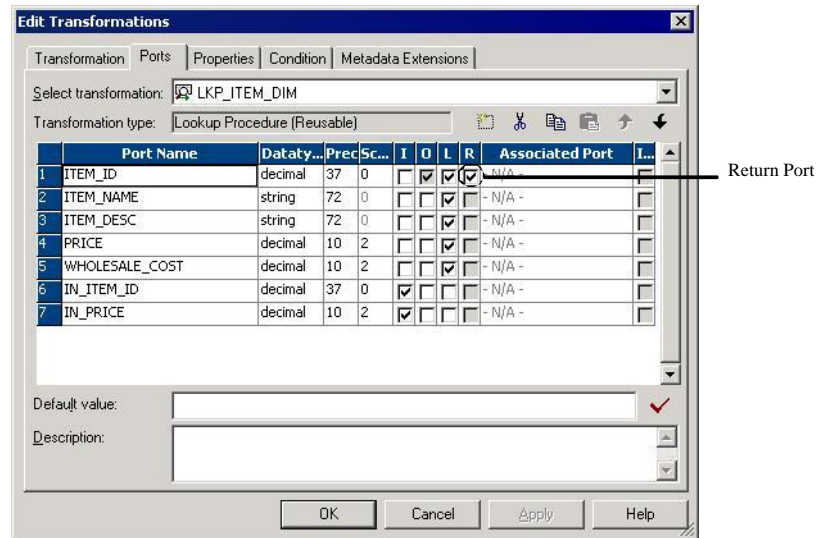
Step 3. Designate a Return Value

You can pass multiple input values into a Lookup transformation and return one column of data. Designate one lookup/output port as a return port. The Integration Service can return one value from the lookup query. Use the return port to specify the return value. If you call the unconnected lookup from an update strategy or filter expression, you are generally checking for null values. In this case, the return port can be anything. If you call the lookup from an expression performing a calculation, the return value needs to be the value you want to include in the calculation.

To continue the update strategy example, you can define the ITEM_ID port as the return port. The update strategy expression checks for null values returned. If the lookup condition is true, the Integration Service returns the ITEM_ID. If the condition is false, the Integration Service returns NULL.

Figure 14-1 shows a return port in a Lookup transformation:

Figure 14-1. Return Port in a Lookup Transformation



Step 4. Call the Lookup Through an Expression

Supply input values for an unconnected Lookup transformation from a :LKP expression in another transformation. The arguments are local input ports that match the Lookup transformation input ports used in the lookup condition. Use the following syntax for a :LKP expression:

```
:LKP.lookup_transformation_name(argument, argument, ...)
```

To continue the example about the retail store, when you write the update strategy expression, the order of ports in the expression must match the order in the lookup condition. In this case, the ITEM_ID condition is the first lookup condition, and therefore, it is the first argument in the update strategy expression.

```
IIF(ISNULL(:LKP.lkpITEMS_DIM(ITEM_ID, PRICE)), DD_UPDATE, DD_REJECT)
```

Use the following guidelines to write an expression that calls an unconnected Lookup transformation:

- φ The order in which you list each argument must match the order of the lookup conditions in the Lookup transformation.
- φ The datatypes for the ports in the expression must match the datatypes for the input ports in the Lookup transformation. The Designer does not validate the expression if the datatypes do not match.
- φ If one port in the lookup condition is not a lookup/output port, the Designer does not validate the expression.
- φ The argument ports in the expression must be in the same order as the input ports in the lookup condition.
- φ If you use incorrect :LKP syntax, the Designer marks the mapping invalid.
- φ If you call a connected Lookup transformation in a :LKP expression, the Designer marks the mapping invalid.

Tip: Avoid syntax errors when you enter expressions by using the point-and-click method to select functions and ports.

Creating a Lookup Transformation

Create a reusable Lookup transformation in the Transformation Developer. Create a non-reusable Lookup transformation in the Mapping Designer.

RELATED TOPICS:

- ϕ “Creating a Reusable Pipeline Lookup Transformation”
- ϕ “Creating a Non-Reusable Pipeline Lookup Transformation”

To create a Lookup transformation:

1. To create a reusable Lookup transformation, open the Transformation Developer.

-or-

To create a non-reusable Lookup transformation, open a mapping in the Mapping Designer. If you are creating pipeline Lookup transformation, drag in a source definition to use as a lookup source.

2. Click Transformation > Create. Select the Lookup transformation.
3. Enter a name for the transformation. Click OK.

The naming convention for Lookup transformations is *LKP_TransformationName*.

4. In the Select Lookup Table dialog box, choose one of the following options for the lookup source:

- ϕ Source definition from the repository.
- ϕ Target definition from the repository.
- ϕ Source qualifier from the mapping.
- ϕ Import a relational table or file from the repository.
- ϕ Skip. Do not import a definition. Create the Lookup transformation ports manually.

When you choose the lookup source, the Designer creates ports in the transformation based on the ports in the object that you choose. The Designer configures each port as a lookup port and an output port. The lookup ports represent the columns in the lookup source. The Lookup transformation receives data from the lookup source in each lookup port and passes the data to the target.

Choose Skip to manually add the lookup ports instead of importing a definition. You can choose which lookup ports are also output ports.

5. For a connected Lookup transformation, add input and output ports.

You can pass data through the transformation and return data from the lookup table to the target.

6. For an unconnected Lookup transformation, create a return port for the value you want to return from the lookup.

You can return one column to the transformation that called the lookup.

7. Click the Properties tab to configure the Lookup transformation properties. Configure lookup caching.

Lookup caching is enabled by default for pipeline and flat file Lookup transformations.

8. For a Lookup transformation that has a dynamic lookup cache, associate an input port, output port, or sequence ID with each lookup port.

The Integration Service inserts or updates rows in the lookup cache with the data from the associated ports. If you associate a sequence ID, the Integration Service generates a primary key for inserted rows in the lookup cache.

9. Add the lookup condition on the Condition tab.

The lookup condition compares the source column values with values in the lookup source. The Condition tab Transformation Port represents the source column values. The Lookup Table represents the lookup source.

Creating a Reusable Pipeline Lookup Transformation

Create a reusable pipeline Lookup transformation in the Transformation Developer. When you create the transformation, choose Source for the lookup table location. The Transformation Developer displays a list of source definitions from the repository.

When you choose a source qualifier that represents a relational table or flat file source definition, the Designer creates a relational or flat file Lookup transformation. When the source qualifier represents an application source, the Designer creates a pipeline Lookup transformation. To create a pipeline Lookup transformation for a relational or flat file lookup source, change the source type to Source Qualifier after you create the transformation. Enter the name of the source definition in the Lookup Table property.

When you drag a reusable pipeline Lookup transformation into a mapping, the Mapping Designer adds the source definition from the Lookup Table property to the mapping. The Designer adds the Source Qualifier transformation.

To change the Lookup Table Name to another source qualifier in the mapping, click the Open button in the Lookup Table Name property. Choose a Source Qualifier from the list.

Creating a Non-Reusable Pipeline Lookup Transformation

Create a non-reusable pipeline Lookup transformation in the Mapping Designer. Drag a source definition into a mapping. When you create the transformation in the Mapping Designer, select Source Qualifier as the lookup table location. The Mapping Designer displays a list of the source qualifiers in the mapping. When you select a source qualifier, the Mapping Designer populates the Lookup transformation with port names and attributes from the source qualifier you choose.

Rank Transformation

This chapter includes the following topics:

- φ Overview
- φ Ports in a Rank Transformation
- φ Defining Groups
- φ Creating a Rank Transformation

Overview

Transformation type:

Active
Connected

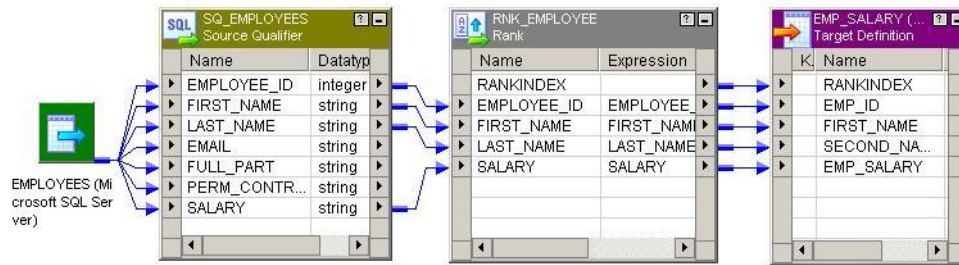
You can select only the top or bottom rank of data with Rank transformation. Use a Rank transformation to return the largest or smallest numeric value in a port or group. You can also use a Rank transformation to return the strings at the top or the bottom of a session sort order. During the session, the Integration Service caches input data until it can perform the rank calculations.

The Rank transformation differs from the transformation functions MAX and MIN, in that it lets you select a group of top or bottom values, not just one value. For example, use Rank to select the top 10 salespersons in a given territory. Or, to generate a financial report, you might also use a Rank transformation to identify the three departments with the lowest expenses in salaries and overhead. While the SQL language provides many functions designed to handle groups of data, identifying top or bottom strata within a set of rows is not possible using standard SQL functions.

You connect all ports representing the same row set to the transformation. Only the rows that fall within that rank, based on some measure you set when you configure the transformation, pass through the Rank transformation. You can also write expressions to transform data or perform calculations.

Figure 17-1 shows a mapping that passes employee data from a human resources table through a Rank transformation. The Rank transformation only passes the rows for the top 10 highest paid employees to the next transformation.

Figure 17-1. Sample Mapping with a Rank Transformation



As an active transformation, the Rank transformation might change the number of rows passed through it. You might pass 100 rows to the Rank transformation, but select to rank only the top 10 rows, which pass from the Rank transformation to another transformation.

You can connect ports from only one transformation to the Rank transformation. You can also create local variables and write non-aggregate expressions.

Ranking String Values

When the Integration Service runs in the ASCII data movement mode, it sorts session data using a binary sort order.

When the Integration Service runs in Unicode data movement mode, the Integration Service uses the sort order configured for the session. You select the session sort order in the session properties. The session properties lists all available sort orders based on the code page used by the Integration Service.

For example, you have a Rank transformation configured to return the top three values of a string port. When you configure the workflow, you select the Integration Service on which you want the workflow to run. The session properties display all sort orders associated with the code page of the selected Integration Service, such as French, German, and Binary. If you configure the session to use a binary sort order, the Integration Service calculates the binary value of each string, and returns the three rows with the highest binary values for the string.

Rank Caches

During a session, the Integration Service compares an input row with rows in the data cache. If the input row out-ranks a cached row, the Integration Service replaces the cached row with the input row. If you configure the Rank transformation to rank across multiple groups, the Integration Service ranks incrementally for each group it finds.

The Integration Service stores group information in an index cache and row data in a data cache. If you create multiple partitions in a pipeline, the Integration Service creates separate caches for each partition.

Rank Transformation Properties

When you create a Rank transformation, you can configure the following properties:

- φ Enter a cache directory.
- φ Select the top or bottom rank.
- φ Select the input/output port that contains values used to determine the rank. You can select only one port to define a rank.
- φ Select the number of rows falling within a rank.

φ Define groups for ranks, such as the 10 least expensive products for each manufacturer.

Ports in a Rank Transformation

The Rank transformation includes input or input/output ports connected to another transformation in the mapping. It also includes variable ports and a rank port. Use the rank port to specify the column you want to rank.

The following table describes the ports in a Rank transformation:

Ports	Number Required	Description
I	Minimum of one	Input port. Create an input port to receive data from another transformation.
O	Minimum of one	Output port. Create an output port for each port you want to link to another transformation. You can designate input ports as output ports.
V	Not Required	Variable port. Can use to store values or calculations to use in an expression. Variable ports cannot be input or output ports. They pass data within the transformation only.
R	One only	Rank port. Use to designate the column for which you want to rank values. You can designate only one Rank port in a Rank transformation. The Rank port is an input/output port. You must link the Rank port to another transformation.

Rank Index

The Designer creates a RANKINDEX port for each Rank transformation. The Integration Service uses the Rank Index port to store the ranking position for each row in a group. For example, if you create a Rank transformation that ranks the top five salespersons for each quarter, the rank index numbers the salespeople from 1 to 5:

RANKINDEX	SALES_PERSON	SALES
1	Sam	10,000
2	Mary	9,000
3	Alice	8,000
4	Ron	7,000
5	Alex	6,000

The RANKINDEX is an output port only. You can pass the rank index to another transformation in the mapping or directly to a target.

Defining Groups

Like the Aggregator transformation, the Rank transformation lets you group information. For example, if you want to select the 10 most expensive items by manufacturer, you would first define a group for each manufacturer. When you configure the Rank transformation, you can set one of its input/output ports as a group by port. For each unique value in the group port, the transformation creates a group of rows falling within the rank definition (top or bottom, and a particular number in each rank).

Therefore, the Rank transformation changes the number of rows in two different ways. By filtering all but the rows falling within a top or bottom rank, you reduce the number of rows that pass through the transformation. By defining groups, you create one set of ranked rows for each group.

For example, you might create a Rank transformation to identify the 50 highest paid employees in the company.

In this case, you would identify the SALARY column as the input/output port used to measure the ranks, and configure the transformation to filter out all rows except the top 50.

After the Rank transformation identifies all rows that belong to a top or bottom rank, it then assigns rank index values. In the case of the top 50 employees, measured by salary, the highest paid employee receives a rank index of 1. The next highest-paid employee receives a rank index of 2, and so on. When measuring a bottom rank, such as the 10 lowest priced products in the inventory, the Rank transformation assigns a rank index from lowest to highest. Therefore, the least expensive item would receive a rank index of 1.

If two rank values match, they receive the same value in the rank index and the transformation skips the next value. For example, if you want to see the top five retail stores in the country and two stores have the same sales, the return data might look similar to the following:

RANKINDEX	SALES	STORE
1	10000	Orange
1	10000	Brea
3	90000	Los Angeles
4	80000	Ventura

Creating a Rank Transformation

You can add a Rank transformation anywhere in the mapping after the source qualifier.

To create a Rank transformation:

1. In the Mapping Designer, click Transformation > Create. Select the Rank transformation. Enter a name for the Rank. The naming convention for Rank transformations is *RNK_TransformationName*.

Enter a description for the transformation. This description appears in the Repository Manager.

2. Click Create, and then click Done.

The Designer creates the Rank transformation.

3. Link columns from an input transformation to the Rank transformation.

4. Click the Ports tab and select the Rank (R) option for the rank port.

If you want to create groups for ranked rows, select Group By for the port that defines the group.

5. Click the Properties tab and select whether you want the top or bottom rank.

6. For the Number of Ranks option, enter the number of rows you want to select for the rank.

7. Change the other Rank transformation properties, if necessary.

The following table describes the Rank transformation properties:

Setting	Description
Cache Directory	Local directory where the Integration Service creates the index and data cache files. By default, the Integration Service uses the directory entered in the Workflow Manager for the process variable \$PMCacheDir. If you enter a new directory, make sure the directory exists and contains enough disk space for the cache files.
Top/Bottom	Specifies whether you want the top or bottom ranking for a column.

Setting	Description
Number of Ranks	Number of rows you want to rank.
Case-Sensitive String Comparison	When running in Unicode mode, the Integration Service ranks strings based on the sort order selected for the session. If the session sort order is case sensitive, select this option to enable case-sensitive string comparisons, and clear this option to have the Integration Service ignore case for strings. If the sort order is not case sensitive, the Integration Service ignores this setting. By default, this option is selected.
Tracing Level	Determines the amount of information the Integration Service writes to the session log about data passing through this transformation in a session.
Rank Data Cache Size	Data cache size for the transformation. Default is 2,000,000 bytes. If the total configured session cache size is 2 GB (2,147,483,648 bytes) or more, you must run the session on a 64-bit Integration Service. You can configure a numeric value, or you can configure the Integration Service to determine the cache size at runtime. If you configure the Integration Service to determine the cache size, you can also configure a maximum amount of memory for the Integration Service to allocate to the cache.
Rank Index Cache Size	Index cache size for the transformation. Default is 1,000,000 bytes. If the total configured session cache size is 2 GB (2,147,483,648 bytes) or more, you must run the session on a 64-bit Integration Service. You can configure a numeric value, or you can configure the Integration Service to determine the cache size at runtime. If you configure the Integration Service to determine the cache size, you can also configure a maximum amount of memory for the Integration Service to allocate to the cache.
Transformation Scope	Specifies how the Integration Service applies the transformation logic to incoming data: <ul style="list-style-type: none"> - Transaction. Applies the transformation logic to all rows in a transaction. Choose Transaction when a row of data depends on all rows in the same transaction, but does not depend on rows in other transactions. - All Input. Applies the transformation logic on all incoming data. When you choose All Input, the PowerCenter drops incoming transaction boundaries. Choose All Input when a row of data depends on all rows in the source.

8. Click OK.

Router Transformation

This chapter includes the following topics:

- φ Overview
- φ Working with Groups
- φ Working with Ports
- φ Connecting Router Transformations in a Mapping
- φ Creating a Router Transformation

Overview

Transformation type:

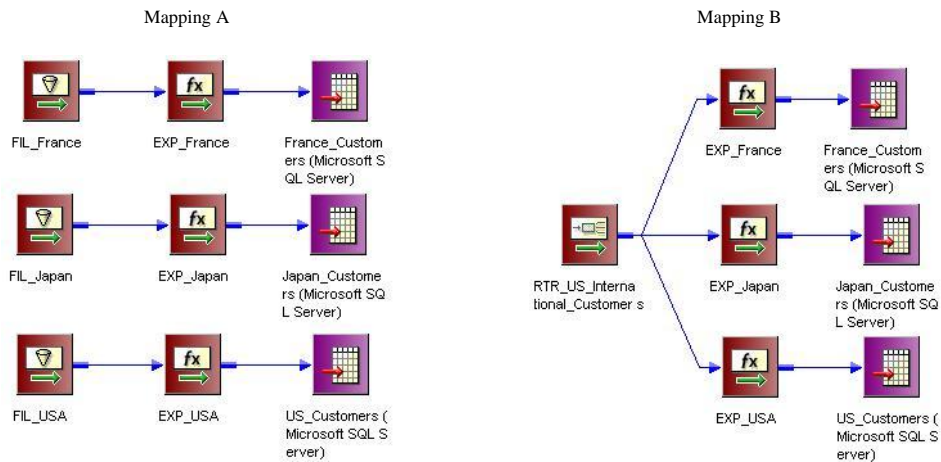
Active
Connected

A Router transformation is similar to a Filter transformation because both transformations allow you to use a condition to test data. A Filter transformation tests data for one condition and drops the rows of data that do not meet the condition. However, a Router transformation tests data for one or more conditions and gives you the option to route rows of data that do not meet any of the conditions to a default output group.

If you need to test the same input data based on multiple conditions, use a Router transformation in a mapping instead of creating multiple Filter transformations to perform the same task. The Router transformation is more efficient. For example, to test data based on three conditions, you only need one Router transformation instead of three filter transformations to perform this task. Likewise, when you use a Router transformation in a mapping, the Integration Service processes the incoming data only once. When you use multiple Filter transformations in a mapping, the Integration Service processes the incoming data for each transformation.

Figure 18-1 shows two mappings that perform the same task. Mapping A uses three Filter transformations while Mapping B produces the same result with one Router transformation:

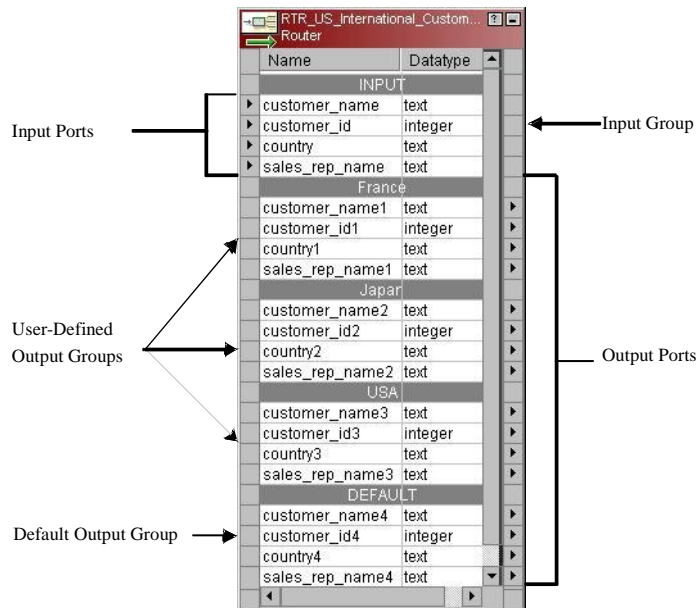
Figure 18-1. Comparing Router and Filter Transformations



A Router transformation consists of input and output groups, input and output ports, group filter conditions, and properties that you configure in the Designer.

Figure 18-2 shows a sample Router transformation and its components:

Figure 18-2. Sample Router Transformation



Working with Groups

A Router transformation has the following types of groups:

- φ Input
- φ Output

Input Group

The Designer copies property information from the input ports of the input group to create a set of output ports for each output group.

Output Groups

There are two types of output groups:

- ϕ User-defined groups
- ϕ Default group

You cannot modify or delete output ports or their properties.

User-Defined Groups

You create a user-defined group to test a condition based on incoming data. A user-defined group consists of output ports and a group filter condition. You can create and edit user-defined groups on the Groups tab with the Designer. Create one user-defined group for each condition that you want to specify.

The Integration Service uses the condition to evaluate each row of incoming data. It tests the conditions of each user-defined group before processing the default group. The Integration Service determines the order of evaluation for each condition based on the order of the connected output groups. The Integration Service processes user-defined groups that are connected to a transformation or a target in a mapping. The Integration Service only processes user-defined groups that are *not* connected in a mapping if the default group is connected to a transformation or a target.

If a row meets more than one group filter condition, the Integration Service passes this row multiple times.

The Default Group

The Designer creates the default group after you create one new user-defined group. The Designer does not allow you to edit or delete the default group. This group does not have a group filter condition associated with it. If *all* of the conditions evaluate to FALSE, the Integration Service passes the row to the default group. If you want the Integration Service to drop all rows in the default group, do not connect it to a transformation or a target in a mapping.

The Designer deletes the default group when you delete the last user-defined group from the list.

Using Group Filter Conditions

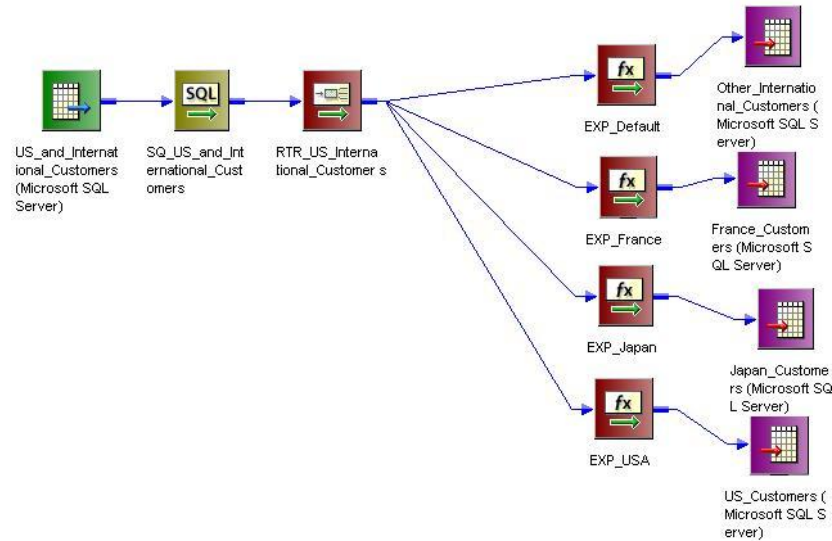
You can test data based on one or more group filter conditions. You create group filter conditions on the Groups tab using the Expression Editor. You can enter any expression that returns a single value. You can also specify a constant for the condition. A group filter condition returns TRUE or FALSE for each row that passes through the transformation, depending on whether a row satisfies the specified condition. Zero (0) is the equivalent of FALSE, and any non-zero value is the equivalent of TRUE. The Integration Service passes the rows of data that evaluate to TRUE to each transformation or target that is associated with each user-defined group.

For example, you have customers from nine countries, and you want to perform different calculations on the data from only three countries. You might want to use a Router transformation in a mapping to filter this data to three different Expression transformations.

There is no group filter condition associated with the default group. However, you can create an Expression transformation to perform a calculation based on the data from the other six countries.

Figure 18-3 shows a mapping with a Router transformation that filters data based on multiple conditions:

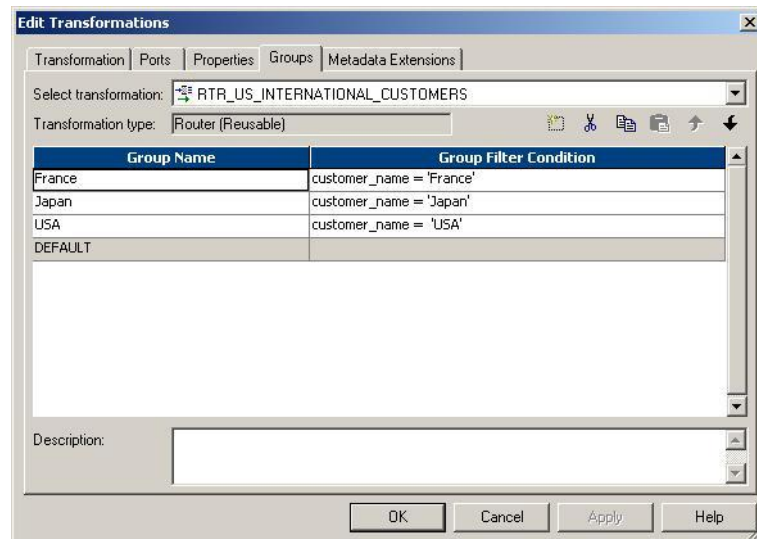
Figure 18-3. Using a Router Transformation in a Mapping



Since you want to perform multiple calculations based on the data from three different countries, create three user-defined groups and specify three group filter conditions on the Groups tab.

Figure 18-4 shows group filter conditions that filter customer data:

Figure 18-4. Specifying Group Filter Conditions



In the session, the Integration Service passes the rows of data that evaluate to TRUE to each transformation or target that is associated with each user-defined group, such as Japan, France, and USA. The Integration Service passes the row to the default group if *all* of the conditions evaluate to FALSE. If this happens, the Integration Service passes the data of the other six countries to the transformation or target that is associated with the default group. If you want the Integration Service to drop all rows in the default group, do not connect it to a transformation or a target in a mapping.

Adding Groups

Adding a group is similar to adding a port in other transformations. The Designer copies property information from the input ports to the output ports.

To add a group to a Router transformation:

1. Click the Groups tab.
2. Click the Add button.
3. Enter a name for the new group in the Group Name section.
4. Click the Group Filter Condition field and open the Expression Editor.
5. Enter the group filter condition.
6. Click Validate to check the syntax of the condition.
7. Click OK.

Working with Ports

A Router transformation has input ports and output ports. Input ports are in the input group, and output ports are in the output groups. You can create input ports by copying them from another transformation or by manually creating them on the Ports tab.

The Designer creates output ports by copying the following properties from the input ports:

- φ Port name
- φ Datatype
- φ Precision
- φ Scale
- φ Default value

When you make changes to the input ports, the Designer updates the output ports to reflect these changes. You cannot edit or delete output ports. The output ports display in the Normal view of the Router transformation.

The Designer creates output port names based on the input port names. For each input port, the Designer creates a corresponding output port in each output group.

Figure 18-5 shows the output port names of a Router transformation that correspond to the input port names:

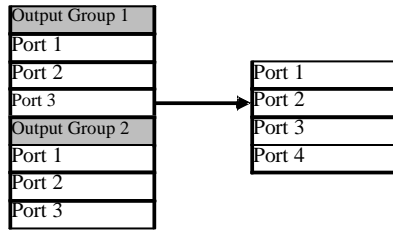
Figure 18-5. Input Port Name and Corresponding Output Port Names

Name	Datatype
INPUT	
customer_name	text
customer_id	integer
country	text
sales_rep_name	text
France	
customer_name1	text
customer_id1	integer
country1	text
sales_rep_name1	text
Japan	
customer_name2	text
customer_id2	integer
country2	text
sales_rep_name2	text
USA	
customer_name3	text
customer_id3	integer
country3	text
sales_rep_name3	text
DEFAULT	
customer_name4	text
customer_id4	integer
country4	text
sales_rep_name4	text

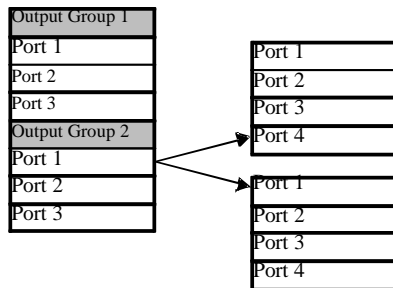
Connecting Router Transformations in a Mapping

When you connect transformations to a Router transformation in a mapping, consider the following rules:

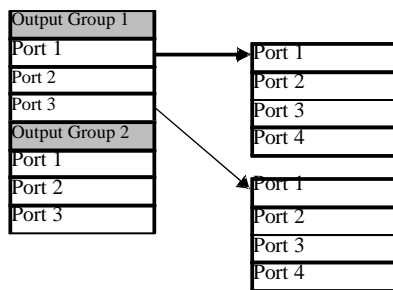
- ϕ You can connect one group to one transformation or target.



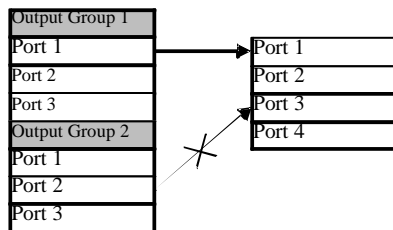
- ϕ You can connect one output port in a group to multiple transformations or targets.



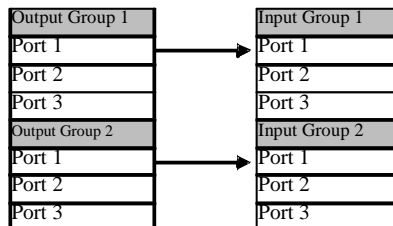
- ϕ You can connect multiple output ports in one group to multiple transformations or targets.



- ϕ You cannot connect more than one group to one target or a single input group transformation.



- ϕ You can connect more than one group to a multiple input group transformation, except for Joiner transformations, when you connect each output group to a different input group.



Creating a Router Transformation

To add a Router transformation to a mapping, complete the following steps.

To create a Router transformation:

1. In the Mapping Designer, open a mapping.
2. Click Transformation > Create.
Select Router transformation, and enter the name of the new transformation. The naming convention for the Router transformation is *RTR_TransformationName*. Click Create, and then click Done.
3. Select and drag all the ports from a transformation to add them to the Router transformation, or you can manually create input ports on the Ports tab.
4. Double-click the title bar of the Router transformation to edit transformation properties.
5. Click the Transformation tab and configure transformation properties.
6. Click the Properties tab and configure tracing levels.
7. Click the Groups tab, and then click the Add button to create a user-defined group.
The Designer creates the default group when you create the first user-defined group.
8. Click the Group Filter Condition field to open the Expression Editor.
9. Enter a group filter condition.
10. Click Validate to check the syntax of the conditions you entered.
11. Click OK.
12. Connect group output ports to transformations or targets.

CHAPTER 13

Sequence Generator Transformation

This chapter includes the following topics:

- φ Overview
- φ Common Uses
- φ Sequence Generator Ports
- φ Transformation Properties
- φ Creating a Sequence Generator Transformation

Overview

Transformation type:

Passive
Connected

The Sequence Generator transformation generates numeric values. Use the Sequence Generator to create unique primary key values, replace missing primary keys, or cycle through a sequential range of numbers.

The Sequence Generator transformation is a connected transformation. It contains two output ports that you can connect to one or more transformations. The Integration Service generates a block of sequence numbers each time a block of rows enters a connected transformation. If you connect CURRVAL, the Integration Service processes one row in each block. When NEXTVAL is connected to the input port of another transformation, the Integration Service generates a sequence of numbers. When CURRVAL is connected to the input port of another transformation, the Integration Service generates the NEXTVAL value plus the Increment By value.

You can make a Sequence Generator reusable, and use it in multiple mappings. You might reuse a Sequence Generator when you perform multiple loads to a single target.

For example, if you have a large input file that you separate into three sessions running in parallel, use a Sequence Generator to generate primary key values. If you use different Sequence Generators, the Integration Service might generate duplicate key values. Instead, use the reusable Sequence Generator for all three sessions to provide a unique value for each target row.

Common Uses

You can complete the following tasks with a Sequence Generator transformation:

- ϕ Create keys.
- ϕ Replace missing values.
- ϕ Cycle through a sequential range of numbers.

Creating Keys

You can create primary or foreign key values with the Sequence Generator transformation by connecting the NEXTVAL port to a target or downstream transformation. You can use a range of values from 1 to 9,223,372,036,854,775,807 with the smallest interval of 1.

When you create primary or foreign keys, use the Cycle option to prevent the Integration Service from creating duplicate primary keys. You might do this by selecting the Truncate Target Table option in the session properties or by creating composite keys.

To create a composite key, you can configure the Integration Service to cycle through a smaller set of values. For example, if you have three stores generating order numbers, you might have a Sequence Generator cycling through values from 1 to 3, incrementing by 1. When you pass the following set of foreign keys, the generated values then create unique composite keys:

COMPOSITE_KEY	ORDER_NO
1	12345
2	12345
3	12345
1	12346
2	12346
3	12346

Replacing Missing Values

Use the Sequence Generator transformation to replace missing keys by using NEXTVAL with the IIF and ISNULL functions.

For example, to replace null values in the ORDER_NO column, you create a Sequence Generator transformation with the properties and drag the NEXTVAL port to an Expression transformation. In the Expression transformation, drag the ORDER_NO port into the transformation along with any other necessary ports. Then create an output port, ALL_ORDERS.

In ALL_ORDERS, you can then enter the following expression to replace null orders:

```
IIF( ISNULL( ORDER_NO ), NEXTVAL, ORDER_NO )
```

Sequence Generator Ports

The Sequence Generator transformation has two output ports: NEXTVAL and CURRVAL. You cannot edit or delete these ports. Likewise, you cannot add ports to the transformation.

NEXTVAL

Connect NEXTVAL to multiple transformations to generate unique values for each row in each transformation. Use the NEXTVAL port to generate sequence numbers by connecting it to a downstream transformation or target. You connect the NEXTVAL port to generate the sequence based on the Current Value and Increment By properties. If the Sequence Generator is not configured to cycle through the sequence, the NEXTVAL port generates sequence numbers up to the configured End Value.

For example, you might connect NEXTVAL to two targets in a mapping to generate unique primary key values. The Integration Service creates a column of unique primary key values for each target table. The column of unique primary key values is sent to one target table as a block of sequence numbers. The other target receives a block of sequence numbers from the Sequence Generator transformation after the first target receives the block of sequence numbers.

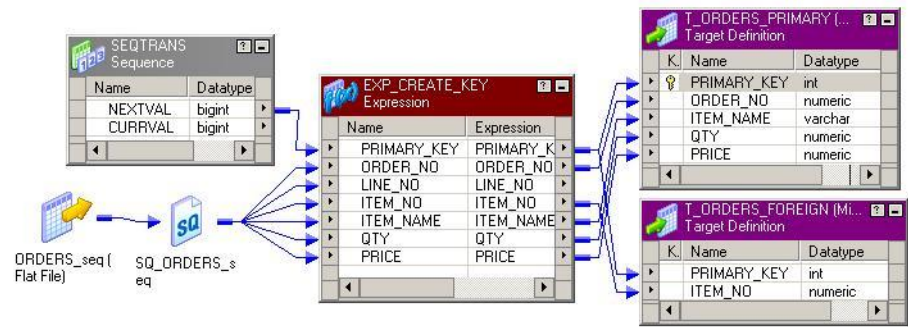
For example, you configure the Sequence Generator transformation as follows: Current Value = 1, Increment By = 1. The Integration Service generates the following primary key values for the T_ORDERS_PRIMARY and T_ORDERS_FOREIGN target tables:

T_ORDERS_PRIMARY TABLE:	T_ORDERS_FOREIGN TABLE:
PRIMARY KEY	PRIMARY KEY
1	6
2	7
3	8
4	9
5	10

If you want the *same* values to go to more than one target that receives data from a single transformation, you can connect a Sequence Generator transformation to that preceding transformation. The Integration Service processes the values into a block of sequence numbers. This allows the Integration Service to pass unique values to the transformation, and then route rows from the transformation to targets.

Figure 19-1 shows a mapping with a Sequence Generator that passes unique values to the Expression transformation. The Expression transformation populates both targets with identical primary key values.

Figure 19-1. Mapping with a Sequence Generator and an Expression Transformation



For example, you configure the Sequence Generator transformation as follows: Current Value = 1, Increment By = 1. The Integration Service generates the following primary key values for the T_ORDERS_PRIMARY and T_ORDERS_FOREIGN target tables:

T_ORDERS_PRIMARY TABLE:	T_ORDERS_FOREIGN TABLE:
PRIMARY KEY	PRIMARY KEY
1	1
2	2
3	3
4	4

T_ORDERS_PRIMARY TABLE:

PRIMARY KEY

5

T_ORDERS_FOREIGN TABLE:

PRIMARY KEY

5

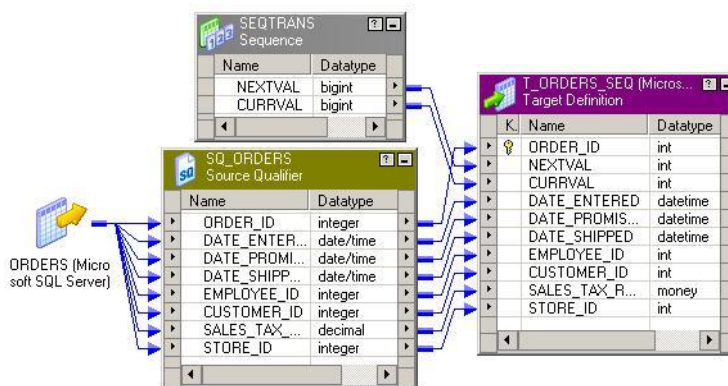
Note: When you run a partitioned session on a grid, the Sequence Generator transformation skips values depending on the number of rows in each partition.

CURRVAL

CURRVAL is NEXTVAL plus the Increment By value. You typically only connect the CURRVAL port when the NEXTVAL port is already connected to a downstream transformation. When a row enters a transformation connected to the CURRVAL port, the Integration Service passes the last created NEXTVAL value plus one.

Figure 19-2 shows connecting CURRVAL and NEXTVAL ports to a target:

Figure 19-2. Connecting CURRVAL and NEXTVAL Ports to a Target



For example, you configure the Sequence Generator transformation as follows: Current Value = 1, Increment By = 1. The Integration Service generates the following values for NEXTVAL and CURRVAL:

NEXTVAL	CURRVAL
1	2
2	3
3	4
4	5
5	6

If you connect the CURRVAL port without connecting the NEXTVAL port, the Integration Service passes a constant value for each row. When you connect the CURRVAL port in a Sequence Generator transformation, the Integration Service processes one row in each block. You can optimize performance by connecting only the NEXTVAL port in a mapping.

Note: When you run a partitioned session on a grid, the Sequence Generator transformation might skip values depending on the number of rows in each partition.

Transformation Properties

The Sequence Generator transformation is unique among all transformations because you cannot add, edit, or delete the default ports, NEXTVAL and CURRVAL.

The following table describes the Sequence Generator transformation properties you can configure:

Sequence Generator Setting	Description
Start Value	Start value of the generated sequence that you want the Integration Service to use if you use the Cycle option. If you select Cycle, the Integration Service cycles back to this value when it reaches the end value. Default is 0. Maximum value is 9,223,372,036,854,775,806.
Increment By	Difference between two consecutive values from the NEXTVAL port. Default is 1. Maximum value is 2,147,483,647.
End Value	Maximum value the Integration Service generates. If the Integration Service reaches this value during the session and the sequence is not configured to cycle, the session fails. Maximum value is 9,223,372,036,854,775,807. If you connect the NEXTVAL port to a downstream integer port, set End Value to a value no larger than the integer maximum value. If the NEXTVAL exceeds the datatype maximum value for the downstream port, the session fails.
Current Value	Current value of the sequence. Enter the value you want the Integration Service to use as the first value in the sequence. If you want to cycle through a series of values, the value must be greater than or equal to the start value and less than the end value. If the Number of Cached Values is set to 0, the Integration Service updates the current value to reflect the last-generated value for the session plus one, and then uses the updated current value as the basis for the next time you run this session. However, if you use the Reset option, the Integration Service resets this value to its original value after each session. Note: If you edit this setting, you reset the sequence to the new setting. If you reset Current Value to 10, and the increment is 1, the next time you use the session, the Integration Service generates a first value of 10. Maximum value is 9,223,372,036,854,775,806. The Integration Service sets the value to NULL if the current value exceeds the maximum value.
Cycle	If enabled, the Integration Service cycles through the sequence range. If disabled, the Integration Service stops the sequence at the configured end value. The Integration Service fails the session with overflow errors if it reaches the end value and still has rows to process.
Number of Cached Values	Number of sequential values the Integration Service caches at a time. Use this option when multiple sessions use the same reusable Sequence Generator at the same time to ensure each session receives unique values. The Integration Service updates the repository as it caches each value. When set to 0, the Integration Service does not cache values. Default value for a standard Sequence Generator is 0. Default value for a reusable Sequence Generator is 1,000. Maximum value is 9,223,372,036,854,775,807.
Reset	If enabled, the Integration Service generates values based on the original current value for each session. Otherwise, the Integration Service updates the current value to reflect the last-generated value for the session plus one, and then uses the updated current value as the basis for the next session run. Disabled for reusable Sequence Generator transformations.
Tracing Level	Level of detail about the transformation that the Integration Service writes into the session log.

Start Value and Cycle

Use Cycle to generate a repeating sequence, such as numbers 1 through 12 to correspond to the months in a year.

To cycle the Integration Service through a sequence:

1. Enter the lowest value in the sequence that you want the Integration Service to use for the Start Value.
2. Enter the highest value to be used for End Value.
3. Select Cycle.

As it cycles, the Integration Service reaches the configured end value for the sequence, it wraps around and starts the cycle again, beginning with the configured Start Value.

Increment By

The Integration Service generates a sequence (NEXTVAL) based on the Current Value and Increment By properties in the Sequence Generator transformation.

The Current Value property is the value at which the Integration Service starts creating the sequence for each session. Increment By is the integer the Integration Service adds to the existing value to create the new value in the sequence. By default, the Current Value is set to 1, and Increment By is set to 1.

For example, you might create a Sequence Generator transformation with a current value of 1,000 and an increment of 10. If you pass three rows through the mapping, the Integration Service generates the following set of values:

```
1000
1010
1020
```

End Value

End Value is the maximum value you want the Integration Service to generate. If the Integration Service reaches the end value and the Sequence Generator is not configured to cycle through the sequence, the session fails with the following error message:

```
TT_11009 Sequence Generator Transformation: Overflow error.
```

Set the end value to any integer between 1 and 9,233,372,036,854,775,807. If you connect the NEXTVAL port to a downstream integer port, set the end value to a value no larger than the integer maximum value. For example, if you connect the NEXTVAL port to a Small Integer port, set the end value to a maximum of 32,767. If the NEXTVAL exceeds the datatype maximum value for the downstream port, the session fails.

Current Value

The Integration Service uses the current value as the basis for generated values for each session. To indicate which value you want the Integration Service to use the first time it uses the Sequence Generator transformation, you must enter that value as the current value. If you want to use the Sequence Generator transformation to cycle through a series of values, the current value must be greater than or equal to Start Value and less than the end value.

At the end of each session, the Integration Service updates the current value to the last value generated for the session plus one if the Sequence Generator Number of Cached Values is 0. For example, if the Integration Service ends a session with a generated value of 101, it updates the Sequence Generator current value to 102 in the repository. The next time the Sequence Generator is used, the Integration Service uses 102 as the basis for the next generated value. If the Sequence Generator Increment By is 1, when the Integration Service starts another session using the Sequence Generator, the first generated value is 102.

If you have multiple versions of a Sequence Generator transformation, the Integration Service updates the current value across all versions when it runs a session. The Integration Service updates the current value across versions regardless of whether you have checked out the Sequence Generator transformation or the parent mapping. The updated current value overrides an edited current value for a Sequence Generator transformation if the two values are different.

For example, User 1 creates Sequence Generator transformation and checks it in, saving a current value of 10 to Sequence Generator version 1. Then User 1 checks out the Sequence Generator transformation and enters a new current value of 100 to Sequence Generator version 2. User 1 keeps the Sequence Generator transformation checked out. Meanwhile, User 2 runs a session that uses the Sequence Generator transformation version 1. The Integration Service uses the checked-in value of 10 as the current value when User 2 runs the session. When the session completes, the current value is 150. The Integration Service updates the current value to 150 for version 1 and version 2 of the Sequence Generator transformation even though User 1 has the Sequence Generator transformation checked out.

If you open the mapping after you run the session, the current value displays the last value generated for the session plus one. Since the Integration Service uses the current value to determine the first value for each session, you should edit the current value only when you want to reset the sequence.

If you have multiple versions of the Sequence Generator transformation, and you want to reset the sequence, you must check in the mapping or reusable Sequence Generator transformation after you modify the current value.

Note: If you configure the Sequence Generator to Reset, the Integration Service uses the current value as the basis for the first generated value for each session.

Number of Cached Values

Number of Cached Values determines the number of values the Integration Service caches at one time. When Number of Cached Values is greater than zero, the Integration Service caches the configured number of values and updates the current value each time it caches values.

When multiple sessions use the same reusable Sequence Generator transformation at the same time, there might be multiple instances of the Sequence Generator transformation. To avoid generating the same values for each session, reserve a range of sequence values for each session by configuring Number of Cached Values.

Tip: To increase performance when running a session on a grid, increase the number of cached values for the Sequence Generator transformation. This reduces the communication required between the master and worker DTM processes and the repository.

Non-Reusable Sequence Generators

For non-reusable Sequence Generator transformations, Number of Cached Values is set to zero by default, and the Integration Service does not cache values during the session. When the Integration Service does not cache values, it accesses the repository for the current value at the start of a session. The Integration Service then generates values for the sequence. At the end of the session, the Integration Service updates the current value in the repository.

When you set Number of Cached Values greater than zero, the Integration Service caches values during the session. At the start of the session, the Integration Service accesses the repository for the current value, caches the configured number of values, and updates the current value accordingly. If the Integration Service uses all values in the cache, it accesses the repository for the next set of values and updates the current value. At the end of the session, the Integration Service discards any remaining values in the cache.

For non-reusable Sequence Generator transformations, setting Number of Cached Values greater than zero can increase the number of times the Integration Service accesses the repository during the session. It also causes sections of skipped values since unused cached values are discarded at the end of each session.

For example, you configure a Sequence Generator transformation as follows: Number of Cached Values = 50, Current Value = 1, Increment By = 1. When the Integration Service starts the session, it caches 50 values for the session and updates the current value to 50 in the repository. The Integration Service uses values 1 to 39 for the session and discards the unused values, 40 to 49. When the Integration Service runs the session again, it checks the repository for the current value, which is 50. It then caches the next 50 values and updates the current value to 100. During the session, it uses values 50 to 98. The values generated for the two sessions are 1 to 39 and 50 to 98.

Reusable Sequence Generators

When you have a reusable Sequence Generator transformation in several sessions and the sessions run at the same time, use Number of Cached Values to ensure each session receives unique values in the sequence. By default, Number of Cached Values is set to 1000 for reusable Sequence Generators.

When multiple sessions use the same Sequence Generator transformation at the same time, you risk generating the same values for each session. To avoid this, have the Integration Service cache a set number of values for each session by configuring Number of Cached Values.

For example, you configure a reusable Sequence Generator transformation as follows: Number of Cached Values = 50, Current Value = 1, Increment By = 1. Two sessions use the Sequence Generator, and they are scheduled to run at approximately the same time. When the Integration Service starts the first session, it caches 50 values for the session and updates the current value to 50 in the repository. The Integration Service begins using values 1 to 50 in the session. When the Integration Service starts the second session, it checks the repository for the current value, which is 50. It then caches the next 50 values and updates the current value to 100. It then uses values 51 to 100 in the second session. When either session uses all its cached values, the Integration Service caches a new set of values and updates the current value to ensure these values remain unique to the Sequence Generator.

For reusable Sequence Generator transformations, you can reduce Number of Cached Values to minimize discarded values, however it must be greater than one. When you reduce the Number of Cached Values, you might increase the number of times the Integration Service accesses the repository to cache values during the session.

Reset

If you select Reset for a non-reusable Sequence Generator transformation, the Integration Service generates values based on the original current value each time it starts the session. Otherwise, the Integration Service updates the current value to reflect the last-generated value plus one, and then uses the updated value the next time it uses the Sequence Generator transformation.

For example, you might configure a Sequence Generator transformation to create values from 1 to 1,000 with an increment of 1, and a current value of 1 and choose Reset. During the first session run, the Integration Service generates numbers 1 through 234. Each subsequent time the session runs, the Integration Service again generates numbers beginning with the current value of 1.

If you do not select Reset, the Integration Service updates the current value to 235 at the end of the first session run. The next time it uses the Sequence Generator transformation, the first value generated is 235.

Note: Reset is disabled for reusable Sequence Generator transformations.

Creating a Sequence Generator Transformation

To use a Sequence Generator transformation in a mapping, add it to the mapping, configure the transformation properties, and then connect NEXTVAL or CURRVAL to one or more transformations.

To create a Sequence Generator transformation:

1. In the Mapping Designer, click Transformation > Create. Select the Sequence Generator transformation.
The naming convention for Sequence Generator transformations is `SEQ_TransformationName`.
2. Enter a name for the Sequence Generator, and click Create. Click Done.
The Designer creates the Sequence Generator transformation.
3. Double-click the title bar of the transformation.
4. Enter a description for the transformation.

5. Select the Properties tab. Enter settings.

For a list of transformation properties, see “Transformation Properties”

Note: You cannot override the Sequence Generator transformation properties at the session level. This protects the integrity of the sequence values generated.

6. Click OK.
7. To generate new sequences during a session, connect the NEXTVAL port to at least one transformation in the mapping.

Use the NEXTVAL or CURRVAL ports in an expression in other transformations.

CHAPTER 14

Sorter Transformation

This chapter includes the following topics:

- φ Overview
- φ Sorting Data
- φ Sorter Transformation Properties
- φ Creating a Sorter Transformation

Overview

Transformation type:

Active
Connected

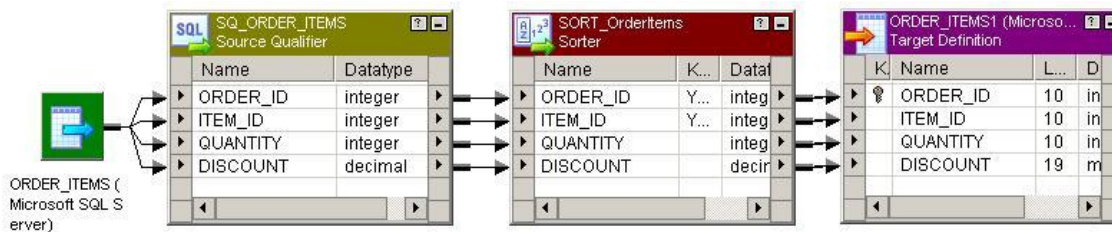
You can sort data with the Sorter transformation. You can sort data in ascending or descending order according to a specified sort key. You can also configure the Sorter transformation for case-sensitive sorting, and specify whether the output rows should be distinct. The Sorter transformation is an active transformation. It must be connected to the data flow.

You can sort data from relational or flat file sources. You can also use the Sorter transformation to sort data passing through an Aggregator transformation configured to use sorted input.

When you create a Sorter transformation in a mapping, you specify one or more ports as a sort key and configure each sort key port to sort in ascending or descending order. You also configure sort criteria the Integration Service applies to all sort key ports and the system resources it allocates to perform the sort operation.

Figure 20-1 shows a simple mapping that uses a Sorter transformation. The mapping passes rows from a sales table containing order information through a Sorter transformation before loading to the target.

Figure 20-1. Sample Mapping with a Sorter Transformation



Sorting Data

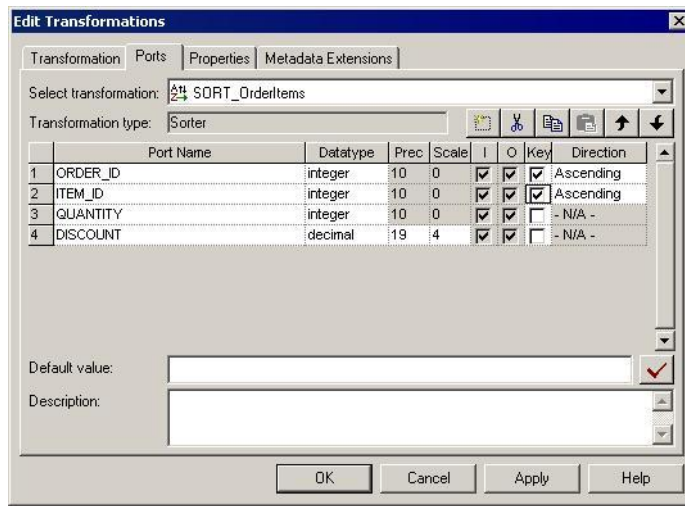
The Sorter transformation contains only input/output ports. All data passing through the Sorter transformation is sorted according to a sort key. The sort key is one or more ports that you want to use as the sort criteria.

You can specify more than one port as part of the sort key. When you specify multiple ports for the sort key, the Integration Service sorts each port sequentially. The order the ports appear in the Ports tab determines the succession of sort operations. The Sorter transformation treats the data passing through each successive sort key port as a secondary sort of the previous port.

At session run time, the Integration Service sorts data according to the sort order specified in the session properties. The sort order determines the sorting criteria for special characters and symbols.

Figure 20-2 shows the Ports tab configuration for the Sorter transformation sorting the data in ascending order by order ID and item ID:

Figure 20-2. Sample Sorter Transformation Ports Configuration



At session run time, the Integration Service passes the following rows into the Sorter transformation:

ORDER_ID	ITEM_ID	QUANTITY	DISCOUNT
45	123456	3	3.04
45	456789	2	12.02
43	000246	6	34.55
41	000468	5	.56

After sorting the data, the Integration Service passes the following rows out of the Sorter transformation:

ORDER_ID	ITEM_ID	QUANTITY	DISCOUNT
41	000468	5	.56
43	000246	6	34.55
45	123456	3	3.04
45	456789	2	12.02

Sorter Transformation Properties

The Sorter transformation has several properties that specify additional sort criteria. The Integration Service applies these criteria to all sort key ports. The Sorter transformation properties also determine the system resources the Integration Service allocates when it sorts data.

Sorter Cache Size

The Integration Service uses the Sorter Cache Size property to determine the maximum amount of memory it can allocate to perform the sort operation. The Integration Service passes all incoming data into the Sorter transformation before it performs the sort operation. You can configure a numeric value for the Sorter cache, or you can configure the Integration Service to determine the cache size at run time. If you configure the Integration Service to determine the cache size, you can also configure a maximum amount of memory for the Integration Service to allocate to the cache. If the total configured session cache size is 2 GB (2,147,483,648 bytes) or greater, you must run the session on a 64-bit Integration Service.

Before starting the sort operation, the Integration Service allocates the amount of memory configured for the Sorter cache size. If the Integration Service runs a partitioned session, it allocates the specified amount of Sorter cache memory for each partition.

If it cannot allocate enough memory, the Integration Service fails the session. For best performance, configure Sorter cache size with a value less than or equal to the amount of available physical RAM on the Integration Service machine. Allocate at least 16 MB (16,777,216 bytes) of physical memory to sort data using the Sorter transformation. Sorter cache size is set to 16,777,216 bytes by default.

If the amount of incoming data is greater than the amount of Sorter cache size, the Integration Service temporarily stores data in the Sorter transformation work directory. The Integration Service requires disk space of at least twice the amount of incoming data when storing data in the work directory. If the amount of incoming data is significantly greater than the Sorter cache size, the Integration Service may require much more than twice the amount of disk space available to the work directory.

Use the following formula to determine the size of incoming data:

$$\text{number_of_input_rows} [(\text{column_size}) + 16]$$

Table 20-1 gives the column size values by datatype for Sorter data calculations:

Table 20-1. Column Sizes for Sorter Data Calculations

Datatype	Column Size
Binary	precision + 8 Round to nearest multiple of 8
Date/Time	29
Decimal, high precision off (all precision)	16
Decimal, high precision on (precision <=18)	24
Decimal, high precision on (precision >18, <=28)	32
Decimal, high precision on (precision >28)	16
Decimal, high precision on (negative scale)	16
Double	16
Real	16
Integer	16
Small integer	16
Bigint	64
NString, NText, String, Text	Unicode mode: 2*(precision + 5) ASCII mode: precision + 9

The column sizes include the bytes required for a null indicator.

To increase performance for the sort operation, the Integration Service aligns all data for the Sorter transformation memory on an 8-byte boundary. Each Sorter column includes rounding to the nearest multiple of eight.

The Integration Service also writes the row size and amount of memory the Sorter transformation uses to the session log when you configure the Sorter transformation tracing level to Normal.

Case Sensitive

The Case Sensitive property determines whether the Integration Service considers case when sorting data.

When you enable the Case Sensitive property, the Integration Service sorts uppercase characters higher than lowercase characters.

Work Directory

You must specify a work directory the Integration Service uses to create temporary files while it sorts data. After the Integration Service sorts the data, it deletes the temporary files. You can specify any directory on the Integration Service machine to use as a work directory. By default, the Integration Service uses the value specified for the \$PMTempDir process variable.

When you partition a session with a Sorter transformation, you can specify a different work directory for each partition in the pipeline. To increase session performance, specify work directories on physically separate disks on the Integration Service system.

Distinct Output Rows

You can configure the Sorter transformation to treat output rows as distinct. If you configure the Sorter transformation for distinct output rows, the Mapping Designer configures all ports as part of the sort key. The Integration Service discards duplicate rows compared during the sort operation.

Tracing Level

Configure the Sorter transformation tracing level to control the number and type of Sorter error and status messages the Integration Service writes to the session log. At Normal tracing level, the Integration Service writes the size of the row passed to the Sorter transformation and the amount of memory the Sorter transformation allocates for the sort operation. The Integration Service also writes the time and date when it passes the first and last input rows to the Sorter transformation.

If you configure the Sorter transformation tracing level to Verbose Data, the Integration Service writes the time the Sorter transformation finishes passing all data to the next transformation in the pipeline. The Integration Service also writes the time to the session log when the Sorter transformation releases memory resources and removes temporary files from the work directory.

Null Treated Low

You can configure the way the Sorter transformation treats null values. Enable this property if you want the Integration Service to treat null values as lower than any other value when it performs the sort operation. Disable this option if you want the Integration Service to treat null values as higher than any other value.

Transformation Scope

The transformation scope specifies how the Integration Service applies the transformation logic to incoming data:

- φ **Transaction.** Applies the transformation logic to all rows in a transaction. Choose Transaction when a row of data depends on all rows in the same transaction, but does not depend on rows in other transactions.

- φ **All Input.** Applies the transformation logic on all incoming data. When you choose All Input, the PowerCenter drops incoming transaction boundaries. Choose All Input when a row of data depends on all rows in the source.

Creating a Sorter Transformation

To add a Sorter transformation to a mapping, complete the following steps.

To create a Sorter transformation:

1. In the Mapping Designer, click Transformation > Create. Select the Sorter transformation.
The naming convention for Sorter transformations is *SRT_TransformationName*. Enter a description for the transformation. This description appears in the Repository Manager, making it easier to understand what the transformation does.
2. Enter a name for the Sorter and click Create.
The Designer creates the Sorter transformation.
3. Click Done.
4. Drag the ports you want to sort into the Sorter transformation.
The Designer creates the input/output ports for each port you include.
5. Double-click the title bar of the transformation to open the Edit Transformations dialog box.
6. Select the Ports tab.
7. Select the ports you want to use as the sort key.
8. For each port selected as part of the sort key, specify whether you want the Integration Service to sort data in ascending or descending order.
9. Select the Properties tab. Modify the Sorter transformation properties.
10. Select the Metadata Extensions tab. Create or edit metadata extensions for the Sorter transformation.
11. Click OK.

CHAPTER 15

Source Qualifier Transformation

This chapter includes the following topics:

- ϕ Overview
- ϕ Source Qualifier Transformation Properties
- ϕ Default Query
- ϕ Joining Source Data
- ϕ Adding an SQL Query
- ϕ Entering a User-Defined Join
- ϕ Outer Join Support
- ϕ Entering a Source Filter
- ϕ Using Sorted Ports
- ϕ Select Distinct
- ϕ Adding Pre- and Post-Session SQL Commands
- ϕ Creating a Source Qualifier Transformation
- ϕ Troubleshooting

Overview

Transformation type:

Active
Connected

When you add a relational or a flat file source definition to a mapping, you need to connect it to a Source Qualifier transformation. The Source Qualifier transformation represents the rows that the Integration Service reads when it runs a session.

Use the Source Qualifier transformation to complete the following tasks:

- ϕ **Join data originating from the same source database.** You can join two or more tables with primary key-foreign key relationships by linking the sources to one Source Qualifier transformation.
- ϕ **Filter rows when the Integration Service reads source data.** If you include a filter condition, the Integration Service adds a WHERE clause to the default query.
- ϕ **Specify an outer join rather than the default inner join.** If you include a user-defined join, the Integration Service replaces the join information specified by the metadata in the SQL query.

- φ **Specify sorted ports.** If you specify a number for sorted ports, the Integration Service adds an ORDER BY clause to the default SQL query.
- φ **Select only distinct values from the source.** If you choose Select Distinct, the Integration Service adds a SELECT DISTINCT statement to the default SQL query.
- φ **Create a custom query to issue a special SELECT statement for the Integration Service to read source data.** For example, you might use a custom query to perform aggregate calculations.

Transformation Datatypes

The Source Qualifier transformation displays the transformation datatypes. The transformation datatypes determine how the source database binds data when the Integration Service reads it. Do not alter the datatypes in the Source Qualifier transformation. If the datatypes in the source definition and Source Qualifier transformation do not match, the Designer marks the mapping invalid when you save it.

Target Load Order

You specify a target load order based on the Source Qualifier transformations in a mapping. If you have multiple Source Qualifier transformations connected to multiple targets, you can designate the order in which the Integration Service loads data into the targets.

If one Source Qualifier transformation provides data for multiple targets, you can enable constraint-based loading in a session to have the Integration Service load data based on target table primary and foreign key relationships.

Datetime Values

When you use a datetime value or a datetime parameter or variable in the SQL query, change the date format to the format used in the source. The Integration Service passes datetime values to source systems as strings in the SQL query. The Integration Service converts a datetime value to a string, based on the source database.

The following table describes the datetime formats for each database type:

Source	Date Format
DB2	YYYY-MM-DD-HH24:MI:SS
Informix	YYYY-MM-DD HH24:MI:SS
Microsoft SQL Server	MM/DD/YYYY HH24:MI:SS
ODBC	YYYY-MM-DD HH24:MI:SS
Oracle	MM/DD/YYYY HH24:MI:SS
Sybase	MM/DD/YYYY HH24:MI:SS
Teradata	YYYY-MM-DD HH24:MI:SS

Some databases require you to identify datetime values with additional punctuation, such as single quotation marks or database specific functions. For example, to convert the \$\$\$SessStartTime value for an Oracle source, use the following Oracle function in the SQL override:

```
to_date ('$$$SessStartTime', 'mm/dd/yyyy hh24:mi:ss')
```

For Informix, use the following Informix function in the SQL override to convert the \$\$\$SessStartTime value:

```
DATE TIME ($$$SessStartTime) YEAR TO SECOND
```

For information about database specific functions, see the database documentation.

Parameters and Variables

You can use parameters and variables in the SQL query, user-defined join, source filter, and pre- and post-session SQL commands of a Source Qualifier transformation. Use any parameter or variable type that you can define in the parameter file. You can enter a parameter or variable within the SQL statement, or you can use a parameter or variable as the SQL query. For example, you can use a session parameter, `$ParamMyQuery`, as the SQL query, and set `$ParamMyQuery` to the SQL statement in a parameter file.

The Integration Service first generates an SQL query and expands each parameter or variable. It replaces each mapping parameter, mapping variable, and workflow variable with its start value. Then it runs the query on the source database.

When you use a string mapping parameter or variable in the Source Qualifier transformation, use a string identifier appropriate to the source system. Most databases use a single quotation mark as a string identifier. For example, to use the string parameter `$$IPAddress` in a source filter for a Microsoft SQL Server database table, enclose the parameter in single quotes as follows: `'$$IPAddress'`.

When you use a datetime mapping parameter or variable, or when you use the built-in variable `$$$SessStartTime`, change the date format to the format used in the source. The Integration Service passes datetime values to source systems as strings in the SQL query.

Tip: To ensure the format of a datetime parameter or variable matches that used by the source, validate the SQL query.

Source Qualifier Transformation Properties

Configure the following Source Qualifier transformation properties on the Properties tab:

Option	Description
SQL Query	Defines a custom query that replaces the default query the Integration Service uses to read data from sources represented in this Source Qualifier transformation. A custom query overrides entries for a custom join or a source filter.
User-Defined Join	Specifies the condition used to join data from multiple sources represented in the same Source Qualifier transformation.
Source Filter	Specifies the filter condition the Integration Service applies when querying rows.
Number of Sorted Ports	Indicates the number of columns used when sorting rows queried from relational sources. If you select this option, the Integration Service adds an ORDER BY to the default query when it reads source rows. The ORDER BY includes the number of ports specified, starting from the top of the transformation. When selected, the database sort order must match the session sort order.
Tracing Level	Sets the amount of detail included in the session log when you run a session containing this transformation.
Select Distinct	Specifies if you want to select only unique rows. The Integration Service includes a SELECT DISTINCT statement if you choose this option.
Pre-SQL	Pre-session SQL commands to run against the source database before the Integration Service reads the source.
Post-SQL	Post-session SQL commands to run against the source database after the Integration Service writes to the target.

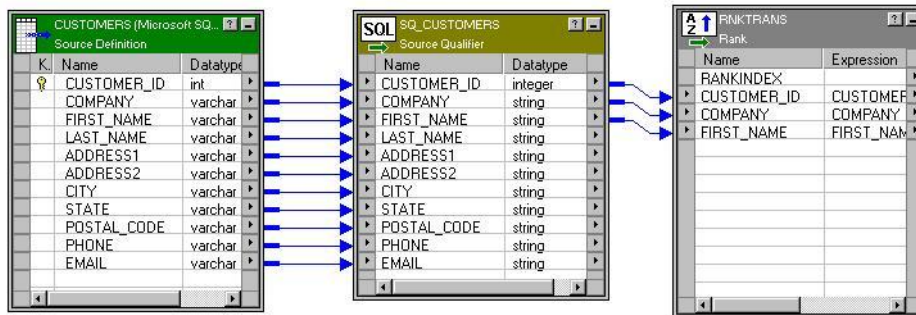
Option	Description
Output is Deterministic	Relational source or transformation output that does not change between session runs when the input data is consistent between runs. When you configure this property, the Integration Service does not stage source data for recovery if transformations in the pipeline always produce repeatable data.
Output is Repeatable	Relational source or transformation output that is in the same order between session runs when the order of the input data is consistent. When output is deterministic and output is repeatable, the Integration Service does not stage source data for recovery.

Warning: If you configure a transformation as repeatable and deterministic, it is your responsibility to ensure that the data is repeatable and deterministic. If you try to recover a session with transformations that do not produce the same data between the session and the recovery, the recovery process can result in corrupted data.

Default Query

For relational sources, the Integration Service generates a query for each Source Qualifier transformation when it runs a session. The default query is a SELECT statement for each source column used in the mapping. In other words, the Integration Service reads only the columns that are connected to another transformation.

The following figure shows a single source definition connected to a Source Qualifier transformation:



Although there are many columns in the source definition, only three columns are connected to another transformation. In this case, the Integration Service generates a default query that selects only those three columns:

```
SELECT CUSTOMERS.CUSTOMER_ID, CUSTOMERS.COMPANY, CUSTOMERS.FIRST_NAME
FROM CUSTOMERS
```

If any table name or column name contains a database reserved word, you can create and maintain a file, `reswords.txt`, containing reserved words. When the Integration Service initializes a session, it searches for `reswords.txt` in the Integration Service installation directory. If the file exists, the Integration Service places quotes around matching reserved words when it executes SQL against the database. If you override the SQL, you must enclose any reserved word in quotes.

When generating the default query, the Designer delimits table and field names containing the following characters with double quotes:

```
/ + - = ~ ` ! % ^ & * ( ) [ ] { } ' ; ? , < > \ | <space>
```

Viewing the Default Query

You can view the default query in the Source Qualifier transformation.

To view the default query:

1. From the Properties tab, select SQL Query.

The SQL Editor displays the default query the Integration Service uses to select source data.

2. Click Generate SQL.
3. Click Cancel to exit.

Note: If you do not cancel the SQL query, the Integration Service overrides the default query with the custom SQL query.

Do not connect to the source database. You only connect to the source database when you enter an SQL query that overrides the default query.

You must connect the columns in the Source Qualifier transformation to another transformation or target before you can generate the default query.

Overriding the Default Query

You can alter or override the default query in the Source Qualifier transformation by changing the default settings of the transformation properties. Do not change the list of selected ports or the order in which they appear in the query. This list must match the connected transformation output ports.

When you edit transformation properties, the Source Qualifier transformation includes these settings in the default query. However, if you enter an SQL query, the Integration Service uses only the defined SQL statement. The SQL Query overrides the User-Defined Join, Source Filter, Number of Sorted Ports, and Select Distinct settings in the Source Qualifier transformation.

Note: When you override the default SQL query, you must enclose all database reserved words in quotes.

Joining Source Data

Use one Source Qualifier transformation to join data from multiple relational tables. These tables must be accessible from the same instance or database server.

When a mapping uses related relational sources, you can join both sources in one Source Qualifier transformation. During the session, the source database performs the join before passing data to the Integration Service. This can increase performance when source tables are indexed.

Tip: Use the Joiner transformation for heterogeneous sources and to join flat files.

Default Join

When you join related tables in one Source Qualifier transformation, the Integration Service joins the tables based on the related keys in each table.

This default join is an inner equijoin, using the following syntax in the WHERE clause:

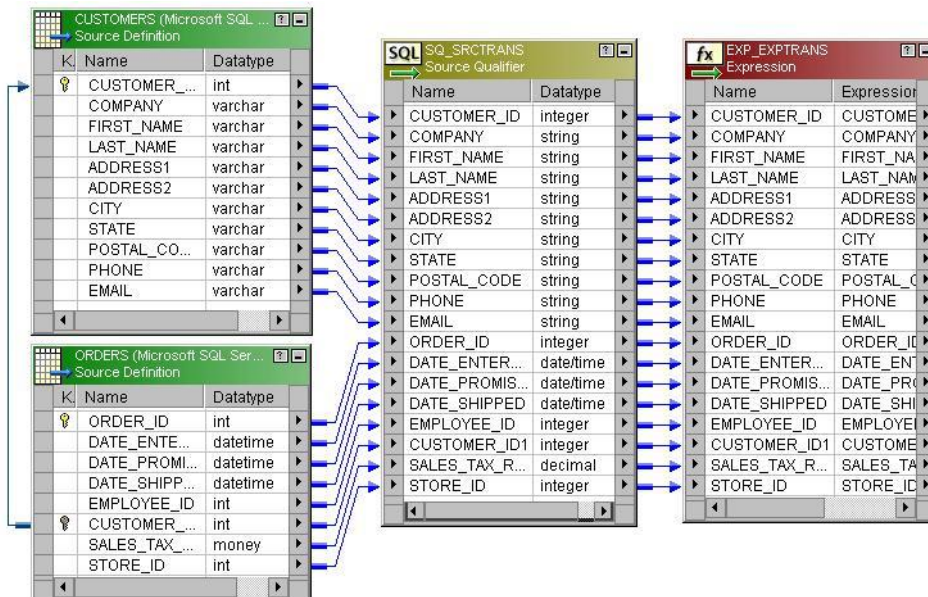
```
Source1.column_name = Source2.column_name
```

The columns in the default join must have:

- ϕ A primary key-foreign key relationship
- ϕ Matching datatypes

For example, you might see all the orders for the month, including order number, order amount, and customer name. The ORDERS table includes the order number and amount of each order, but not the customer name. To include the customer name, you need to join the ORDERS and CUSTOMERS tables. Both tables include a customer ID, so you can join the tables in one Source Qualifier transformation.

The following figure shows joining two tables with one Source Qualifier transformation:



When you include multiple tables, the Integration Service generates a SELECT statement for all columns used in the mapping. In this case, the SELECT statement looks similar to the following statement:

```
SELECT CUSTOMERS.CUSTOMER_ID, CUSTOMERS.COMPANY, CUSTOMERS.FIRST_NAME,
CUSTOMERS.LAST_NAME, CUSTOMERS.ADDRESS1, CUSTOMERS.ADDRESS2, CUSTOMERS.CITY,
CUSTOMERS.STATE, CUSTOMERS.POSTAL_CODE, CUSTOMERS.PHONE, CUSTOMERS.EMAIL,
ORDERS.ORDER_ID, ORDERS.DATE_ENTERED, ORDERS.DATE_PROMISED, ORDERS.DATE_SHIPPED,
ORDERS.EMPLOYEE_ID, ORDERS.CUSTOMER_ID, ORDERS.SALES_TAX_RATE, ORDERS.STORE_ID
FROM CUSTOMERS, ORDERS
WHERE CUSTOMERS.CUSTOMER_ID=ORDERS.CUSTOMER_ID
```

The WHERE clause is an equijoin that includes the CUSTOMER_ID from the ORDERS and CUSTOMER tables.

Custom Joins

If you need to override the default join, you can enter contents of the WHERE clause that specifies the join in the custom query. If the query performs an outer join, the Integration Service may insert the join syntax in the WHERE clause or the FROM clause, depending on the database syntax.

You might need to override the default join under the following circumstances:

- φ Columns do not have a primary key-foreign key relationship.
- φ The datatypes of columns used for the join do not match.
- φ You want to specify a different type of join, such as an outer join.

Heterogeneous Joins

To perform a heterogeneous join, use the Joiner transformation. Use the Joiner transformation when you need to join the following types of sources:

- φ Join data from different source databases
- φ Join data from different flat file systems

φ Join relational sources and flat files

Creating Key Relationships

You can join tables in the Source Qualifier transformation if the tables have primary key-foreign key relationships. However, you can create primary key-foreign key relationships in the Source Analyzer by linking matching columns in different tables. These columns do not have to be keys, but they should be included in the index for each table.

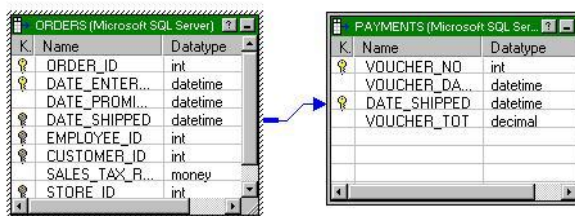
Tip: If the source table has more than 1,000 rows, you can increase performance by indexing the primary key-foreign keys. If the source table has fewer than 1,000 rows, you might decrease performance if you index the primary key-foreign keys.

For example, the corporate office for a retail chain wants to extract payments received based on orders. The ORDERS and PAYMENTS tables do not share primary and foreign keys. Both tables, however, include a DATE_SHIPPED column. You can create a primary key-foreign key relationship in the metadata in the Source Analyzer.

Note, the two tables are not linked. Therefore, the Designer does not recognize the relationship on the DATE_SHIPPED columns.

You create a relationship between the ORDERS and PAYMENTS tables by linking the DATE_SHIPPED columns. The Designer adds primary and foreign keys to the DATE_SHIPPED columns in the ORDERS and PAYMENTS table definitions.

The following figure shows a relationship between two tables:



If you do not connect the columns, the Designer does not recognize the relationships.

The primary key-foreign key relationships exist in the metadata only. You do not need to generate SQL or alter the source tables.

Once the key relationships exist, use a Source Qualifier transformation to join the two tables. The default join is based on DATE_SHIPPED.

Adding an SQL Query

The Source Qualifier transformation provides the SQL Query option to override the default query. You can enter an SQL statement supported by the source database. Before entering the query, connect all the input and output ports you want to use in the mapping.

When you edit the SQL Query, you can generate and edit the default query. When the Designer generates the default query, it incorporates all other configured options, such as a filter or number of sorted ports. The resulting query overrides all other options you might subsequently configure in the transformation.

You can use a parameter or variable as the SQL query or include parameters and variables within the query.

When including a string mapping parameter or variable, use a string identifier appropriate to the source system. For most databases, you need to enclose the name of a string parameter or variable in single quotes.

When you include a datetime value or a datetime mapping parameter or variable in the SQL query, change the date format to match the format used by the source. The Integration Service converts a datetime value to a string based on the source system. For more information about date conversion, see “Datetime Values” on page 296.

When creating a custom SQL query, the SELECT statement must list the port names in the order in which they appear in the transformation.

When you override the default SQL query for a session configured for pushdown optimization, the Integration Service creates a view to represent the SQL override. It then runs an SQL query against this view to push the transformation logic to the database.

If you edit the SQL query, you must enclose all database reserved words in quotes.

To override the default query:

1. Open the Source Qualifier transformation, and click the Properties tab.
2. Click the Open button in the SQL Query field.

The SQL Editor dialog box appears.

3. Click Generate SQL.

The Designer displays the default query it generates when querying rows from all sources included in the Source Qualifier transformation.

4. Enter a query in the space where the default query appears.

Every column name must be qualified by the name of the table, view, or synonym in which it appears. For example, if you want to include the ORDER_ID column from the ORDERS table, enter ORDERS.ORDER_ID. You can double-click column names appearing in the Ports window to avoid typing the name of every column.

You can use a parameter or variable as the query, or you can include parameters and variables in the query.

Enclose string mapping parameters and variables in string identifiers. Alter the date format for datetime mapping parameters and variables when necessary.

5. Select the ODBC data source containing the sources included in the query.
6. Enter the user name and password to connect to this database.
7. Click Validate.

The Designer runs the query and reports whether its syntax was correct.

8. Click OK to return to the Edit Transformations dialog box. Click OK again to return to the Designer.

Tip: You can resize the Expression Editor. Expand the dialog box by dragging from the borders. The Designer saves the new size for the dialog box as a client setting.

Entering a User-Defined Join

Entering a user-defined join is similar to entering a custom SQL query. However, you only enter the contents of the WHERE clause, not the entire query. When you perform an outer join, the Integration Service may insert the join syntax in the WHERE clause or the FROM clause of the query, depending on the database syntax.

When you add a user-defined join, the Source Qualifier transformation includes the setting in the default SQL query. However, if you modify the default query after adding a user-defined join, the Integration Service uses only the query defined in the SQL Query property of the Source Qualifier transformation.

You can use a parameter or variable as the user-defined join or include parameters and variables within the join. When including a string mapping parameter or variable, use a string identifier appropriate to the source system. For most databases, you need to enclose the name of a string parameter or variable in single quotes.

When you include a datetime parameter or variable, you might need to change the date format to match the format used by the source. The Integration Service converts a datetime parameter and variable to a string based on the source system. For more information about automatic date conversion, see “Datetime Values” on page 296.

To create a user-defined join:

1. Create a Source Qualifier transformation containing data from multiple sources or associated sources.
2. Open the Source Qualifier transformation, and click the Properties tab.
3. Click the Open button in the User Defined Join field.

The SQL Editor dialog box appears.

4. Enter the syntax for the join.

Do not enter the keyword WHERE at the beginning of the join. The Integration Service adds this keyword when it queries rows.

Enclose string mapping parameters and variables in string identifiers. Alter the date format for datetime mapping parameters and variables when necessary.

5. Click OK to return to the Edit Transformations dialog box, and then click OK to return to the Designer.

Outer Join Support

Use the Source Qualifier and the Application Source Qualifier transformations to perform an outer join of two sources in the same database. When the Integration Service performs an outer join, it returns all rows from one source table and rows from the second source table that match the join condition.

Use an outer join when you want to join two tables and return all rows from one of the tables. For example, you might perform an outer join when you want to join a table of registered customers with a monthly purchases table to determine registered customer activity. Using an outer join, you can join the registered customer table with the monthly purchases table and return all rows in the registered customer table, including customers who did not make purchases in the last month. If you perform a normal join, the Integration Service returns only registered customers who made purchases during the month, and only purchases made by registered customers.

With an outer join, you can generate the same results as a master outer or detail outer join in the Joiner transformation. However, when you use an outer join, you reduce the number of rows in the data flow. This can improve performance.

The Integration Service supports two kinds of outer joins:

- ϕ **Left.** Integration Service returns all rows for the table to the left of the join syntax and the rows from both tables that meet the join condition.
- ϕ **Right.** Integration Service returns all rows for the table to the right of the join syntax and the rows from both tables that meet the join condition.

Note: Use outer joins in nested query statements when you override the default query.

Informatica Join Syntax

When you enter join syntax, use the Informatica or database-specific join syntax. When you use the Informatica join syntax, the Integration Service translates the syntax and passes it to the source database during the session.

Note: Always use database-specific syntax for join conditions.

When you use Informatica join syntax, enclose the entire join statement in braces ({Informatica syntax}). When you use database syntax, enter syntax supported by the source database without braces.

When using Informatica join syntax, use table names to prefix column names. For example, if you have a column named FIRST_NAME in the REG_CUSTOMER table, enter “REG_CUSTOMER.FIRST_NAME” in the join syntax. Also, when using an alias for a table name, use the alias within the Informatica join syntax to ensure the Integration Service recognizes the alias.

The following table lists the join syntax you can enter, in different locations for different Source Qualifier transformations, when you create an outer join:

Transformation	Transformation Setting	Description
Source Qualifier Transformation	User-Defined Join	Create a join override. The Integration Service appends the join override to the WHERE or FROM clause of the default query.
	SQL Query	Enter join syntax immediately after the WHERE in the default query.
Application Source Qualifier Transformation	Join Override	Create a join override. The Integration Service appends the join override to the WHERE clause of the default query.
	Extract Override	Enter join syntax immediately after the WHERE in the default query.

You can combine left outer and right outer joins with normal joins in a single source qualifier. Use multiple normal joins and multiple left outer joins.

When you combine joins, enter them in the following order:

1. Normal
2. Left outer
3. Right outer

Note: Some databases limit you to using one right outer join.

Normal Join Syntax

You can create a normal join using the join condition in a source qualifier. However, if you are creating an outer join, you need to override the default join to perform an outer join. As a result, you need to include the normal join in the join override. When incorporating a normal join in the join override, list the normal join before outer joins. You can enter multiple normal joins in the join override.

To create a normal join, use the following syntax:

```
{ source1 INNER JOIN source2 on join_condition }
```

The following table displays the syntax for Normal Joins in a Join Override:

Syntax	Description
<i>source1</i>	Source table name. The Integration Service returns rows from this table that match the join condition.
<i>source2</i>	Source table name. The Integration Service returns rows from this table that match the join condition.
<i>join_condition</i>	Condition for the join. Use syntax supported by the source database. You can combine multiple join conditions with the AND operator.

For example, you have a REG_CUSTOMER table with data for registered customers:

CUST_ID	FIRST_NAME	LAST_NAME
00001	Marvin	Chi

CUST_ID	FIRST_NAME	LAST_NAME
00002	Dinah	Jones
00003	John	Bowden
00004	J.	Marks

The PURCHASES table, refreshed monthly, contains the following data:

TRANSACTION_NO	CUST_ID	DATE	AMOUNT
06-2000-0001	00002	6/3/2000	55.79
06-2000-0002	00002	6/10/2000	104.45
06-2000-0003	00001	6/10/2000	255.56
06-2000-0004	00004	6/15/2000	534.95
06-2000-0005	00002	6/21/2000	98.65
06-2000-0006	NULL	6/23/2000	155.65
06-2000-0007	NULL	6/24/2000	325.45

To return rows displaying customer names for each transaction in the month of June, use the following syntax:

```
{ REG_CUSTOMER INNER JOIN PURCHASES on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID }
```

The Integration Service returns the following data:

CUST_ID	DATE	AMOUNT	FIRST_NAME	LAST_NAME
00002	6/3/2000	55.79	Dinah	Jones
00002	6/10/2000	104.45	Dinah	Jones
00001	6/10/2000	255.56	Marvin	Chi
00004	6/15/2000	534.95	J.	Marks
00002	6/21/2000	98.65	Dinah	Jones

The Integration Service returns rows with matching customer IDs. It does not include customers who made no purchases in June. It also does not include purchases made by non-registered customers.

Left Outer Join Syntax

You can create a left outer join with a join override. You can enter multiple left outer joins in a single join override. When using left outer joins with other joins, list all left outer joins together, after any normal joins in the statement.

To create a left outer join, use the following syntax:

```
{ source1 LEFT OUTER JOIN source2 on join_condition }
```

The following tables displays syntax for left outer joins in a join override:

Syntax	Description
<i>source1</i>	Source table name. With a left outer join, the Integration Service returns all rows in this table.
<i>source2</i>	Source table name. The Integration Service returns rows from this table that match the join condition.
<i>join_condition</i>	Condition for the join. Use syntax supported by the source database. You can combine multiple join conditions with the AND operator.

For example, using the same REG_CUSTOMER and PURCHASES tables described in “Normal Join Syntax” , you can determine how many customers bought something in June with the following join override:


```
{ REG_CUSTOMER LEFT OUTER JOIN PURCHASES on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID }
```

The Integration Service returns the following data:

CUST_ID	FIRST_NAME	LAST_NAME	DATE	AMOUNT
00001	Marvin	Chi	6/10/2000	255.56
00002	Dinah	Jones	6/3/2000	55.79
00003	John	Bowden	NULL	NULL
00004	J.	Marks	6/15/2000	534.95
00002	Dinah	Jones	6/10/2000	104.45
00002	Dinah	Jones	6/21/2000	98.65

The Integration Service returns all registered customers in the REG_CUSTOMERS table, using null values for the customer who made no purchases in June. It does not include purchases made by non-registered customers.

Use multiple join conditions to determine how many registered customers spent more than \$100.00 in a single purchase in June:

```
{ REG_CUSTOMER LEFT OUTER JOIN PURCHASES on (REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID AND  
PURCHASES.AMOUNT > 100.00) }
```

The Integration Service returns the following data:

CUST_ID	FIRST_NAME	LAST_NAME	DATE	AMOUNT
00001	Marvin	Chi	6/10/2000	255.56
00002	Dinah	Jones	6/10/2000	104.45
00003	John	Bowden	NULL	NULL
00004	J.	Marks	6/15/2000	534.95

You might use multiple left outer joins if you want to incorporate information about returns during the same time period. For example, the RETURNS table contains the following data:

CUST_ID	CUST_ID	RETURN
00002	6/10/2000	55.79
00002	6/21/2000	104.45

To determine how many customers made purchases and returns for the month of June, use two left outer joins:

```
{ REG_CUSTOMER LEFT OUTER JOIN PURCHASES on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID LEFT  
OUTER JOIN RETURNS on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID }
```

The Integration Service returns the following data:

CUST_ID	FIRST_NAME	LAST_NAME	DATE	AMOUNT	RET_DATE	RETURN
00001	Marvin	Chi	6/10/2000	255.56	NULL	NULL
00002	Dinah	Jones	6/3/2000	55.79	NULL	NULL
00003	John	Bowden	NULL	NULL	NULL	NULL
00004	J.	Marks	6/15/2000	534.95	NULL	NULL
00002	Dinah	Jones	6/10/2000	104.45	NULL	NULL
00002	Dinah	Jones	6/21/2000	98.65	NULL	NULL
00002	Dinah	Jones	NULL	NULL	6/10/2000	55.79
00002	Dinah	Jones	NULL	NULL	6/21/2000	104.45

The Integration Service uses NULLs for missing values.

Right Outer Join Syntax

You can create a right outer join with a join override. The right outer join returns the same results as a left outer join if you reverse the order of the tables in the join syntax. Use only one right outer join in a join override. If you want to create more than one right outer join, try reversing the order of the source tables and changing the join types to left outer joins.

When you use a right outer join with other joins, enter the right outer join at the end of the join override.

To create a right outer join, use the following syntax:

```
{ source1 RIGHT OUTER JOIN source2 on join_condition }
```

The following table displays syntax for a right outer join in a join override:

Syntax	Description
<i>source1</i>	Source table name. The Integration Service returns rows from this table that match the join condition.
<i>source2</i>	Source table name. With a right outer join, the Integration Service returns all rows in this table.
<i>join_condition</i>	Condition for the join. Use syntax supported by the source database. You can combine multiple join conditions with the AND operator.

You might use a right outer join with a left outer join to join and return all data from both tables, simulating a full outer join. For example, you can extract all registered customers and all purchases for the month of June with the following join override:

```
{REG_CUSTOMER LEFT OUTER JOIN PURCHASES on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID RIGHT  
OUTER JOIN PURCHASES on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID }
```

The Integration Service returns the following data:

CUST_ID	FIRST_NAME	LAST_NAME	TRANSACTION_NO	DATE	AMOUNT
00001	Marvin	Chi	06-2000-0003	6/10/2000	255.56
00002	Dinah	Jones	06-2000-0001	6/3/2000	55.79
00003	John	Bowden	NULL	NULL	NULL
00004	J.	Marks	06-2000-0004	6/15/2000	534.95
00002	Dinah	Jones	06-2000-0002	6/10/2000	104.45
00002	Dinah	Jones	06-2000-0005	6/21/2000	98.65
NULL	NULL	NULL	06-2000-0006	6/23/2000	155.65
NULL	NULL	NULL	06-2000-0007	6/24/2000	325.45

Creating an Outer Join

You can enter an outer join as a join override or as part of an override of the default query.

When you create a join override, the Designer appends the join override to the WHERE clause of the default query. During the session, the Integration Service translates the Informatica join syntax and includes it in the default query used to extract source data. When possible, enter a join override instead of overriding the default query.

When you override the default query, enter the join syntax in the WHERE clause of the default query. During the session, the Integration Service translates Informatica join syntax and then uses the query to extract source data. If you make changes to the transformation after creating the override, the Integration Service ignores the changes. Therefore, when possible, enter outer join syntax as a join override.

To create an outer join as a join override:

1. Open the Source Qualifier transformation, and click the Properties tab.
2. In a Source Qualifier transformation, click the button in the User Defined Join field.
In an Application Source Qualifier transformation, click the button in the Join Override field.
3. Enter the syntax for the join.
Do not enter WHERE at the beginning of the join. The Integration Service adds this when querying rows.
Enclose Informatica join syntax in braces ({ }).
When using an alias for a table and the Informatica join syntax, use the alias within the Informatica join syntax.
Use table names to prefix columns names, for example, "table.column".
Use join conditions supported by the source database.
When entering multiple joins, group joins together by type, and then list them in the following order: normal, left outer, right outer. Include only one right outer join per nested query.
Select port names from the Ports tab to ensure accuracy.
4. Click OK.

To create an outer join as an extract override:

1. After connecting the input and output ports for the Application Source Qualifier transformation, double-click the title bar of the transformation and select the Properties tab.
2. In an Application Source Qualifier transformation, click the button in the Extract Override field.
3. Click Generate SQL.
4. Enter the syntax for the join in the WHERE clause immediately after the WHERE.
Enclose Informatica join syntax in braces ({ }).
When using an alias for a table and the Informatica join syntax, use the alias within the Informatica join syntax.
Use table names to prefix columns names, for example, "table.column".
Use join conditions supported by the source database.
When entering multiple joins, group joins together by type, and then list them in the following order: normal, left outer, right outer. Include only one right outer join per nested query.
Select port names from the Ports tab to ensure accuracy.
5. Click OK.

Common Database Syntax Restrictions

Different databases have different restrictions on outer join syntax. Consider the following restrictions when you create outer joins:

- ϕ Do not combine join conditions with the OR operator in the ON clause of outer join syntax.
- ϕ Do not use the IN operator to compare columns in the ON clause of outer join syntax.
- ϕ Do not compare a column to a subquery in the ON clause of outer join syntax.
- ϕ When combining two or more outer joins, do not use the same table as the inner table of more than one outer join. For example, do not use either of the following outer joins:

```
{ TABLE1 LEFT OUTER JOIN TABLE2 ON TABLE1.COLUMNA = TABLE2.COLUMNA TABLE3 LEFT OUTER JOIN  
TABLE2 ON TABLE3.COLUMNB = TABLE2.COLUMNB }  
{ TABLE1 LEFT OUTER JOIN TABLE2 ON TABLE1.COLUMNA = TABLE2.COLUMNA TABLE2 RIGHT OUTER  
JOIN TABLE3 ON TABLE2.COLUMNB = TABLE3.COLUMNB }
```

- ϕ Do not use both tables of an outer join in a regular join condition. For example, do not use the following join condition:

```
{ TABLE1 LEFT OUTER JOIN TABLE2 ON TABLE1.COLUMNNA = TABLE2.COLUMNNA WHERE TABLE1.COLUMNB =  
TABLE2.COLUMNNC }
```

However, use both tables in a filter condition, like the following:

```
{ TABLE1 LEFT OUTER JOIN TABLE2 ON TABLE1.COLUMNNA = TABLE2.COLUMNNA WHERE TABLE1.COLUMNB =  
32 AND TABLE2.COLUMNNC > 0 }
```

Note: Entering a condition in the ON clause might return different results from entering the same condition in the WHERE clause.

- ϕ When using an alias for a table, use the alias to prefix columns in the table. For example, if you call the REG_CUSTOMER table C, when referencing the column FIRST_NAME, use “C.FIRST_NAME.”

Entering a Source Filter

You can enter a source filter to reduce the number of rows the Integration Service queries. If you include the string ‘WHERE’ or large objects in the source filter, the Integration Service fails the session.

The Source Qualifier transformation includes source filters in the default SQL query. If, however, you modify the default query after adding a source filter, the Integration Service uses only the query defined in the SQL query portion of the Source Qualifier transformation.

You can use a parameter or variable as the source filter or include parameters and variables within the source filter. When including a string mapping parameter or variable, use a string identifier appropriate to the source system. For most databases, you need to enclose the name of a string parameter or variable in single quotes.

When you include a datetime parameter or variable, you might need to change the date format to match the format used by the source. The Integration Service converts a datetime parameter and variable to a string based on the source system.

Note: When you enter a source filter in the session properties, you override the customized SQL query in the Source Qualifier transformation.

To enter a source filter:

1. In the Mapping Designer, open a Source Qualifier transformation.

The Edit Transformations dialog box appears.

2. Select the Properties tab.
3. Click the Open button in the Source Filter field.
4. In the SQL Editor dialog box, enter the filter.

Include the table name and port name. Do not include the keyword WHERE in the filter.

Enclose string mapping parameters and variables in string identifiers. Alter the date format for datetime mapping parameters and variables when necessary.

5. Click OK.

Using Sorted Ports

When you use sorted ports, the Integration Service adds the ports to the ORDER BY clause in the default query. The Integration Service adds the configured number of ports, starting at the top of the Source Qualifier

transformation. You might use sorted ports to improve performance when you include any of the following transformations in a mapping:

- φ **Aggregator.** When you configure an Aggregator transformation for sorted input, you can send sorted data by using sorted ports. The group by ports in the Aggregator transformation must match the order of the sorted ports in the Source Qualifier transformation.
- φ **Joiner.** When you configure a Joiner transformation for sorted input, you can send sorted data by using sorted ports. Configure the order of the sorted ports the same in each Source Qualifier transformation.

Note: You can also use the Sorter transformation to sort relational and flat file data before Aggregator and Joiner transformations.

Use sorted ports for relational sources only. When using sorted ports, the sort order of the source database must match the sort order configured for the session. The Integration Service creates the SQL query used to extract source data, including the ORDER BY clause for sorted ports. The database server performs the query and passes the resulting data to the Integration Service. To ensure data is sorted as the Integration Service requires, the database sort order must be the same as the user-defined session sort order.

When you configure the Integration Service for data code page validation and run a workflow in Unicode data movement mode, the Integration Service uses the selected sort order to sort character data.

When you configure the Integration Service for relaxed data code page validation, the Integration Service uses the selected sort order to sort all character data that falls in the language range of the selected sort order. The Integration Service sorts all character data outside the language range of the selected sort order according to standard Unicode sort ordering.

When the Integration Service runs in ASCII mode, it ignores this setting and sorts all character data using a binary sort order. The default sort order depends on the code page of the Integration Service.

The Source Qualifier transformation includes the number of sorted ports in the default SQL query. However, if you modify the default query after choosing the Number of Sorted Ports, the Integration Service uses only the query defined in the SQL Query property.

To use sorted ports:

1. In the Mapping Designer, open a Source Qualifier transformation, and click the Properties tab.
2. Click in Number of Sorted Ports and enter the number of ports you want to sort.

The Integration Service adds the configured number of columns to an ORDER BY clause, starting from the top of the Source Qualifier transformation.

The source database sort order must correspond to the session sort order.

Tip: Sybase supports a maximum of 16 columns in an ORDER BY clause. If the source is Sybase, do not sort more than 16 columns.

3. Click OK.

Select Distinct

If you want the Integration Service to select unique values from a source, use the Select Distinct option. You might use this feature to extract unique customer IDs from a table listing total sales. Using Select Distinct filters out unnecessary data earlier in the data flow, which might improve performance.

By default, the Designer generates a SELECT statement. If you choose Select Distinct, the Source Qualifier transformation includes the setting in the default SQL query.

For example, in the Source Qualifier transformation in “Joining Source Data”, you enable the Select Distinct option. The Designer adds SELECT DISTINCT to the default query as follows:

```
SELECT DISTINCT CUSTOMERS.CUSTOMER_ID, CUSTOMERS.COMPANY, CUSTOMERS.FIRST_NAME,  
CUSTOMERS.LAST_NAME, CUSTOMERS.ADDRESS1, CUSTOMERS.ADDRESS2, CUSTOMERS.CITY,
```

```

CUSTOMERS.STATE, CUSTOMERS.POSTAL_CODE, CUSTOMERS.EMAIL, ORDERS.ORDER_ID,
ORDERS.DATE_ENTERED, ORDERS.DATE_PROMISED, ORDERS.DATE_SHIPPED, ORDERS.EMPLOYEE_ID,
ORDERS.CUSTOMER_ID, ORDERS.SALES_TAX_RATE, ORDERS.STORE_ID
FROM
CUSTOMERS, ORDERS
WHERE
CUSTOMERS.CUSTOMER_ID=ORDERS.CUSTOMER_ID

```

However, if you modify the default query after choosing Select Distinct, the Integration Service uses only the query defined in the SQL Query property. In other words, the SQL Query overrides the Select Distinct setting.

To use Select Distinct:

1. Open the Source Qualifier transformation in the mapping, and click on the Properties tab.
2. Check Select Distinct, and Click OK.

Overriding Select Distinct in the Session

You can override the transformation level option to Select Distinct when you configure the session in the Workflow Manager.

To override the Select Distinct option:

1. In the Workflow Manager, open the Session task, and click the Mapping tab.
2. Click the Transformations view, and click the Source Qualifier transformation under the Sources node.
3. In the Properties settings, enable Select Distinct, and click OK.

Adding Pre- and Post-Session SQL Commands

You can add pre- and post-session SQL commands on the Properties tab in the Source Qualifier transformation. You might want to use pre-session SQL to write a timestamp row to the source table when a session begins.

The Integration Service runs pre-session SQL commands against the source database before it reads the source. It runs post-session SQL commands against the source database after it writes to the target.

You can override the SQL commands in the Transformations view on the Mapping tab in the session properties.

You can also configure the Integration Service to stop or continue when it encounters errors running pre- or post-session SQL commands.

Use the following guidelines when you enter pre- and post-session SQL commands in the Source Qualifier transformation:

- ϕ Use any command that is valid for the database type. However, the Integration Service does not allow nested comments, even though the database might.
- ϕ You can use parameters and variables in source pre- and post-session SQL commands, or you can use a parameter or variable as the command. Use any parameter or variable type that you can define in the parameter file.
- ϕ Use a semicolon (;) to separate multiple statements. The Integration Service issues a commit after each statement.
- ϕ The Integration Service ignores semicolons within /*...*/.
- ϕ If you need to use a semicolon outside of comments, you can escape it with a backslash (\). When you escape the semicolon, the Integration Service ignores the backslash, and it does not use the semicolon as a statement separator.
- ϕ The Designer does not validate the SQL.

Note: You can also enter pre- and post-session SQL commands on the Properties tab of the target instance in a mapping.

Creating a Source Qualifier Transformation

You can configure the Designer to create a Source Qualifier transformation by default when you drag a source into a mapping, or you can create a Source Qualifier transformation manually.

Creating a Source Qualifier Transformation By Default

You can configure the Designer to create a Source Qualifier transformation when you drag a source into a mapping.

To create a Source Qualifier transformation by default:

1. In the Designer, click Tools > Options.
2. Select the Tables tab.
3. In the Tools options, select Mapping Designer.
4. Select Create Source Qualifier When Opening Sources.

Creating a Source Qualifier Transformation Manually

You can manually create a Source Qualifier transformation in the Mapping Designer.

To create a Source Qualifier transformation manually:

1. In the Mapping Designer, click Transformation > Create.
2. Enter a name for the transformation, and click Create.
3. Select a source, and click OK.
4. Click Done.

Configuring Source Qualifier Transformation Options

After you create the Source Qualifier transformation, you can configure several options.

To configure a Source Qualifier transformation:

1. In the Designer, open a mapping.
2. Double-click the title bar of the Source Qualifier transformation.
3. In the Edit Transformations dialog box, click Rename, enter a descriptive name for the transformation, and click OK.

The naming convention for Source Qualifier transformations is *SQ_TransformationName*, such as *SQ_AllSources*.

4. Click the Properties tab.
5. Enter the Source Qualifier transformation properties.
6. Click the Sources tab and indicate any associated source definitions you want to define for this transformation.

Identify associated sources only when you need to join data from multiple databases or flat file systems.

7. Click OK.

Troubleshooting

I cannot perform a drag and drop operation, such as connecting ports.

Review the error message on the status bar for details.

I cannot connect a source definition to a target definition.

You cannot directly connect sources to targets. Instead, you need to connect them through a Source Qualifier transformation for relational and flat file sources, or through a Normalizer transformation for COBOL sources.

I cannot connect multiple sources to one target.

The Designer does not allow you to connect multiple Source Qualifier transformations to a single target. There are two workarounds:

- φ **Reuse targets.** Since target definitions are reusable, you can add the same target to the mapping multiple times. Then connect each Source Qualifier transformation to each target.
- φ **Join the sources in a Source Qualifier transformation.** Then remove the WHERE clause from the SQL query.

The source has QNAN (not a number) values in some columns, but the target shows 1.#QNAN.

Operating systems have different string representations of NaN. The Integration Service converts QNAN values to 1.#QNAN on Win64EMT platforms. 1.#QNAN is a valid representation of QNAN.

I entered a custom query, but it is not working when I run the workflow containing the session.

Be sure to test this setting for the Source Qualifier transformation before you run the workflow. Return to the Source Qualifier transformation and reopen the dialog box in which you entered the custom query. You can connect to a database and click the Validate button to test the SQL. The Designer displays any errors. Review the session log file if you need further information.

The most common reason a session fails is because the database login in both the session and Source Qualifier transformation is not the table owner. You need to specify the table owner in the session and when you generate the SQL Query in the Source Qualifier transformation.

You can test the SQL Query by cutting and pasting it into the database client tool (such as Oracle Net) to see if it returns an error.

I used a mapping variable in a source filter and now the session fails.

Try testing the query by generating and validating the SQL in the Source Qualifier transformation. If the variable or parameter is a string, you probably need to enclose it in single quotes. If it is a datetime variable or parameter, you might need to change its format for the source system.

Stored Procedure Transformation

This chapter includes the following topics:

- φ Overview
- φ Using a Stored Procedure in a Mapping
- φ Writing a Stored Procedure
- φ Creating a Stored Procedure Transformation
- φ Configuring a Connected Transformation
- φ Configuring an Unconnected Transformation
- φ Error Handling
- φ Supported Databases
- φ Expression Rules
- φ Tips
- φ Troubleshooting

Overview

Transformation type:

Passive
Connected/Unconnected

A Stored Procedure transformation is an important tool for populating and maintaining databases. Database administrators create stored procedures to automate tasks that are too complicated for standard SQL statements.

A stored procedure is a precompiled collection of Transact-SQL, PL-SQL or other database procedural statements and optional flow control statements, similar to an executable script. Stored procedures are stored and run within the database. You can run a stored procedure with the EXECUTE SQL statement in a database client tool, just as you can run SQL statements. Unlike standard SQL, however, stored procedures allow user-defined variables, conditional statements, and other powerful programming features.

Not all databases support stored procedures, and stored procedure syntax varies depending on the database. You might use stored procedures to complete the following tasks:

- φ Check the status of a target database before loading data into it.
- φ Determine if enough space exists in a database.
- φ Perform a specialized calculation.

- ϕ Drop and recreate indexes.

Database developers and programmers use stored procedures for various tasks within databases, since stored procedures allow greater flexibility than SQL statements. Stored procedures also provide error handling and logging necessary for critical tasks. Developers create stored procedures in the database using the client tools provided with the database.

The stored procedure must exist in the database before creating a Stored Procedure transformation, and the stored procedure can exist in a source, target, or any database with a valid connection to the Integration Service.

You might use a stored procedure to perform a query or calculation that you would otherwise make part of a mapping. For example, if you already have a well-tested stored procedure for calculating sales tax, you can perform that calculation through the stored procedure instead of recreating the same calculation in an Expression transformation.

Input and Output Data

One of the most useful features of stored procedures is the ability to send data to the stored procedure, and receive data from the stored procedure. There are three types of data that pass between the Integration Service and the stored procedure:

- ϕ Input/output parameters

- ϕ Return values

- ϕ Status codes

Some limitations exist on passing data, depending on the database implementation, which are discussed throughout this chapter. Additionally, not all stored procedures send and receive data. For example, if you write a stored procedure to rebuild a database index at the end of a session, you cannot receive data, since the session has already finished.

Input/Output Parameters

For many stored procedures, you provide a value and receive a value in return. These values are known as input and output parameters. For example, a sales tax calculation stored procedure can take a single input parameter, such as the price of an item. After performing the calculation, the stored procedure returns two output parameters, the amount of tax, and the total cost of the item including the tax.

The Stored Procedure transformation sends and receives input and output parameters using ports, variables, or by entering a value in an expression, such as 10 or SALES.

Return Values

Most databases provide a return value after running a stored procedure. Depending on the database implementation, this value can either be user-definable, which means that it can act similar to a single output parameter, or it may only return an integer value.

The Stored Procedure transformation captures return values in a similar manner as input/output parameters, depending on the method that the input/output parameters are captured. In some instances, only a parameter or a return value can be captured.

If a stored procedure returns a result set rather than a single return value, the Stored Procedure transformation takes only the first value returned from the procedure.

Note: An Oracle stored function is similar to an Oracle stored procedure, except that the stored function supports output parameters or return values. In this chapter, any statements regarding stored procedures also apply to stored functions, unless otherwise noted.

Status Codes

Status codes provide error handling for the Integration Service during a workflow. The stored procedure issues a status code that notifies whether or not the stored procedure completed successfully. You cannot see this value.

The Integration Service uses it to determine whether to continue running the session or stop. You configure options in the Workflow Manager to continue or stop the session in the event of a stored procedure error.

Connected and Unconnected

Stored procedures run in either connected or unconnected mode. The mode you use depends on what the stored procedure does and how you plan to use it in a session. You can configure connected and unconnected Stored Procedure transformations in a mapping.

- ϕ **Connected.** The flow of data through a mapping in connected mode also passes through the Stored Procedure transformation. All data entering the transformation through the input ports affects the stored procedure. You should use a connected Stored Procedure transformation when you need data from an input port sent as an input parameter to the stored procedure, or the results of a stored procedure sent as an output parameter to another transformation.
- ϕ **Unconnected.** The unconnected Stored Procedure transformation is not connected directly to the flow of the mapping. It either runs before or after the session, or is called by an expression in another transformation in the mapping.

The following table compares connected and unconnected transformations:

If you want to	Use this mode
Run a stored procedure before or after a session.	Unconnected
Run a stored procedure once during a mapping, such as pre- or post-session.	Unconnected
Run a stored procedure every time a row passes through the Stored Procedure transformation.	Connected or Unconnected
Run a stored procedure based on data that passes through the mapping, such as when a specific port does not contain a null value.	Unconnected
Pass parameters to the stored procedure and receive a single output parameter.	Connected or Unconnected
Pass parameters to the stored procedure and receive multiple output parameters. Note: To get multiple output parameters from an unconnected Stored Procedure transformation, you must create variables for each output parameter.	Connected or Unconnected
Run nested stored procedures.	Unconnected
Call multiple times within a mapping.	Unconnected

RELATED TOPICS:

- ϕ [“Configuring a Connected Transformation”](#)
- ϕ [“Configuring an Unconnected Transformation”](#)

Specifying when the Stored Procedure Runs

In addition to specifying the mode of the Stored Procedure transformation, you also specify when it runs. In the case of the unconnected stored procedure above, the Expression transformation references the stored procedure, which means the stored procedure runs every time a row passes through the Expression transformation. However, if no transformation references the Stored Procedure transformation, you have the option to run the stored procedure once before or after the session.

The following list describes the options for running a Stored Procedure transformation:

- ϕ **Normal.** The stored procedure runs where the transformation exists in the mapping on a row-by-row basis. This is useful for calling the stored procedure for each row of data that passes through the mapping, such as running a calculation against an input port. Connected stored procedures run only in normal mode.
- ϕ **Pre-load of the Source.** Before the session retrieves data from the source, the stored procedure runs. This is useful for verifying the existence of tables or performing joins of data in a temporary table.

- ϕ **Post-load of the Source.** After the session retrieves data from the source, the stored procedure runs. This is useful for removing temporary tables.
- ϕ **Pre-load of the Target.** Before the session sends data to the target, the stored procedure runs. This is useful for verifying target tables or disk space on the target system.
- ϕ **Post-load of the Target.** After the session sends data to the target, the stored procedure runs. This is useful for re-creating indexes on the database.

You can run more than one Stored Procedure transformation in different modes in the same mapping. For example, a pre-load source stored procedure can check table integrity, a normal stored procedure can populate the table, and a post-load stored procedure can rebuild indexes in the database. However, you cannot run the same instance of a Stored Procedure transformation in both connected and unconnected mode in a mapping. You must create different instances of the transformation.

If the mapping calls more than one source or target pre- or post-load stored procedure in a mapping, the Integration Service executes the stored procedures in the execution order that you specify in the mapping.

The Integration Service executes each stored procedure using the database connection you specify in the transformation properties. The Integration Service opens the database connection when it encounters the first stored procedure. The database connection remains open until the Integration Service finishes processing all stored procedures for that connection. The Integration Service closes the database connections and opens a new one when it encounters a stored procedure using a different database connection.

To run multiple stored procedures that use the same database connection, set these stored procedures to run consecutively. If you do not set them to run consecutively, you might have unexpected results in the target. For example, you have two stored procedures: Stored Procedure A and Stored Procedure B. Stored Procedure A begins a transaction, and Stored Procedure B commits the transaction. If you run Stored Procedure C before Stored Procedure B, using another database connection, Stored Procedure B cannot commit the transaction because the Integration Service closes the database connection when it runs Stored Procedure C.

Use the following guidelines to run multiple stored procedures within a database connection:

- ϕ The stored procedures use the same database connect string defined in the stored procedure properties.
- ϕ You set the stored procedures to run in consecutive order.
- ϕ The stored procedures have the same stored procedure type:
 - Source pre-load
 - Source post-load
 - Target pre-load
 - Target post-load

Using a Stored Procedure in a Mapping

You must perform several steps to use a Stored Procedure transformation in a mapping. Since the stored procedure exists in the database, you must configure not only the mapping and session, but the stored procedure in the database as well.

To use a Stored Procedure transformation, complete the following steps:

1. Create the stored procedure in the database.

Before using the Designer to create the transformation, you must create the stored procedure in the database. You should also test the stored procedure through the provided database client tools.
2. Import or create the Stored Procedure transformation.

Use the Designer to import or create the Stored Procedure transformation, providing ports for any necessary input/output and return values.

3. Determine whether to use the transformation as connected or unconnected.
You must determine how the stored procedure relates to the mapping before configuring the transformation.
4. If connected, map the appropriate input and output ports.
You use connected Stored Procedure transformations just as you would most other transformations. Drag the appropriate input flow ports to the transformation, and create mappings from output ports to other transformations.
5. If unconnected, either configure the stored procedure to run pre- or post-session, or configure it to run from an expression in another transformation.
Since stored procedures can run before or after the session, you may need to specify when the unconnected transformation should run. On the other hand, if the stored procedure is called from another transformation, you write the expression in another transformation that calls the stored procedure. The expression can contain variables, and may or may not include a return value.
6. Configure the session.
The session properties in the Workflow Manager includes options for error handling when running stored procedures and several SQL override options.

Writing a Stored Procedure

Write SQL statements to create a stored procedure in the database, and you can add other Transact-SQL statements and database-specific functions. These can include user-defined datatypes and execution order statements.

Sample Stored Procedure

In the following example, the source database has a stored procedure that takes an input parameter of an employee ID number, and returns an output parameter of the employee name. In addition, a return value of 0 is returned as a notification that the stored procedure completed successfully. The database table that contains employee IDs and names appears as follows:

Employee ID	Employee Name
101	Bil Takash
102	Louis Li
103	Sarah Ferguson

The stored procedure receives the employee ID 101 as an input parameter, and returns the name Bill Takash. Depending on how the mapping calls this stored procedure, any or all of the IDs may be passed to the stored procedure.

Since the syntax varies between databases, the SQL statements to create this stored procedure may vary. The client tools used to pass the SQL statements to the database also vary. Most databases provide a set of client tools, including a standard SQL editor. Some databases, such as Microsoft SQL Server, provide tools that create some of the initial SQL statements.

Note: The Integration Service fails sessions that contain stored procedure arguments with large objects.

Informix

In Informix, the syntax for declaring an output parameter differs from other databases. With most databases, you declare variables using IN or OUT to specify if the variable acts as an input or output parameter. Informix

uses the keyword RETURNING, making it difficult to distinguish input/output parameters from return values. For example, you use the RETURN command to return one or more output parameters:

```
CREATE PROCEDURE GET_NAME_USING_ID (nID integer)
RETURNING varchar(20);
define nID integer;
define outVAR as varchar(20);
SELECT FIRST_NAME INTO outVAR FROM CONTACT WHERE ID = nID
return outVAR;
END PROCEDURE;
```

Notice that in this case, the RETURN statement passes the value of outVAR. Unlike other databases, however, outVAR is not a return value, but an output parameter. Multiple output parameters would be returned in the following manner:

```
return outVAR1, outVAR2, outVAR3
```

Informix does pass a return value. The return value is not user-defined, but generated as an error-checking value. In the transformation, the R value must be checked.

Oracle

In Oracle, any stored procedure that returns a value is called a stored function. Rather than using the CREATE PROCEDURE statement to make a new stored procedure based on the example, you use the CREATE FUNCTION statement. In this sample, the variables are declared as IN and OUT, but Oracle also supports an INOUT parameter type, which lets you pass in a parameter, modify it, and return the modified value:

```
CREATE OR REPLACE FUNCTION GET_NAME_USING_ID (
nID IN NUMBER,
outVAR OUT VARCHAR2)
RETURN VARCHAR2 IS
RETURN_VAR varchar2(100);
BEGIN
SELECT FIRST_NAME INTO outVAR FROM CONTACT WHERE ID = nID;
RETURN_VAR := 'Success';
RETURN (RETURN_VAR);
END;
/
```

Notice that the return value is a string value (Success) with the datatype VARCHAR2. Oracle is the only database to allow return values with string datatypes.

Sybase ASE and Microsoft SQL Server

Sybase and Microsoft implement stored procedures identically, as the following syntax shows:

```
CREATE PROCEDURE GET_NAME_USING_ID @nID int = 1, @outVar varchar(20) OUTPUT
AS
SELECT @outVar = FIRST_NAME FROM CONTACT WHERE ID = @nID
return 0
```

Notice that the return value does not need to be a variable. In this case, if the SELECT statement is successful, a 0 is returned as the return value.

IBM DB2

The following text is an example of an SQL stored procedure on IBM DB2:

```
CREATE PROCEDURE get_name_using_id ( IN id_in int,
                                     OUT emp_out char(18),
                                     OUT sqlcode_out int)

LANGUAGE SQL

P1: BEGIN
-- Declare variables
DECLARE SQLCODE INT DEFAULT 0;
DECLARE emp_TMP char(18) DEFAULT ' ';
-- Declare handler
DECLARE EXIT HANDLER FOR SQLEXCEPTION
```

```

        SET SQLCODE_OUT = SQLCODE;
select employee into emp_TMP
    from doc_employee
    where id = id_in;
SET emp_out      = EMP_TMP;
SET sqlcode_out = SQLCODE;
END P1

```

Teradata

The following text is an example of an SQL stored procedure on Teradata. It takes an employee ID number as an input parameter and returns the employee name as an output parameter:

```

CREATE PROCEDURE GET_NAME_USING_ID (IN nID integer, OUT outVAR varchar(40))
BEGIN
    SELECT FIRST_NAME INTO :outVAR FROM CONTACT where ID = :nID;
END;

```

Creating a Stored Procedure Transformation

After you configure and test a stored procedure in the database, you must create the Stored Procedure transformation in the Mapping Designer. There are two ways to configure the Stored Procedure transformation:

- ϕ Use the Import Stored Procedure dialog box to configure the ports used by the stored procedure.
- ϕ Configure the transformation manually, creating the appropriate ports for any input or output parameters.

Stored Procedure transformations are created as Normal type by default, which means that they run during the mapping, not before or after the session.

New Stored Procedure transformations are not created as reusable transformations. To create a reusable transformation, click Make Reusable in the Transformation properties after creating the transformation.

Note: Configure the properties of reusable transformations in the Transformation Developer, not the Mapping Designer, to make changes globally for the transformation.

Importing Stored Procedures

When you import a stored procedure, the Designer creates ports based on the stored procedure input and output parameters. You should import the stored procedure whenever possible.

There are three ways to import a stored procedure in the Mapping Designer:

- ϕ Select the stored procedure icon and add a Stored Procedure transformation.
- ϕ Click Transformation > Import Stored Procedure.
- ϕ Click Transformation > Create, and then select Stored Procedure.

When you import a stored procedure containing a period (.) in the stored procedure name, the Designer substitutes an underscore (_) for the period in the Stored Procedure transformation name.

To import a stored procedure:

1. In the Mapping Designer, click Transformation > Import Stored Procedure.
2. Select the database that contains the stored procedure from the list of ODBC sources. Enter the user name, owner name, and password to connect to the database and click Connect.

The folder in the dialog box displays FUNCTIONS. The stored procedures in this folder contain input parameters, output parameters, or a return value. If stored procedures exist in the database that do not contain parameters or return values, they appear in a folder called PROCEDURES. This applies primarily to Oracle stored procedures. For a normal connected Stored Procedure to appear in the functions list, it requires at least one input and one output port.

Tip: You can select Skip to add a Stored Procedure transformation without importing the stored procedure. In this case, you need to manually add the ports and connect information within the transformation.

3. Select the procedure to import and click OK.

The Stored Procedure transformation appears in the mapping. The Stored Procedure transformation name is the same as the stored procedure you selected. If the stored procedure contains input parameters, output parameters, or a return value, you see the appropriate ports that match each parameter or return value in the Stored Procedure transformation.

In this Stored Procedure transformation, you can see that the stored procedure contains the following value and parameters:

- ϕ An integer return value, called RETURN_VALUE, with an output port.
- ϕ A string input parameter, called nNAME, with an input port.
- ϕ An integer output parameter, called outVar, with an input and output port.

Note: If you change the transformation name, you need to configure the name of the stored procedure in the transformation properties. If you have multiple instances of the same stored procedure in a mapping, you must also configure the name of the stored procedure.

4. Open the transformation, and click the Properties tab.

Select the database where the stored procedure exists from the Connection Information row. If you changed the name of the Stored Procedure transformation to something other than the name of the stored procedure, enter the Stored Procedure Name.

5. Click OK.

Manually Creating Stored Procedure Transformations

To create a Stored Procedure transformation manually, you need to know the input parameters, output parameters, and return values of the stored procedure, if there are any. You must also know the datatypes of those parameters, and the name of the stored procedure. All these are configured through Import Stored Procedure.

To create a Stored Procedure transformation:

1. In the Mapping Designer, click Transformation > Create, and then select Stored Procedure.

The naming convention for a Stored Procedure transformation is the name of the stored procedure, which happens automatically. If you change the transformation name, then you need to configure the name of the stored procedure in the Transformation Properties. If you have multiple instances of the same stored procedure in a mapping, you must perform this step.

2. Click Skip.

The Stored Procedure transformation appears in the Mapping Designer.

3. Open the transformation, and click the Ports tab.

You must create ports based on the input parameters, output parameters, and return values in the stored procedure. Create a port in the Stored Procedure transformation for each of the following stored procedure parameters:

- ϕ An integer input parameter
- ϕ A string output parameter
- ϕ A return value

For the integer input parameter, you would create an integer input port. The parameter and the port must be the same datatype and precision. Repeat this for the output parameter and the return value.

The R column should be selected and the output port for the return value. For stored procedures with multiple parameters, you must list the ports in the same order that they appear in the stored procedure.

4. Click the Properties tab.

Enter the name of the stored procedure in the Stored Procedure Name row, and select the database where the stored procedure exists from the Connection Information row.

5. Click OK.

Although the repository validates and saves the mapping, the Designer does not validate the manually entered Stored Procedure transformation. No checks are completed to verify that the proper parameters or return value exist in the stored procedure. If the Stored Procedure transformation is not configured properly, the session fails.

Setting Options for the Stored Procedure

The following table describes the properties for a Stored Procedure transformation:

Setting	Description
Stored Procedure Name	Name of the stored procedure in the database. The Integration Service uses this text to call the stored procedure if the name of the transformation is different than the actual stored procedure name in the database. Leave this field blank if the transformation name matches the stored procedure name. When using the Import Stored Procedure feature, this name matches the stored procedure.
Connection Information	Specifies the database containing the stored procedure. You can define the database in the mapping, session, or parameter file: <ul style="list-style-type: none">- Mapping. Select the relational connection object.- Session. Use the \$Source or \$Target connection variable. If you use one of these variables, the stored procedure must reside in the source or target database you specify when you run the session. Specify the database connection for each variable in the session properties.- Parameter file. Use the session parameter \$DBConnectionName, and define it in the parameter file. By default, the Designer specifies \$Target for Normal stored procedure types. For source pre- and post-load, the Designer specifies \$Source. For target pre- and post-load, the Designer specifies \$Target. You can override these values in the session properties.
Call Text	Text used to call the stored procedure. Only used when the Stored Procedure Type is not Normal. You must include all input parameters passed to the stored procedure within the call text. You can also use a PowerCenter parameter or variable in the call text. Use any parameter or variable type that you can define in the parameter file.
Stored Procedure Type	Determines when the Integration Service calls the stored procedure. The options include Normal (during the mapping) or pre- or post-load on the source or target database. Default is Normal.
Tracing Level	Amount of transaction detail reported in the session log file. Use the following tracing levels: <ul style="list-style-type: none">- Terse- Normal- Verbose Initialization- Verbose Data Default is Normal.
Execution Order	Order in which the Integration Service calls the stored procedure used in the transformation, relative to any other stored procedures in the same mapping. Only used when the Stored Procedure Type is set to anything except Normal and more than one stored procedure exists.
Subsecond Precision	Specifies the subsecond precision for datetime ports. You can change the precision for databases that have an editable scale for datetime data. You can change subsecond precision for Oracle Timestamp, Informix Datetime, and Teradata Timestamp datatypes. Enter a positive integer value from 0 to 9. Default is 6 (microseconds).

Setting	Description
Output is Repeatable	<p>Indicates whether the transformation generates rows in the same order between session runs. The Integration Service can resume a session from the last checkpoint when the output is repeatable and deterministic. Use the following values:</p> <ul style="list-style-type: none"> - Always. The order of the output data is consistent between session runs even if the order of the input data is inconsistent between session runs. - Based on Input Order. The transformation produces repeatable data between session runs when the order of the input data from all input groups is consistent between session runs. If the input data from any input group is not ordered, then the output is not ordered. - Never. The order of the output data is inconsistent between session runs. You cannot configure recovery to resume from the last checkpoint if a transformation does not produce repeatable data. <p>Default is Based on Input Order.</p>
Output is Deterministic	<p>Indicates whether the transformation generates consistent output data between session runs. You must enable this property to perform recovery on sessions that use this transformation.</p> <p>Default is disabled.</p>

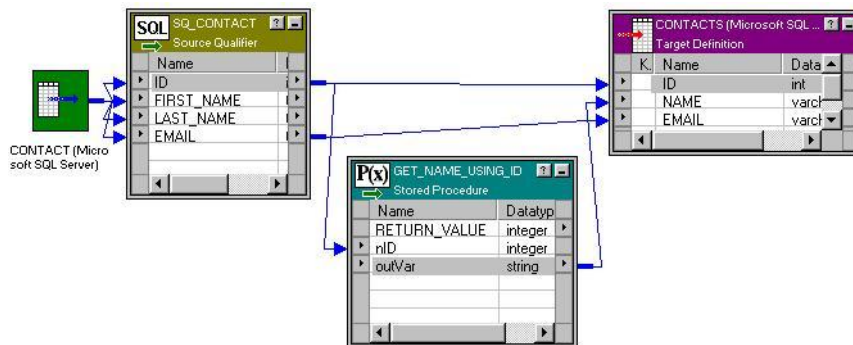
Warning: If you configure a transformation as repeatable and deterministic, it is your responsibility to ensure that the data is repeatable and deterministic. If you try to recover a session with transformations that do not produce the same data between the session and the recovery, the recovery process can result in corrupted data.

Changing the Stored Procedure

If the number of parameters or the return value in a stored procedure changes, you can either re-import it or edit the Stored Procedure transformation manually. The Designer does not verify the Stored Procedure transformation each time you open the mapping. After you import or create the transformation, the Designer does not validate the stored procedure. The session fails if the stored procedure does not match the transformation.

Configuring a Connected Transformation

The following figure shows a mapping that sends the ID from the Source Qualifier to an input parameter in the Stored Procedure transformation. The Stored Procedure transformation passes an output parameter to the target. Every row of data in the Source Qualifier transformation passes data through the Stored Procedure transformation:



Although not required, almost all connected Stored Procedure transformations contain input and output parameters. Required input parameters are specified as the input ports of the Stored Procedure transformation. Output parameters appear as output ports in the transformation. A return value is also an output port, and has

the R value selected in the transformation Ports configuration. For a normal connected Stored Procedure to appear in the functions list, it requires at least one input and one output port.

Output parameters and return values from the stored procedure are used as any other output port in a transformation. You can link these ports to another transformation or target.

To configure a connected Stored Procedure transformation:

1. Create the Stored Procedure transformation in the mapping.
2. Drag the output ports of the Stored Procedure to other transformations or targets.
3. Open the Stored Procedure transformation, and select the Properties tab.
4. Select the appropriate database in the Connection Information if you did not select it when creating the transformation.
5. Select the Tracing level for the transformation.

If you are testing the mapping, select the Verbose Initialization option to provide the most information in the event that the transformation fails.

6. Click OK.

Configuring an Unconnected Transformation

An unconnected Stored Procedure transformation is not directly connected to the flow of data through the mapping. Instead, the stored procedure runs either:

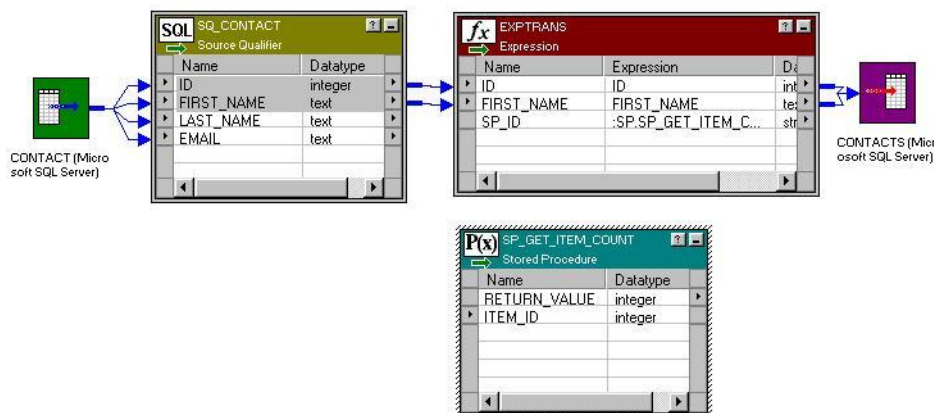
- φ **From an expression.** Called from an expression written in the Expression Editor within another transformation in the mapping.
- φ **Pre- or post-session.** Runs before or after a session.

The sections below explain how you can run an unconnected Stored Procedure transformation.

Calling a Stored Procedure From an Expression

In an unconnected mapping, the Stored Procedure transformation does not connect to the pipeline.

The following figure shows a mapping with an Expression transformation that references the Stored Procedure transformation:



However, just like a connected mapping, you can apply the stored procedure to the flow of data through the mapping. In fact, you have greater flexibility since you use an expression to call the stored procedure, which means you can select the data that you pass to the stored procedure as an input parameter.

When using an unconnected Stored Procedure transformation in an expression, you need a method of returning the value of output parameters to a port. Use one of the following methods to capture the output values:

- φ Assign the output value to a local variable.
- φ Assign the output value to the system variable PROC_RESULT.

By using PROC_RESULT, you assign the value of the return parameter directly to an output port, which can apply directly to a target. You can also combine the two options by assigning one output parameter as PROC_RESULT, and the other parameter as a variable.

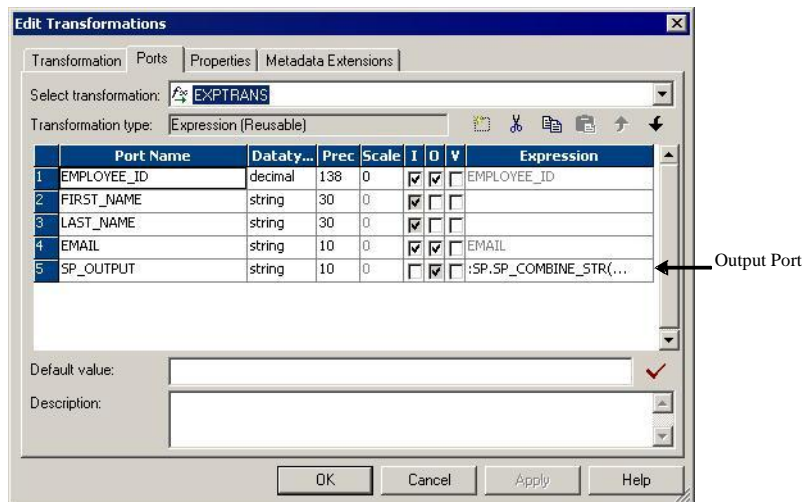
Use PROC_RESULT only within an expression. If you do not use PROC_RESULT or a variable, the port containing the expression captures a NULL. You cannot use PROC_RESULT in a connected Lookup transformation or within the Call Text for a Stored Procedure transformation.

If you require nested stored procedures, where the output parameter of one stored procedure passes to another stored procedure, use PROC_RESULT to pass the value.

The Integration Service calls the unconnected Stored Procedure transformation from the Expression transformation. Notice that the Stored Procedure transformation has two input ports and one output port. All three ports are string datatypes.

To call a stored procedure from within an expression:

1. Create the Stored Procedure transformation in the mapping.
2. In any transformation that supports output and variable ports, create a new output port in the transformation that calls the stored procedure. Name the output port.



The output port that calls the stored procedure must support expressions. Depending on how the expression is configured, the output port contains the value of the output parameter or the return value.

3. Open the Expression Editor for the port.

The value for the new port is set up in the Expression Editor as a call to the stored procedure using the :SP keyword in the Transformation Language. The easiest way to set this up properly is to select the Stored Procedures node in the Expression Editor, and click the name of Stored Procedure transformation listed. For a normal connected Stored Procedure to appear in the functions list, it requires at least one input and one output port.

The stored procedure appears in the Expression Editor with a pair of empty parentheses. The necessary input and/or output parameters are displayed in the lower left corner of the Expression Editor.

4. Configure the expression to send input parameters and capture output parameters or return value.

You must know whether the parameters shown in the Expression Editor are input or output parameters. You insert variables or port names between the parentheses in the order that they appear in the stored procedure. The datatypes of the ports and variables must match those of the parameters passed to the stored procedure.

For example, when you click the stored procedure, something similar to the following appears:

```
:SP.GET_NAME_FROM_ID()
```

This particular stored procedure requires an integer value as an input parameter and returns a string value as an output parameter. How the output parameter or return value is captured depends on the number of output parameters and whether the return value needs to be captured.

If the stored procedure returns a single output parameter or a return value (but not both), you should use the reserved variable `PROC_RESULT` as the output variable. In the previous example, the expression would appear as:

```
:SP.GET_NAME_FROM_ID(inID, PROC_RESULT)
```

`inID` can be either an input port for the transformation or a variable in the transformation. The value of `PROC_RESULT` is applied to the output port for the expression.

If the stored procedure returns multiple output parameters, you must create variables for each output parameter. For example, if you create a port called `varOUTPUT2` for the stored procedure expression, and a variable called `varOUTPUT1`, the expression appears as:

```
:SP.GET_NAME_FROM_ID(inID, varOUTPUT1, PROC_RESULT)
```

The value of the second output port is applied to the output port for the expression, and the value of the first output port is applied to `varOUTPUT1`. The output parameters are returned in the order they are declared in the stored procedure.

With all these expressions, the datatypes for the ports and variables must match the datatypes for the input/output variables and return value.

5. Click Validate to verify the expression, and then click OK to close the Expression Editor.

Validating the expression ensures that the datatypes for parameters in the stored procedure match those entered in the expression.

6. Click OK.

When you save the mapping, the Designer does not validate the stored procedure expression. If the stored procedure expression is not configured properly, the session fails. When testing a mapping using a stored procedure, set the Override Tracing session option to a verbose mode and configure the On Stored Procedure session option to stop running if the stored procedure fails. Configure these session options in the Error Handling settings of the Config Object tab in the session properties.

The stored procedure in the expression entered for a port does not have to affect all values that pass through the port. Using the IIF statement, for example, you can pass only certain values, such as ID numbers that begin with 5, to the stored procedure and skip all other values. You can also set up nested stored procedures so the return value of one stored procedure becomes an input parameter for a second stored procedure.

Calling a Pre- or Post-Session Stored Procedure

You may want to run a stored procedure once per session. For example, if you need to verify that tables exist in a target database before running a mapping, a pre-load target stored procedure can check the tables, and then either continue running the workflow or stop it. You can run a stored procedure on the source, target, or any other connected database.

To create a pre- or post-load stored procedure:

1. Create the Stored Procedure transformation in the mapping.
2. Double-click the Stored Procedure transformation, and select the Properties tab.

3. Enter the name of the stored procedure.

If you imported the stored procedure, the stored procedure name appears by default. If you manually set up the stored procedure, enter the name of the stored procedure.

4. Select the database that contains the stored procedure in Connection Information.

5. Enter the call text of the stored procedure.

This is the name of the stored procedure, followed by all applicable input parameters in parentheses. If there are no input parameters, you must include an empty pair of parentheses, or the call to the stored procedure fails. You do not need to include the SQL statement EXEC, nor do you need to use the :SP keyword. For example, to call a stored procedure called check_disk_space, enter the following text:

```
check_disk_space()
```

To pass a string input parameter, enter it without quotes. If the string has spaces in it, enclose the parameter in double quotes. For example, if the stored procedure check_disk_space required a machine name as an input parameter, enter the following text:

```
check_disk_space(oracle_db)
```

You must enter values for the input parameters, since pre- and post-session procedures cannot pass variables.

When passing a datetime value through a pre- or post-session stored procedure, the value must be in the Informatica default date format and enclosed in double quotes as follows:

```
SP("12/31/2000 11:45:59")
```

You can use PowerCenter parameters and variables in the call text. Use any parameter or variable type that you can define in the parameter file. You can enter a parameter or variable within the call text, or you can use a parameter or variable as the call text. For example, you can use a session parameter, \$ParamMyCallText, as the call text, and set \$ParamMyCallText to the call text in a parameter file.

6. Select the stored procedure type.

The options for stored procedure type include:

- φ **Source Pre-load.** Before the session retrieves data from the source, the stored procedure runs. This is useful for verifying the existence of tables or performing joins of data in a temporary table.
- φ **Source Post-load.** After the session retrieves data from the source, the stored procedure runs. This is useful for removing temporary tables.
- φ **Target Pre-load.** Before the session sends data to the target, the stored procedure runs. This is useful for verifying target tables or disk space on the target system.
- φ **Target Post-load.** After the session sends data to the target, the stored procedure runs. This is useful for re-creating indexes on the database.

7. Select Execution Order, and click the Up or Down arrow to change the order, if necessary.

If you have added several stored procedures that execute at the same point in a session, such as two procedures that both run at Source Post-load, you can set a stored procedure execution plan to determine the order in which the Integration Service calls these stored procedures. You need to repeat this step for each stored procedure you wish to change.

8. Click OK.

Although the repository validates and saves the mapping, the Designer does not validate whether the stored procedure expression runs without an error. If the stored procedure expression is not configured properly, the session fails. When testing a mapping using a stored procedure, set the Override Tracing session option to a verbose mode and configure the On Stored Procedure session option to stop running if the stored procedure fails. Configure these session options on the Error Handling settings of the Config Object tab in the session properties.

You lose output parameters or return values called during pre- or post-session stored procedures, since there is no place to capture the values. If you need to capture values, you might want to configure the stored procedure to save the value in a table in the database.

Error Handling

Sometimes a stored procedure returns a database error, such as “divide by zero” or “no more rows.” The final result of a database error during a stored procedure depends on when the stored procedure takes place and how the session is configured.

You can configure the session to either stop or continue running the session upon encountering a pre- or post-session stored procedure error. By default, the Integration Service stops a session when a pre- or post-session stored procedure database error occurs.

Pre-Session Errors

Pre-read and pre-load stored procedures are considered pre-session stored procedures. Both run before the Integration Service begins reading source data. If a database error occurs during a pre-session stored procedure, the Integration Service performs a different action depending on the session configuration.

- ϕ If you configure the session to stop upon stored procedure error, the Integration Service fails the session.
- ϕ If you configure the session to continue upon stored procedure error, the Integration Service continues with the session.

Post-Session Errors

Post-read and post-load stored procedures are considered post-session stored procedures. Both run after the Integration Service commits all data to the database. If a database error occurs during a post-session stored procedure, the Integration Service performs a different action depending on the session configuration:

- ϕ If you configure the session to stop upon stored procedure error, the Integration Service fails the session.
However, the Integration Service has already committed all data to session targets.
- ϕ If you configure the session to continue upon stored procedure error, the Integration Service continues with the session.

Session Errors

Connected or unconnected stored procedure errors occurring during the session are not affected by the session error handling option. If the database returns an error for a particular row, the Integration Service skips the row and continues to the next row. As with other row transformation errors, the skipped row appears in the session log.

Supported Databases

The supported options for Oracle, and other databases, such as Informix, Microsoft SQL Server, and Sybase are described below.

RELATED TOPICS:

- ϕ “Writing a Stored Procedure”

SQL Declaration

In the database, the statement that creates a stored procedure appears similar to the following Oracle stored procedure:

```
create or replace procedure sp_combine_str
(str1_inout IN OUT varchar2,
str2_inout IN OUT varchar2,
```

```

str_out OUT varchar2)
is
begin
    str1_inout := UPPER(str1_inout);
    str2_inout := upper(str2_inout);
    str_out := str1_inout || ' ' || str2_inout;
end;

```

In this case, the Oracle statement begins with CREATE OR REPLACE PROCEDURE. Since Oracle supports both stored procedures and stored functions, only Oracle uses the optional CREATE FUNCTION statement.

Parameter Types

There are three possible parameter types in stored procedures:

- ϕ **IN.** Defines the parameter something that must be passed to the stored procedure.
- ϕ **OUT.** Defines the parameter as a returned value from the stored procedure.
- ϕ **INOUT.** Defines the parameter as both input and output. Only Oracle supports this parameter type.

Input/Output Port in Mapping

Since Oracle supports the INOUT parameter type, a port in a Stored Procedure transformation can act as both an input and output port for the same stored procedure parameter. Other databases should not have both the input and output check boxes selected for a port.

Type of Return Value Supported

Different databases support different types of return value datatypes, and only Informix does not support user-defined return values.

Expression Rules

Unconnected Stored Procedure transformations can be called from an expression in another transformation. Use the following rules and guidelines when configuring the expression:

- ϕ A single output parameter is returned using the variable PROC_RESULT.
- ϕ When you use a stored procedure in an expression, use the :SP reference qualifier. To avoid typing errors, select the Stored Procedure node in the Expression Editor, and double-click the name of the stored procedure.
- ϕ However, the same instance of a Stored Procedure transformation cannot run in both connected and unconnected mode in a mapping. You must create different instances of the transformation.
- ϕ The input/output parameters in the expression must match the input/output ports in the Stored Procedure transformation. If the stored procedure has an input parameter, there must also be an input port in the Stored Procedure transformation.
- ϕ When you write an expression that includes a stored procedure, list the parameters in the same order that they appear in the stored procedure and the Stored Procedure transformation.
- ϕ The parameters in the expression must include all of the parameters in the Stored Procedure transformation. You cannot leave out an input parameter. If necessary, pass a dummy variable to the stored procedure.
- ϕ The arguments in the expression must be the same datatype and precision as those in the Stored Procedure transformation.
- ϕ Use PROC_RESULT to apply the output parameter of a stored procedure expression directly to a target. You cannot use a variable for the output parameter to pass the results directly to a target. Use a local variable to pass the results to an output port within the same transformation.

φ Nested stored procedures allow passing the return value of one stored procedure as the input parameter of another stored procedure. For example, if you have the following two stored procedures:

- `get_employee_id (employee_name)`
- `get_employee_salary (employee_id)`

And the return value for `get_employee_id` is an employee ID number, the syntax for a nested stored procedure is:

```
:sp.get_employee_salary (:sp.get_employee_id (employee_name))
```

You can have multiple levels of nested stored procedures.

φ Do not use single quotes around string parameters. If the input parameter does not contain spaces, do not use any quotes. If the input parameter contains spaces, use double quotes.

Tips

[Do not run unnecessary instances of stored procedures.](#)

Each time a stored procedure runs during a mapping, the session must wait for the stored procedure to complete in the database. You have two possible options to avoid this:

- φ **Reduce the row count.** Use an active transformation prior to the Stored Procedure transformation to reduce the number of rows that must be passed the stored procedure. Or, create an expression that tests the values before passing them to the stored procedure to make sure that the value does not really need to be passed.
- φ **Create an expression.** Most of the logic used in stored procedures can be easily replicated using expressions in the Designer.

Troubleshooting

[I get the error “stored procedure not found” in the session log file.](#)

Make sure the stored procedure is being run in the correct database. By default, the Stored Procedure transformation uses the target database to run the stored procedure. Double-click the transformation in the mapping, select the Properties tab, and check which database is selected in Connection Information.

[My output parameter was not returned using a Microsoft SQL Server stored procedure.](#)

Check if the parameter to hold the return value is declared as OUTPUT in the stored procedure. With Microsoft SQL Server, OUTPUT implies input/output. In the mapping, you probably have checked both the I and O boxes for the port. Clear the input port.

[The session did not have errors before, but now it fails on the stored procedure.](#)

The most common reason for problems with a Stored Procedure transformation results from changes made to the stored procedure in the database. If the input/output parameters or return value changes in a stored procedure, the Stored Procedure transformation becomes invalid. You must either import the stored procedure again, or manually configure the stored procedure to add, remove, or modify the appropriate ports.

[The session has been invalidated since I last edited the mapping. Why?](#)

Any changes you make to the Stored Procedure transformation may invalidate the session. The most common reason is that you have changed the type of stored procedure, such as from a Normal to a Post-load Source type.

Union Transformation

This chapter includes the following topics:

- ϕ Overview
- ϕ Working with Groups and Ports
- ϕ Creating a Union Transformation
- ϕ Using a Union Transformation in a Mapping

Overview

Transformation type:

Active
Connected

The Union transformation is a multiple input group transformation that you use to merge data from multiple pipelines or pipeline branches into one pipeline branch. It merges data from multiple sources similar to the UNION ALL SQL statement to combine the results from two or more SQL statements. Similar to the UNION ALL statement, the Union transformation does not remove duplicate rows.

The Integration Service processes all input groups in parallel. It concurrently reads sources connected to the Union transformation and pushes blocks of data into the input groups of the transformation. The Union transformation processes the blocks of data based on the order it receives the blocks from the Integration Service.

You can connect heterogeneous sources to a Union transformation. The transformation merges sources with matching ports and outputs the data from one output group with the same ports as the input groups.

The Union transformation is developed using the Custom transformation.

Union Transformation Rules and Guidelines

Use the following rules and guidelines when you work with a Union transformation:

- ϕ You can create multiple input groups, but only one output group.
- ϕ All input groups and the output group must have matching ports. The precision, datatype, and scale must be identical across all groups.
- ϕ The Union transformation does not remove duplicate rows. To remove duplicate rows, you must add another transformation such as a Router or Filter transformation.

- ϕ You cannot use a Sequence Generator or Update Strategy transformation upstream from a Union transformation.
- ϕ The Union transformation does not generate transactions.

Union Transformation Components

When you configure a Union transformation, define the following components:

- ϕ **Transformation tab.** You can rename the transformation and add a description.
- ϕ **Properties tab.** You can specify the tracing level.
- ϕ **Groups tab.** You can create and delete input groups. The Designer displays groups you create on the Ports tab.
- ϕ **Group Ports tab.** You can create and delete ports for the input groups. The Designer displays ports you create on the Ports tab.

You cannot modify the Ports, Initialization Properties, Metadata Extensions, or Port Attribute Definitions tabs in a Union transformation.

Working with Groups and Ports

A Union transformation has multiple input groups and one output group. Create input groups on the Groups tab, and create ports on the Group Ports tab.

You can create one or more input groups on the Groups tab. The Designer creates one output group by default.

You cannot edit or delete the output group.

You can create ports by copying ports from a transformation, or you can create ports manually. When you create ports on the Group Ports tab, the Designer creates input ports in each input group and output ports in the output group. The Designer uses the port names you specify on the Group Ports tab for each input and output port, and it appends a number to make each port name in the transformation unique. It also uses the same metadata for each port, such as datatype, precision, and scale.

The Ports tab displays the groups and ports you create. You cannot edit group and port information on the Ports tab. Use the Groups and Group Ports tab to edit groups and ports.

Creating a Union Transformation

Use the following procedure to create a Union transformation.

To create a Union transformation:

1. In the Mapping Designer, click Transformations > Create.
2. Select Union Transformation and enter the name of the transformation.
The naming convention for Union transformations is *UN_TransformationName*.
3. Enter a description for the transformation. Click Create, and then click Done.
4. Click the Groups tab.
5. Add an input group for each pipeline or pipeline branch you want to merge.
The Designer assigns a default name for each group but they can be renamed.
6. Click the Group Ports tab.

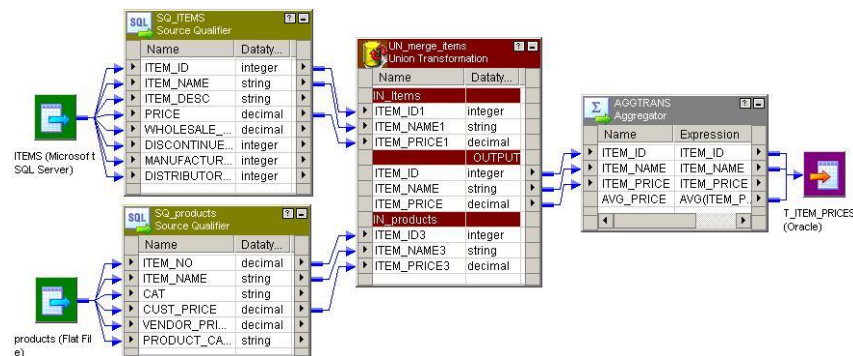
7. Add a new port for each row of data you want to merge.
8. Enter port properties, such as name and datatype.
9. Click the Properties tab to configure the tracing level.
10. Click OK.

Using a Union Transformation in a Mapping

The Union transformation is a non-blocking multiple input group transformation. You can connect the input groups to different branches in a single pipeline or to different source pipelines.

When you add a Union transformation to a mapping, you must verify that you connect the same ports in all input groups. If you connect all ports in one input group, but do not connect a port in another input group, the Integration Service passes NULLs to the unconnected port.

The following figure shows a mapping with a Union transformation:



When a Union transformation in a mapping receives data from a single transaction generator, the Integration Service propagates transaction boundaries. When the transformation receives data from multiple transaction generators, the Integration Service drops all incoming transaction boundaries and outputs rows in an open transaction.

CHAPTER 18

Update Strategy Transformation

This chapter includes the following topics:

- φ Overview
- φ Flagging Rows Within a Mapping
- φ Setting the Update Strategy for a Session
- φ Update Strategy Checklist

Overview

Transformation type:

Active
Connected

When you design a data warehouse, you need to decide what type of information to store in targets. As part of the target table design, you need to determine whether to maintain all the historic data or just the most recent changes.

For example, you might have a target table, T_CUSTOMERS, that contains customer data. When a customer address changes, you may want to save the original address in the table instead of updating that portion of the customer row. In this case, you would create a new row containing the updated address, and preserve the original row with the old customer address. This shows how you might store historical information in a target table. However, if you want the T_CUSTOMERS table to be a snapshot of current customer data, you would update the existing customer row and lose the original address.

The model you choose determines how you handle changes to existing rows. In PowerCenter, you set the update strategy at two different levels:

- φ **Within a session.** When you configure a session, you can instruct the Integration Service to either treat all rows in the same way (for example, treat all rows as inserts), or use instructions coded into the session mapping to flag rows for different database operations.
- φ **Within a mapping.** Within a mapping, you use the Update Strategy transformation to flag rows for insert, delete, update, or reject.

Note: You can also use the Custom transformation to flag rows for insert, delete, update, or reject.

Setting the Update Strategy

To define an update strategy, complete the following steps:

1. To control how rows are flagged for insert, update, delete, or reject within a mapping, add an Update Strategy transformation to the mapping. Update Strategy transformations are essential if you want to flag rows destined for the same target for different database operations, or if you want to reject rows.
2. Define how to flag rows when you configure a session. You can flag all rows for insert, delete, or update, or you can select the data driven option, where the Integration Service follows instructions coded into Update Strategy transformations within the session mapping.
3. Define insert, update, and delete options for each target when you configure a session. On a target-by-target basis, you can allow or disallow inserts and deletes, and you can choose three different ways to handle updates.

RELATED TOPICS:

φ “Setting the Update Strategy for a Session”

Flagging Rows Within a Mapping

For the greatest degree of control over the update strategy, you add Update Strategy transformations to a mapping. The most important feature of this transformation is its update strategy expression, used to flag individual rows for insert, delete, update, or reject.

The following table lists the constants for each database operation and their numeric equivalent:

Operation	Constant	Numeric Value
Insert	DD_INSERT	0
Update	DD_UPDATE	1
Delete	DD_DELETE	2
Reject	DD_REJECT	3

The Integration Service treats any other value as an insert.

Forwarding Rejected Rows

You can configure the Update Strategy transformation to either pass rejected rows to the next transformation or drop them. By default, the Integration Service forwards rejected rows to the next transformation. The Integration Service flags the rows for reject and writes them to the session reject file. If you do not select Forward Rejected Rows, the Integration Service drops rejected rows and writes them to the session log file.

If you enable row error handling, the Integration Service writes the rejected rows and the dropped rows to the row error logs. It does not generate a reject file. If you want to write the dropped rows to the session log in addition to the row error logs, you can enable verbose data tracing.

Update Strategy Expressions

Frequently, the update strategy expression uses the IIF or DECODE function from the transformation language to test each row to see if it meets a particular condition. If it does, you can then assign each row a numeric code to flag it for a particular database operation. For example, the following IIF statement flags a row for reject if the entry date is after the apply date. Otherwise, it flags the row for update:

```
IIF( ( ENTRY_DATE > APPLY_DATE), DD_REJECT, DD_UPDATE )
```

To create an Update Strategy transformation:

1. In the Mapping Designer, add an Update Strategy transformation to a mapping.
2. Click Layout > Link Columns.
3. Drag all ports from another transformation representing data you want to pass through the Update Strategy transformation.

In the Update Strategy transformation, the Designer creates a copy of each port you drag. The Designer also connects the new port to the original port. Each port in the Update Strategy transformation is a combination input/output port.

Normally, you would select all of the columns destined for a particular target. After they pass through the Update Strategy transformation, this information is flagged for update, insert, delete, or reject.

4. Open the Update Strategy transformation and rename it.

The naming convention for Update Strategy transformations is *UPD_TransformationName*.

5. Click the Properties tab.
6. Click the button in the Update Strategy Expression field.

The Expression Editor appears.

7. Enter an update strategy expression to flag rows as inserts, deletes, updates, or rejects.
8. Validate the expression and click OK.
9. Click OK.
10. Connect the ports in the Update Strategy transformation to another transformation or a target instance.

Aggregator and Update Strategy Transformations

When you connect Aggregator and Update Strategy transformations as part of the same pipeline, you have the following options:

- φ **Position the Aggregator before the Update Strategy transformation.** In this case, you perform the aggregate calculation, and then use the Update Strategy transformation to flag rows that contain the results of this calculation for insert, delete, or update.
- φ **Position the Aggregator after the Update Strategy transformation.** Here, you flag rows for insert, delete, update, or reject before you perform the aggregate calculation. How you flag a particular row determines how the Aggregator transformation treats any values in that row used in the calculation. For example, if you flag a row for delete and then later use the row to calculate the sum, the Integration Service subtracts the value appearing in this row. If the row had been flagged for insert, the Integration Service would add its value to the sum.

Lookup and Update Strategy Transformations

When you create a mapping with a Lookup transformation that uses a dynamic lookup cache, you must use Update Strategy transformations to flag the rows for the target tables. When you configure a session using Update Strategy transformations and a dynamic lookup cache, you must define certain session properties.

You must define the Treat Source Rows As option as Data Driven. Specify this option on the Properties tab in the session properties.

You must also define the following update strategy target table options:

- φ Select Insert
- φ Select Update as Update
- φ Do not select Delete

These update strategy target table options ensure that the Integration Service updates rows marked for update and inserts rows marked for insert.

If you do not choose Data Driven, the Integration Service flags all rows for the database operation you specify in the Treat Source Rows As option and does not use the Update Strategy transformations in the mapping to flag the rows. The Integration Service does not insert and update the correct rows. If you do not choose Update as Update, the Integration Service does not correctly update the rows flagged for update in the target table. As a result, the lookup cache and target table might become unsynchronized.

Setting the Update Strategy for a Session

When you configure a session, you have several options for handling database operations, including updates.

Specifying an Operation for All Rows

When you configure a session, you can select a single database operation for all rows using the Treat Source Rows As setting.

Table 28-1 displays the options for the Treat Source Rows As setting:

Table 28-1. Specifying an Operation for All Rows

Setting	Description
Insert	Treat all rows as inserts. If inserting the row violates a primary or foreign key constraint in the database, the Integration Service rejects the row.
Delete	Treat all rows as deletes. For each row, if the Integration Service finds a corresponding row in the target table (based on the primary key value), the Integration Service deletes it. Note that the primary key constraint must exist in the target definition in the repository.
Update	Treat all rows as updates. For each row, the Integration Service looks for a matching primary key value in the target table. If it exists, the Integration Service updates the row. The primary key constraint must exist in the target definition.
Data Driven	Integration Service follows instructions coded into Update Strategy and Custom transformations within the session mapping to determine how to flag rows for insert, delete, update, or reject. If the mapping for the session contains an Update Strategy transformation, this field is marked Data Driven by default. If you do not choose Data Driven when a mapping contains an Update Strategy or Custom transformation, the Workflow Manager displays a warning. When you run the session, the Integration Service does not follow instructions in the Update Strategy or Custom transformation in the mapping to determine how to flag rows.

Table 28-2 describes the update strategy for each setting:

Table 28-2. Update Strategy Settings

Setting	Use To
Insert	Populate the target tables for the first time, or maintain a historical data warehouse. In the latter case, you must set this strategy for the entire data warehouse, not just a select group of target tables.
Delete	Clear target tables.
Update	Update target tables. You might choose this setting whether the data warehouse contains historical data or a snapshot. Later, when you configure how to update individual target tables, you can determine whether to insert updated rows as new rows or use the updated information to modify existing rows in the target.
Data Driven	Exert finer control over how you flag rows for insert, delete, update, or reject. Choose this setting if rows destined for the same table need to be flagged on occasion for one operation (for example, update), or for a different operation (for example, reject). In addition, this setting provides the only way you can flag rows for reject.

Specifying Operations for Individual Target Tables

Once you determine how to treat all rows in the session, you also need to set update strategy options for individual targets. Define the update strategy options in the Transformations view on Mapping tab of the session properties.

You can set the following update strategy options:

- φ **Insert.** Select this option to insert a row into a target table.
- φ **Delete.** Select this option to delete a row from a table.
- φ **Update.** You have the following options in this situation:
 - ☐ **Update as Update.** Update each row flagged for update if it exists in the target table.
 - ☐ **Update as Insert.** Insert each row flagged for update.
 - ☐ **Update else Insert.** Update the row if it exists. Otherwise, insert it.
- φ **Truncate table.** Select this option to truncate the target table before loading data.

Update Strategy Checklist

Choosing an update strategy requires setting the right options within a session and possibly adding Update Strategy transformations to a mapping. This section summarizes what you need to implement different versions of an update strategy.

Only perform inserts into a target table.

When you configure the session, select Insert for the Treat Source Rows As session property. Also, make sure that you select the Insert option for all target instances in the session.

Delete all rows in a target table.

When you configure the session, select Delete for the Treat Source Rows As session property. Also, make sure that you select the Delete option for all target instances in the session.

Only perform updates on the contents of a target table.

When you configure the session, select Update for the Treat Source Rows As session property. When you configure the update options for each target table instance, make sure you select the Update option for each target instance.

Perform different database operations with different rows destined for the same target table.

Add an Update Strategy transformation to the mapping. When you write the transformation update strategy expression, use either the DECODE or IIF function to flag rows for different operations (insert, delete, update, or reject). When you configure a session that uses this mapping, select Data Driven for the Treat Source Rows As session property. Make sure that you select the Insert, Delete, or one of the Update options for each target table instance.

Reject data.

Add an Update Strategy transformation to the mapping. When you write the transformation update strategy expression, use DECODE or IIF to specify the criteria for rejecting the row. When you configure a session that uses this mapping, select Data Driven for the Treat Source Rows As session property.

XML Transformations

This chapter includes the following topics:

- φ XML Source Qualifier Transformation
- φ XML Parser Transformation
- φ XML Generator Transformation

XML Source Qualifier Transformation

Transformation type:

Active
Connected

You can add an XML Source Qualifier transformation to a mapping by dragging an XML source definition to the Mapping Designer workspace or by manually creating one. When you add an XML source definition to a mapping, you need to connect it to an XML Source Qualifier transformation. The XML Source Qualifier transformation defines the data elements that the Integration Service reads when it executes a session. It determines how the PowerCenter reads the source data.

An XML Source Qualifier transformation always has one input or output port for every column in the XML source. When you create an XML Source Qualifier transformation for a source definition, the Designer links each port in the XML source definition to a port in the XML Source Qualifier transformation. You cannot remove or edit any of the links. If you remove an XML source definition from a mapping, the Designer also removes the corresponding XML Source Qualifier transformation. You can link one XML source definition to one XML Source Qualifier transformation.

You can link ports of one XML Source Qualifier group to ports of different transformations to form separate data flows. However, you cannot link ports from more than one group in an XML Source Qualifier transformation to ports in the same target transformation.

You can edit some of the properties and add metadata extensions to an XML Source Qualifier transformation.

For more information about the XML Source Qualifier transformation, see the *PowerCenter XML Guide*.

XML Parser Transformation

Transformation type:

Active
Connected

Use an XML Parser transformation to extract XML inside a pipeline. The XML Parser transformation lets you extract XML data from messaging systems, such as TIBCO or MQ Series, and from other sources, such as files or databases. The XML Parser transformation functionality is similar to the XML source functionality, except it parses the XML in the pipeline. For example, you might want to extract XML data from a TIBCO source and pass the data to relational targets.

The XML Parser transformation reads XML data from a single input port and writes data to one or more output ports.

For more information about the XML Parser transformation, see the *PowerCenter XML Guide*.

XML Generator Transformation

Transformation type:

Active
Connected

Use an XML Generator transformation to create XML inside a pipeline. The XML Generator transformation lets you read data from messaging systems, such as TIBCO and MQ Series, or from other sources, such as files or databases. The XML Generator transformation functionality is similar to the XML target functionality, except it generates the XML in the pipeline. For example, you might want to extract data from relational sources and pass XML data to targets.

The XML Generator transformation accepts data from multiple ports and writes XML through a single output port.

CHAPTER 20

Mapplets

This chapter includes the following topics:

- ϕ Overview
- ϕ Understanding Mapplet Input and Output
- ϕ Using the Mapplet Designer
- ϕ Using Mapplets in Mappings

Overview

A mapplet is a reusable object that you create in the Mapplet Designer. It contains a set of transformations and lets you reuse the transformation logic in multiple mappings.

For example, if you have several fact tables that require a series of dimension keys, you can create a mapplet containing a series of Lookup transformations to find each dimension key. You can then use the mapplet in each fact table mapping, rather than recreate the same lookup logic in each mapping.

When you use a mapplet in a mapping, you use an instance of the mapplet. Like a reusable transformation, any change made to the mapplet is inherited by all instances of the mapplet.

Mapplets help simplify mappings in the following ways:

- ϕ **Include source definitions.** Use multiple source definitions and source qualifiers to provide source data for a mapping.
- ϕ **Accept data from sources in a mapping.** If you want the mapplet to receive data from the mapping, use an Input transformation to receive source data.
- ϕ **Include multiple transformations.** A mapplet can contain as many transformations as you need.
- ϕ **Pass data to multiple transformations.** You can create a mapplet to feed data to multiple transformations. Each Output transformation in a mapplet represents one output group in a mapplet.
- ϕ **Contain unused ports.** You do not have to connect all mapplet input and output ports in a mapping.

Understanding Mapplet Input and Output

To use a mapplet in a mapping, you must configure it for input and output. In addition to transformation logic that you configure, a mapplet has the following components:

- ϕ **Mapplet input.** You can pass data into a mapplet using source definitions or Input transformations or both. When you use an Input transformation, you connect it to the source pipeline in the mapping.
- ϕ **Mapplet output.** Each mapplet must contain one or more Output transformations to pass data from the mapplet into the mapping.
- ϕ **Mapplet ports.** Mapplet ports display only in the Mapping Designer. Mapplet ports consist of input ports from Input transformations and output ports from Output transformations. If a mapplet uses source definitions rather than Input transformations for input, it does not contain any input ports in the mapping.

Mapplet Input

Mapplet input can originate from a source definition and/or from an Input transformation in the mapplet. You can create multiple pipelines in a mapplet. Use multiple source definitions and source qualifiers or Input transformations. You can also use a combination of source definitions and Input transformations.

Using Source Definitions for Mapplet Input

Use one or more source definitions in a mapplet to provide source data. When you use the mapplet in a mapping, it is the first object in the mapping pipeline and contains no input ports.

Using Input Transformations for Mapplet Input

Use an Input transformation in a mapplet when you want the mapplet to receive input from a source in a mapping. When you use the mapplet in a mapping, the Input transformation provides input ports so you can pass data through the mapplet. Each port in the Input transformation connected to another transformation in the mapplet becomes a mapplet input port. Input transformations can receive data from a single active source. Unconnected ports do not display in the Mapping Designer.

You can connect an Input transformation to multiple transformations in a mapplet. However, you cannot connect a single port in the Input transformation to multiple transformations in the mapplet.

Mapplet Output

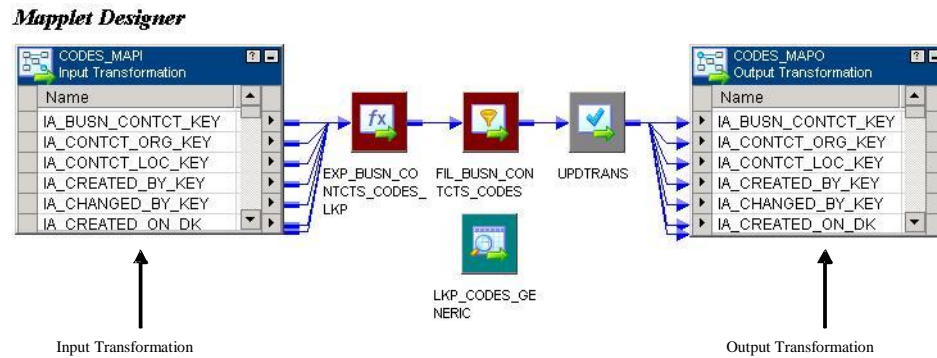
Use an Output transformation in a mapplet to pass data through the mapplet into a mapping. A mapplet must contain at least one Output transformation with at least one connected port in the mapplet. Each connected port in an Output transformation displays as a mapplet output port in a mapping. Each Output transformation in a mapplet displays as an output group in a mapping. An output group can pass data to multiple pipelines in a mapping.

Viewing Mapplet Input and Output

Mapplets and mapplet ports display differently in the Mapplet Designer and the Mapping Designer.

Figure 6-1 shows a mapplet with both an Input transformation and an Output transformation:

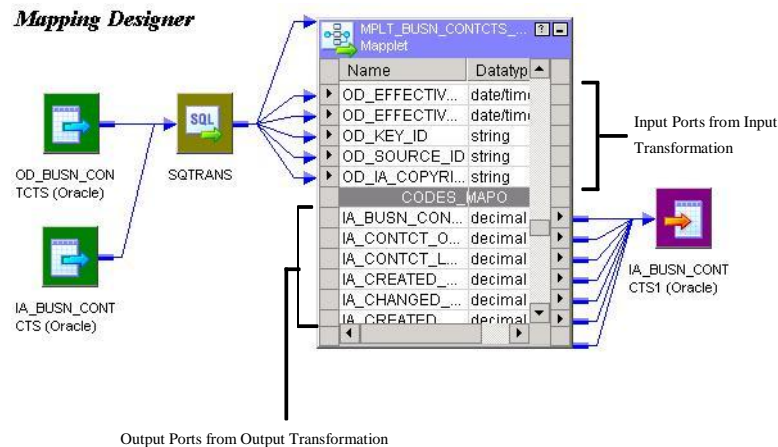
Figure 6-1. Sample Mapplet



When you use the mapplet in a mapping, the mapplet object displays only the ports from the Input and Output transformations. These are referred to as the mapplet input and mapplet output ports.

Figure 6-2 shows a mapplet in the Mapping Designer that contains the output ports from the Output Transformation in Figure 6-1:

Figure 6-2. Sample Mapplet in a Mapping

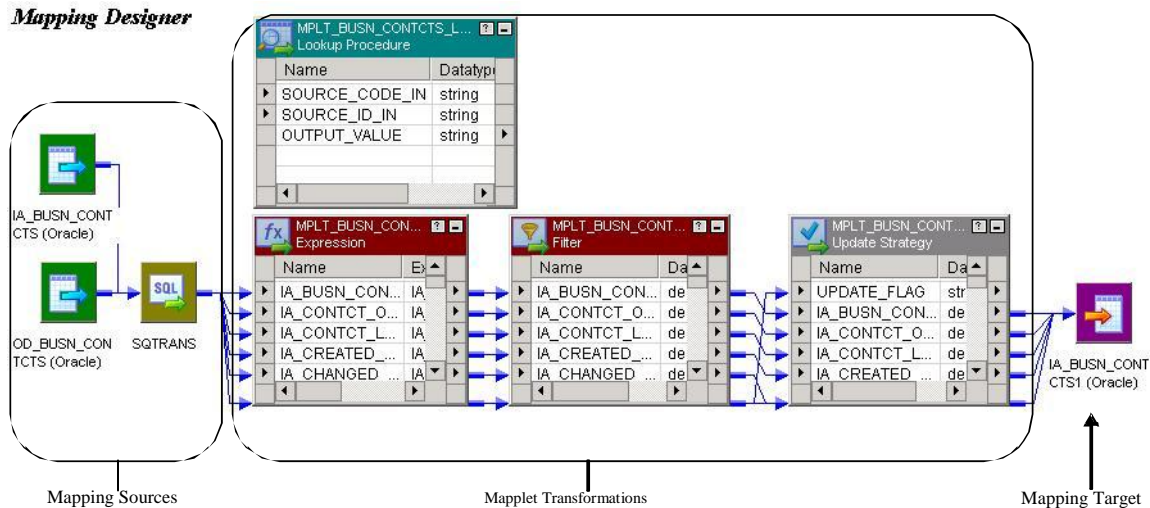


You can expand the mapplet in the Mapping Designer by selecting it and clicking **Mappings > Expand**. This expands the mapplet within the mapping for view. Transformation icons within an expanded mapplet display as shaded.

You can open or iconize all the transformations in the mapplet and mapping. You cannot edit any of the properties, navigate to other folders, or save the repository while the mapplet is expanded.

Figure 6-3 shows an expanded mapplet in the Mapping Designer:

Figure 6-3. Expanded Mapplet in a Mapping



In an expanded mapping, you do not see the Input and Output transformations.

Using the Mapplet Designer

After you create a mapplet, you can validate or edit the mapplet in the Mapplet Designer. You can also use the Designer to copy mapplets, export and import mapplets, view links between ports in a mapplet, create shortcuts to mapplets, and delete mapplets from the repository.

To create and configure a mapplet in the Mapplet Designer, complete the following steps:

1. Create a mapplet. Click Mapplets > Create from the menu in the Mapplet Designer. The recommended naming convention for mapplets is *mplt_MappletName*.
2. Create mapplet transformation logic. Create and link transformations in the same manner as in a mapping.
3. Create mapplet ports.

Creating a Mapplet

A mapplet can be active or passive depending on the transformations in the mapplet. Active mapplets contain one or more active transformations. Passive mapplets contain only passive transformations. When you use a mapplet in a mapping, all transformation rules apply to the mapplet depending on the mapplet type. For example, as with an active transformation, you cannot concatenate data from an active mapplet with a different pipeline.

Use the following rules and guidelines when you add transformations to a mapplet:

- φ If you use a Sequence Generator transformation, you must use a reusable Sequence Generator transformation.
- φ If you use a Stored Procedure transformation, you must configure the Stored Procedure Type to be *Normal*.
- φ You cannot include PowerMart 3.5-style LOOKUP functions in a mapplet.
- φ You cannot include the following objects in a mapplet:
 - Normalizer transformations
 - COBOL sources

- ❑ XML Source Qualifier transformations
- ❑ XML sources
- ❑ Target definitions
- ❑ Other mapplets

Although reusable transformations and shortcuts in a mapplet can be used, to protect the validity of the mapplet, use a copy of a transformation instead. Reusable transformations and shortcuts inherit changes to their original transformations. This might invalidate the mapplet and the mappings that use the mapplet.

Validating Mapplets

The Designer validates a mapplet when you save it. You can also validate a mapplet using the Mapplets > Validate menu command. When you validate a mapplet, the Designer writes all relevant messages about the mapplet in the Output window.

The Designer validates the mapplet pipeline in the same way it validates a mapping. The Designer also performs the following checks specific to mapplets:

- ϕ The mapplet can contain Input transformations and source definitions with at least one port connected to a transformation in the mapplet.
- ϕ The mapplet contains at least one Output transformation with at least one port connected to a transformation in the mapplet.

Editing Mapplets

You can edit a mapplet in the Mapplet Designer. The Designer validates the changes when you save the mapplet. When you save changes to a mapplet, all instances of the mapplet and all shortcuts to the mapplet inherit the changes. These changes might invalidate mappings that use the mapplet.

To see what mappings or shortcuts may be affected by changes you make to a mapplet, select the mapplet in the Navigator, right-click, and select Dependencies. Or, click Mapplets > Dependencies from the menu.

You can make the following changes to a mapplet *without* affecting the validity of existing mappings and sessions:

- ϕ Add input or output ports.
- ϕ Change port names or comments.
- ϕ Change Input or Output transformation names or comments.
- ϕ Change transformation names, comments, or properties.
- ϕ Change port default values for transformations in the mapplet.
- ϕ Add or remove transformations in the mapplet, providing you do not change the mapplet type from active to passive or from passive to active.

Use the following rules and guidelines when you edit a mapplet that is used by mappings:

- ϕ **Do not delete a port from the mapplet.** The Designer deletes mapplet ports in the mapping when you delete links to an Input or Output transformation or when you delete ports connected to an Input or Output transformation.
- ϕ **Do not change the datatype, precision, or scale of a mapplet port.** The datatype, precision, and scale of a mapplet port is defined by the transformation port to which it is connected in the mapplet. Therefore, if you edit a mapplet to change the datatype, precision, or scale of a port *connected* to a port in an Input or Output transformation, you change the mapplet port.
- ϕ **Do not change the mapplet type.** If you remove all active transformations from an active mapplet, the mapplet becomes passive. If you add an active transformation to a passive mapplet, the mapplet becomes active.

Mapplets and Mappings

The following mappings tasks can also be performed on mapplets:

- φ **Set tracing level.** You can set the tracing level on individual transformations within a mapplet in the same manner as in a mapping.
- φ **Copy mapplet.** You can copy a mapplet from one folder to another as you would any other repository object. After you copy the mapplet, it appears in the Mapplets node of the new folder.
If you make changes to a mapplet, but you do not want to overwrite the original mapplet, you can make a copy of the mapplet by clicking Mapplets > Copy As.
- φ **Export and import mapplets.** You can export a mapplet to an XML file or import a mapplet from an XML file through the Designer. You might want to use the export and import feature to copy a mapplet to another repository.
- φ **Delete mapplets.** When you delete a mapplet, you delete all instances of the mapplet. This invalidates each mapping containing an instance of the mapplet or a shortcut to the mapplet.
- φ **Compare mapplets.** You can compare two mapplets to find differences between them. For example, if you have mapplets with the same name in different folders, you can compare them to see if they differ.
- φ **Compare instances within a mapplet.** You can compare instances in a mapplet to see if they contain similar attributes. For example, you can compare a source instance with another source instance, or a transformation with another transformation. You compare instances within a mapplet in the same way you compare instances within a mapping.
- φ **Create shortcuts to mapplets.** You can create a shortcut to a mapplet if the mapplet is in a shared folder. When you use a shortcut to a mapplet in a mapping, the shortcut inherits any changes you might make to the mapplet. However, these changes might not appear until the Integration Service runs the workflow using the shortcut. Therefore, only use a shortcut to a mapplet when you do not expect to edit the mapplet.
- φ **Add a description.** You can add a description to the mapplet in the Mapplet Designer in the same manner as in a mapping. You can also add a description to the mapplet instance in a mapping. When you add a description, you can also create links to documentation files. The links must be a valid URL or file path to reference the business documentation.
- φ **View links to a port.** You can view links to a port in a mapplet in the same way you would view links to a port in a mapping. You can view the forward path, the backward path, or both paths.
- φ **Propagate port attributes.** You can propagate port attributes in a mapplet in the same way you would propagate port attributes in a mapping. You can propagate attributes forward, backward, or in both directions.

Using Mapplets in Mappings

In a mapping, a mapplet has input and output ports that you can connect to other transformations in the mapping. You do not have to connect all mapplet ports in a mapping. However, if the mapplet contains an SQL override, you must connect all mapplet output ports in the mapping.

Like a reusable transformation, when you drag a mapplet into a mapping, the Designer creates an instance of the mapplet. You can enter comments for the instance of the mapplet in the mapping. You cannot otherwise edit the mapplet in the Mapping Designer.

If you edit the mapplet in the Mapplet Designer, each instance of the mapplet inherits the changes.

The PowerCenter Repository Reports has a Mapplets list report that you use to view all mappings using a particular mapplet.

To use a mapplet, complete the following steps:

1. Drag the mapplet into the mapping.
2. If the mapplet contains input ports, connect at least one mapplet input port to a transformation in the mapping.
3. Connect at least one mapplet output port to a transformation in the mapping.

Creating and Configuring Mapplet Ports

After creating transformation logic for a mapplet, you can create mapplet ports. Use an Input transformation to define mapplet input ports if the mapplet contains no source definitions. Use an Output transformation to create a group of output ports. Only connected ports in an Input or Output transformation become mapplet input or output ports in a mapping. Unconnected ports do not display when you use the mapplet in a mapping.

You can create a mapplet port in the following ways:

- φ **Manually create ports in the Input/Output transformation.** You can create port names in Input and Output transformations. You can also enter a description for each port name. The port has no defined datatype, precision, or scale until you connect it to a transformation in the mapplet.
- φ **Drag a port from another transformation.** You can create an input or output port by dragging a port from another transformation into the Input or Output transformation. The new port inherits the port name, description, datatype, and scale of the original port. You can edit the new port name and description in the transformation. If you change a port connection, the Designer updates the Input or Output transformation port to match the attributes of the new connection.

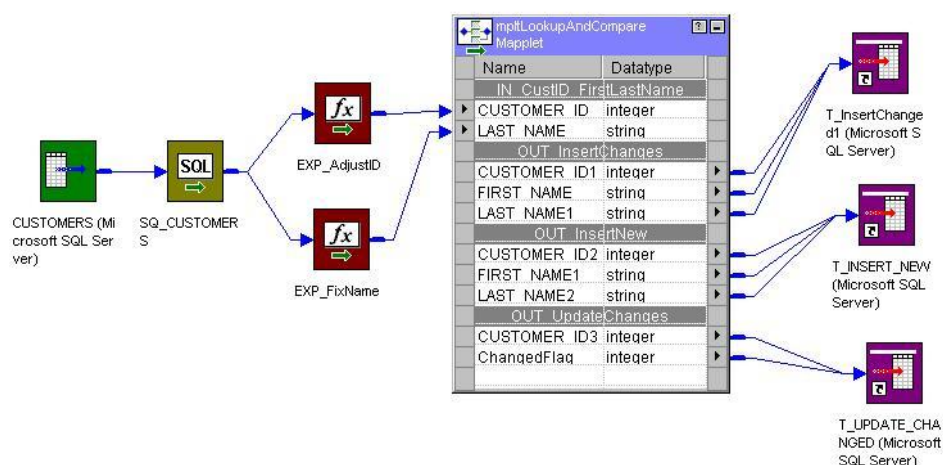
You can view the datatype, precision, and scale of available mapplet ports when you use the mapplet in a mapping.

Connecting to Mapplet Input Ports

When using a mapplet with input ports in a mapping, you connect the mapplet input ports to the mapping pipeline. You can pass data into a mapplet only when it originates from a single active transformation.

For example, in the following figure, the mapplet `mpLkLookupAndCompare` accepts data from two Expression transformations because data from both transformations originate from a single source qualifier. The Source Qualifier `SQ_CUSTOMERS` is the active transformation providing mapplet source data:

Mapping Designer



Connecting to Mapplet Output Groups

Each Output transformation displays as an output group when you use a mapplet in a mapping. Connect the mapplet output ports to the mapping pipeline. Use Autolink to connect the ports.

Use the following rules and guidelines when you connect mapplet output ports in the mapping:

- ϕ When a mapplet contains a source qualifier that has an override for the default SQL query, you must connect all of the source qualifier output ports to the next transformation within the mapplet.
- ϕ If the mapplet contains more than one source qualifier, use a Joiner transformation to join the output into one pipeline.
- ϕ If the mapplet contains only one source qualifier, you must connect the mapplet output ports to separate pipelines. You cannot use a Joiner transformation to join the output.

If you need to join the pipelines, you can create two mappings to perform this task:

- Use the mapplet in the first mapping and write data in each pipeline to separate targets.
- Use the targets as sources in the second mapping to join data, then perform any additional transformation necessary.

Viewing the Mapplet

When you use a mapplet in a mapping, the Designer displays the mapplet object, which contains only the input and output ports of the mapplet. However, you can expand the mapplet by clicking Mappings > Expand from the menu.

When the Designer expands the mapplet, it displays the entire mapping with the mapplet transformations. It does not display the Input and Output transformations. You can view the mapping in this expanded form, but you cannot edit it. To continue designing the mapping, click Mappings > Unexpand.

Setting the Target Load Plan

When you use a mapplet in a mapping, the Mapping Designer lets you set the target load plan for sources within the mapplet.

Pipeline Partitioning

If you have the partitioning option, you can increase the number of partitions in a pipeline to improve session performance. Increasing the number of partitions allows the Integration Service to create multiple connections to sources and process partitions of source data concurrently.

When you create a session, the Workflow Manager validates each pipeline in the mapping for partitioning. You can specify multiple partitions in a pipeline if the Integration Service can maintain data consistency when it processes the partitioned data.

There are partitioning restrictions that apply to mapplets. For more information about restrictions for partitioning data, see the *PowerCenter Workflow Administration Guide*.

CHAPTER 21

Mapping Parameters and Variables

This chapter includes the following topics:

- φ Overview
- φ Mapping Parameters
- φ Mapping Variables

Overview

In the Designer, use mapping parameters and variables to make mappings more flexible. Mapping parameters and variables represent values in mappings and maplets. If you declare mapping parameters and variables in a mapping, you can reuse a mapping by altering the parameter and variable values of the mapping in the session. This can reduce the overhead of creating multiple mappings when only certain attributes of a mapping need to be changed.

When you use a mapping parameter or variable in a mapping, first you declare the mapping parameter or variable for use in each maplet or mapping. Then, you define a value for the mapping parameter or variable before you run the session.

Use mapping parameters and variables in a mapping to incrementally extract data. Use mapping parameters or variables in the source filter of a Source Qualifier transformation to determine the beginning timestamp and end timestamp for incrementally extracting data.

For example, you can create a user-defined mapping variable `$$LastUpdateDateTime` that saves the timestamp of the last row the Integration Service read in the previous session. Use `$$LastUpdateDateTime` for the beginning timestamp and the built-in variable `$$$SessStartTime` for the end timestamp in the source filter. Use the following filter to incrementally extract data based on the `SALES.sales_datetime` column in the source:

```
SALES.sales_datetime > TO_DATE ('$$LastUpdateDateTime') AND SALES.sales_datetime <
TO_DATE ('$$$SessStartTime')
```

Mapping Parameters

A mapping parameter represents a constant value that you can define before running a session. A mapping parameter retains the same value throughout the entire session.

When you use a mapping parameter, you declare and use the parameter in a mapping or mapplet. Then define the value of the parameter in a parameter file. The Integration Service evaluates all references to the parameter to that value.

For example, you want to use the same session to extract transaction records for each of the customers individually. Instead of creating a separate mapping for each customer account, you can create a mapping parameter to represent a single customer account. Then use the parameter in a source filter to extract only data for that customer account. Before running the session, you enter the value of the parameter in the parameter file.

To reuse the same mapping to extract records for other customer accounts, you can enter a new value for the parameter in the parameter file and run the session. Or, you can create a parameter file for each customer account and start the session with a different parameter file each time using *pmcmd*. By using a parameter file, you reduce the overhead of creating multiple mappings and sessions to extract transaction records for different customer accounts.

When you want to use the same value for a mapping parameter each time you run the session, use the same parameter file for each session run. When you want to change the value of a mapping parameter between sessions you can perform one of the following tasks:

- ϕ Update the parameter file between sessions.
- ϕ Create a different parameter file and configure the session to use the new file.
- ϕ Remove the parameter file from the session properties. The Integration Service uses the parameter value in the pre-session variable assignment. If there is no pre-session variable assignment, the Integration Service uses the configured initial value of the parameter in the mapping.

Mapping Variables

Unlike a mapping parameter, a mapping variable represents a value that can change through the session. The Integration Service saves the value of a mapping variable to the repository at the end of each successful session run and uses that value the next time you run the session.

When you use a mapping variable, you declare the variable in the mapping or mapplet, and then use a variable function in the mapping to change the value of the variable. At the beginning of a session, the Integration Service evaluates references to a variable to determine the start value. At the end of a successful session, the Integration Service saves the final value of the variable to the repository. The next time you run the session, the Integration Service evaluates references to the variable to the saved value. To override the saved value, define the start value of the variable in a parameter file or assign a value in the pre-session variable assignment in the session properties.

Use mapping variables to perform incremental reads of a source. For example, the customer accounts in the mapping parameter example above are numbered from 001 to 065, incremented by one. Instead of creating a mapping parameter, you can create a mapping variable with an initial value of 001. In the mapping, use a variable function to increase the variable value by one. The first time the Integration Service runs the session, it extracts the records for customer account 001. At the end of the session, it increments the variable by one and saves that value to the repository. The next time the Integration Service runs the session, it extracts the data for the next customer account, 002. It also increments the variable value so the next session extracts and looks up data for customer account 003.

Using Mapping Parameters and Variables

You can create mapping parameters and variables in the Mapping Designer or Mapplet Designer. Once created, mapping parameters and variables appear on the Variables tab of the Expression Editor. Use them in any expression in the mapplet or mapping. The Designer validates mapping parameters and variables in the Expression Editor of mapplets and mappings.

Use mapping parameters and variables in a source qualifier in a mapplet or mapping. When you use mapping parameters and variables in a Source Qualifier transformation, the Designer expands them before passing the query to the source database for validation. This allows the source database to validate the query.

When you create a reusable transformation in the Transformation Developer, use any mapping parameter or variable. Since a reusable transformation is not contained within any mapplet or mapping, the Designer validates the usage of any mapping parameter or variable in the expressions of reusable transformation. When you use the reusable transformation in a mapplet or mapping, the Designer validates the expression again. If the parameter or variable is not defined in the mapplet or mapping, or if it is used incorrectly in the reusable transformation, the Designer logs an error when you validate the mapplet or mapping.

When the Designer validates a mapping variable in a reusable transformation, it treats the variable as an Integer datatype.

You cannot use mapping parameters and variables interchangeably between a mapplet and a mapping. Mapping parameters and variables declared for a mapping cannot be used within a mapplet. Similarly, you cannot use a mapping parameter or variable declared for a mapplet in a mapping.

Initial and Default Values

When you declare a mapping parameter or variable in a mapping or a mapplet, you can enter an initial value. The Integration Service uses the configured initial value for a mapping parameter when the parameter is not defined in the parameter file. Similarly, the Integration Service uses the configured initial value for a mapping variable when the variable value is not defined in the parameter file, and there is no saved variable value in the repository.

When the Integration Service needs an initial value, and you did not declare an initial value for the parameter or variable, the Integration Service uses a default value based on the datatype of the parameter or variable.

Table 7-1 lists the default values the Integration Service uses for different types of data:

Table 7-1. Default Values for Mapping Parameters and Variables Based on Datatype

Data	Default Value
String	Empty string.
Numeric	0
Datetime	1/1/1753 A.D. or 1/1/1 when the Integration Service is configured for compatibility with 4.0.

For example, you create a new mapping using an Integer mapping variable, `$$MiscellaneousExpenses`. You do not configure an initial value for the variable or define it in a parameter file. The first time you run a session with the mapping, the Integration Service uses the default value for numeric datatypes, 0.

Or, if you create a mapping parameter `$$MiscellaneousCosts` to represent additional expenses that might become relevant in the future, but do not currently exist in source data. You configure the parameter for a Decimal datatype. Since you want `$$MiscellaneousCosts` to evaluate to 0 when you do not have additional expenses, you set the initial value to 0.

As long as you do not define the parameter value in the parameter file, the Integration Service replaces `$$MiscellaneousCosts` with 0. When you want to include miscellaneous expenses in mapping calculations, set `$$MiscellaneousCosts` to that value in the parameter file.

Using String Parameters and Variables

When you enter mapping parameters and variables of a string datatype in a Source Qualifier transformation, use a string identifier appropriate for the source database. When the Integration Service expands a parameter or variable in a Source Qualifier transformation, the Integration Service replaces it with its start value, and then passes the expanded query to the source database. Most databases require single quotation marks around string values.

When you enter string parameters or variables using the PowerCenter transformation language, do not use additional quotes. The Integration Service recognizes mapping parameter and variable naming syntax in the PowerCenter transformation language. For example, you might use a parameter named `$$State` in the filter for a Source Qualifier transformation to extract rows for a particular state:

```
STATE = '$$State'
```

During the session, the Integration Service replaces the parameter with a string. If \$\$State is defined as MD in the parameter file, the Integration Service replaces the parameter as follows:

```
STATE = 'MD'
```

You can perform a similar filter in the Filter transformation using the PowerCenter transformation language as follows:

```
STATE = $$State
```

If you enclose the parameter in single quotes in the Filter transformation, the Integration Service reads it as the string literal “\$\$State” instead of replacing the parameter with “MD.”

Using Datetime Parameters and Variables

When you use a datetime parameter or variable in the Source Qualifier transformation, you might need to change the date format to the format used in the source.

Code Page Relaxation

You can configure the Integration Service to relax code page validation when you run the Integration Service in Unicode data movement mode. However, you might get unexpected results in the following situations:

- ❖ The mapping variable value that the Integration Service saves in the repository is not compatible with the repository code page.

For example, the repository uses the ISO 8859-1 Latin1 code page and you configure the Integration Service to relax code page validation. If the mapping variable value contains Japanese character data, such as JapanEUC, the saved mapping variable value in the repository could be incorrect. There could be data loss converting from the JapanEUC code page to the Latin1 code page. Make sure the saved mapping variable value is two-way compatible with the repository code page.

To ensure the Integration Service can write all metadata to the repository, use 7-bit ASCII characters for all repository metadata or use UTF-8 for the repository.

- ❖ The parameter file contains characters that are not compatible with the Integration Service code page.

The Integration Service interprets the data in the parameter file using the Integration Service code page. For example, the Integration Service uses the ISO 8859-1 Latin1 code page and you configure the Integration Service to relax code page validation. If you create a parameter file and use Greek character data, such as ISO 8859-7, the value the Integration Service reads from the file could be incorrect. There could be data loss converting from the ISO 8859-7 code page to the Latin1 code page. Make sure the characters in the parameter file are a subset of the Integration Service code page.

Mapping Parameters

In the Designer, you can create a mapping parameter in a maplet or mapping. After you create a parameter, it appears in the Expression Editor. You can then use the parameter in any expression in the maplet or mapping. You can also use parameters in a source qualifier filter, user-defined join, or extract override, and in the Expression Editor of reusable transformations.

Before you run a session, define the mapping parameter value in a parameter file for the session. Use any constant value. During the session, the Integration Service evaluates all references to the parameter to the specified value. If the parameter is not defined in the parameter file, the Integration Service uses the user-defined initial value for the parameter. If the initial value is not defined, the Integration Service uses a default value based on the datatype of the mapping parameter.

You can change the value of a mapping parameter between sessions by editing the parameter file or by changing the parameter file used by the session.

You might use a mapping parameter instead of a database lookup. For example, you want to perform calculations using monthly gross earnings. Instead of using a Lookup transformation to connect to a database table for that information, you can create a gross earnings mapping parameter and update its value in the parameter file each month to reflect current earnings.

You might also use a mapping parameter in conjunction with a session parameter to reuse a mapping and session. For example, you have transactional data from different states stored in the same table in different databases, and you want to perform the same calculations on all data, while changing the state sales tax accordingly. Instead of creating a separate mapping and session for each state, you can create one mapping with a sales tax mapping parameter and a session using a source database connection session parameter. You can then create a different parameter file for each state. Before running the session, you can change the parameter file the Integration Service uses by entering a different parameter file name from *pmcmd* or by editing the session in the Workflow Manager.

To use a mapping parameter, complete the following steps:

1. Create a mapping parameter.
2. Use the mapping parameter.
3. Define the parameter value.

Step 1. Create a Mapping Parameter

You can create mapping parameters for any mapping or maplet. You can declare as many mapping parameters as you need. Once declared, use the parameter in the mapping or maplet.

To create a mapping parameter:

1. In the Mapping Designer, click Mappings > Parameters and Variables.
-or-
In the Maplet Designer, click Maplet > Parameters and Variables.
2. Click the Add button.
3. Enter the following information and click OK:

Table 7-2. Options for Declaring Mapping Parameters

Field	Description
Name	Parameter name. Name parameters <i>\$\$ParameterName</i> . The syntax for the parameter name must be \$\$ followed by any alphanumeric or underscore characters.
Type	Variable or parameter. Select Parameter.
Datatype	Datatype of the parameter. Select a valid transformation datatype. Use any datatype except Binary or Raw.
Precision or Scale	Precision and scale of the parameter.
Aggregation	Use for variables.
IsExprVar	Determines how the Integration Service expands the parameter in an expression string. If true, the Integration Service expands the parameter before parsing the expression. If false, the Integration Service expands the parameter after parsing the expression. Default is false. Note: If you set this field to true, you must set the parameter datatype to String, or the Integration Service fails the session.

Table 7-2. Options for Declaring Mapping Parameters

Field	Description
Initial Value	<p>Initial value of the parameter. If you do not set a value for the parameter in the parameter file, the Integration Service uses this value for the parameter during sessions. If this value is also undefined, the Integration Service uses a default value based on the datatype of the mapping variable.</p> <p>Use any of the following formats for initial values for Date/Time parameters:</p> <ul style="list-style-type: none"> - MM/DD/RR - MM/DD/RR HH24:MI:SS - MM/DD/YYYY - MM/DD/YYYY HH24:MI:SS.US
Description	Description associated with the parameter.

Step 2. Use a Mapping Parameter

After you create a parameter, use it in the Expression Editor of any transformation in a mapping or mapplet. You can also use it in Source Qualifier transformations and reusable transformations.

In a Source Qualifier transformation, mapping parameters appear on the Variables tab in the SQL Editor. Use the following rules and guidelines when you use mapping parameters in a Source Qualifier transformation:

- ϕ Enclose string parameters in string identifiers appropriate to the source system.
- ϕ When necessary, change the format of the datetime parameter to match the format in the source.

You can also use mapping parameters in the Expression Editor. When using mapping parameters in the Expression Editor, do not enclose string parameters in string identifiers. The Integration Service handles parameters just like any other port identifiers.

Use mapping parameters in reusable transformations.

You can also use mapping parameters in transformation overrides in the session properties in the Workflow Manager. You can override properties such as a filter or user-defined join in a Source Qualifier transformation.

Step 3. Define a Parameter Value

Before you run a session, define values for mapping parameters in the parameter file. When you do not define a parameter in the parameter file, the Integration Service gets the parameter value from another place. The Integration Service looks for the value in the following order:

1. Value in parameter file
2. Value in pre-session variable assignment
3. Value saved in the repository
4. Initial value
5. Datatype default value

Mapping Variables

In the Designer, you can create mapping variables in a mapping or mapplet. After you create a mapping variable, it appears in the Expression Editor. You can then use it in any expression in the mapping or mapplet. You can also use mapping variables in a source qualifier filter, user-defined join, or extract override, and in the Expression Editor of reusable transformations.

Unlike mapping parameters, mapping variables are values that can change between sessions. The Integration Service saves the latest value of a mapping variable to the repository at the end of each successful session. During the next session run, it evaluates all references to the mapping variable to the saved value. You can override a saved value with the parameter file. You can also clear all saved values for the session in the Workflow Manager.

You might use a mapping variable to perform an incremental read of the source. For example, you have a source table containing timestamped transactions and you want to evaluate the transactions on a daily basis. Instead of manually entering a session override to filter source data each time you run the session, you can create a mapping variable, `$$IncludeDateTime`. In the source qualifier, create a filter to read only rows whose transaction date equals `$$IncludeDateTime`, such as:

```
TIMESTAMP = $$IncludeDateTime
```

In the mapping, use a variable function to set the variable value to increment one day each time the session runs. If you set the initial value of `$$IncludeDateTime` to 8/1/2004, the first time the Integration Service runs the session, it reads only rows dated 8/1/2004. During the session, the Integration Service sets `$$IncludeDateTime` to 8/2/2004. It saves 8/2/2004 to the repository at the end of the session. The next time it runs the session, it reads only rows from August 2, 2004.

Variable Values

The Integration Service holds two different values for a mapping variable during a session run:

- φ Start value of a mapping variable
- φ Current value of a mapping variable

Start Value

The start value is the value of the variable at the start of the session. The start value could be a value defined in the parameter file for the variable, a value assigned in the pre-session variable assignment, a value saved in the repository from the previous run of the session, a user defined initial value for the variable, or the default value based on the variable datatype. The Integration Service looks for the start value in the following order:

1. Value in parameter file
2. Value in pre-session variable assignment
3. Value saved in the repository
4. Initial value
5. Datatype default value

For example, you create a mapping variable in a mapping or mapplet and enter an initial value, but you do not define a value for the variable in a parameter file. The first time the Integration Service runs the session, it evaluates the start value of the variable to the configured initial value. The next time the session runs, the Integration Service evaluates the start value of the variable to the value saved in the repository. If you want to override the value saved in the repository before running a session, you need to define a value for the variable in a parameter file. When you define a mapping variable in the parameter file, the Integration Service uses this value instead of the value saved in the repository or the configured initial value for the variable.

Current Value

The current value is the value of the variable as the session progresses. When a session starts, the current value of a variable is the same as the start value. As the session progresses, the Integration Service calculates the current value using a variable function that you set for the variable. Unlike the start value of a mapping variable, the current value can change as the Integration Service evaluates the current value of a variable as each row passes through the mapping. The final current value for a variable is saved to the repository at the end of a successful session. When a session fails to complete, the Integration Service does not update the value of the

variable in the repository. The Integration Service states the value saved to the repository for each mapping variable in the session log.

Note: If a variable function is not used to calculate the current value of a mapping variable, the start value of the variable is saved to the repository.

Variable Datatype and Aggregation Type

When you declare a mapping variable in a mapping, you need to configure the datatype and aggregation type for the variable.

The datatype you choose for a mapping variable allows the Integration Service to pick an appropriate default value for the mapping variable. The default is used as the start value of a mapping variable when there is no value defined for a variable in the parameter file, in the repository, and there is no user defined initial value.

The Integration Service uses the aggregate type of a mapping variable to determine the final current value of the mapping variable. When you have a pipeline with multiple partitions, the Integration Service combines the variable value from each partition and saves the final current variable value into the repository.

You can create a variable with the following aggregation types:

φ Count

φ Max

φ Min

You can configure a mapping variable for a Count aggregation type when it is an Integer or Small Integer. You can configure mapping variables of any datatype for Max or Min aggregation types.

To keep the variable value consistent throughout the session run, the Designer limits the variable functions you use with a variable based on aggregation type. For example, use the SetMaxVariable function for a variable with a Max aggregation type, but not with a variable with a Min aggregation type.

Table 7-3 describes the available variable functions and the aggregation types and datatypes you use with each function:

Table 7-3. Variable Functions

Variable Function	Valid Aggregation Types	Valid Datatype
SetVariable	Max or Min	All transformation datatypes except binary datatype.
SetMaxVariable	Max only	All transformation datatypes except binary datatype.
SetMinVariable	Min only	All transformation datatypes except binary datatype.
SetCountVariable	Count only	Integer and small integer datatypes only.

Variable Functions

Variable functions determine how the Integration Service calculates the current value of a mapping variable in a pipeline. Use variable functions in an expression to set the value of a mapping variable for the next session run. The transformation language provides the following variable functions to use in a mapping:

- φ **SetMaxVariable.** Sets the variable to the maximum value of a group of values. It ignores rows marked for update, delete, or reject. To use the SetMaxVariable with a mapping variable, the aggregation type of the mapping variable must be set to Max.
- φ **SetMinVariable.** Sets the variable to the minimum value of a group of values. It ignores rows marked for update, delete, or reject. To use the SetMinVariable with a mapping variable, the aggregation type of the mapping variable must be set to Min.
- φ **SetCountVariable.** Increments the variable value by one. In other words, it adds one to the variable value when a row is marked for insertion, and subtracts one when the row is marked for deletion. It ignores rows

marked for update or reject. To use the SetCountVariable with a mapping variable, the aggregation type of the mapping variable must be set to Count.

- ϕ **SetVariable.** Sets the variable to the configured value. At the end of a session, it compares the final current value of the variable to the start value of the variable. Based on the aggregate type of the variable, it saves a final value to the repository. To use the SetVariable function with a mapping variable, the aggregation type of the mapping variable must be set to Max or Min. The SetVariable function ignores rows marked for delete or reject.

Use variable functions only once for each mapping variable in a pipeline. The Integration Service processes variable functions as it encounters them in the mapping. The order in which the Integration Service encounters variable functions in the mapping may not be the same for every session run. This may cause inconsistent results when you use the same variable function multiple times in a mapping.

The Integration Service does not save the final current value of a mapping variable to the repository when any of the following conditions are true:

- ϕ The session fails to complete.
- ϕ The session is configured for a test load.
- ϕ The session is a debug session.
- ϕ The session runs in debug mode and is configured to discard session output.

Mapping Variables in Mapplets

When you declare a mapping variable for a mapplet and use the mapplet multiple times within the same mapping, the same mapping variable value is shared across all mapplet instances.

Using Mapping Variables

To use mapping variables, complete the following steps:

1. Create a mapping variable.
2. Use the variable and set the variable value.
3. Override or clear the variable value.

Step 1. Create a Mapping Variable

You can create a mapping variable for any mapping or mapplet. You can create as many variables as you need. Once created, use the variable in the mapping or mapplet.

To create a mapping variable:

1. In the Mapping Designer, click Mappings > Parameters and Variables.
-or-
In the Mapplet Designer, click Mapplet > Parameters and Variables.
2. Click the Add button.
3. Specify the variable information.

CHAPTER 22

Workflow Manager

This chapter includes the following topics:

- φ Overview
- φ Workflow Manager Options
- φ Navigating the Workspace
- φ Working with Repository Objects

Overview

In the Workflow Manager, you define a set of instructions called a workflow to execute mappings you build in the Designer. Generally, a workflow contains a session and any other task you may want to perform when you run a session. Tasks can include a session, email notification, or scheduling information. You connect each task with links in the workflow.

You can also create a worklet in the Workflow Manager. A worklet is an object that groups a set of tasks. A worklet is similar to a workflow, but without scheduling information. You can run a batch of worklets inside a workflow.

After you create a workflow, you run the workflow in the Workflow Manager and monitor it in the Workflow Monitor.

Workflow Manager Options

You can customize the Workflow Manager default options to control the behavior and look of the Workflow Manager tools. You can also configure options, such as grouping sessions or docking and undocking windows.

RELATED TOPICS:

- φ “Workflow Manager Options”

Workflow Manager Tools

To create a workflow, you first create tasks such as a session, which contains the mapping you build in the Designer. You then connect tasks with conditional links to specify the order of execution for the tasks you created. The Workflow Manager consists of three tools to help you develop a workflow:

- φ **Task Developer.** Use the Task Developer to create tasks you want to run in the workflow.
- φ **Workflow Designer.** Use the Workflow Designer to create a workflow by connecting tasks with links. You can also create tasks in the Workflow Designer as you develop the workflow.
- φ **Worklet Designer.** Use the Worklet Designer to create a worklet.

The following figure shows what a workflow might look like if you want to run a session, perform a shell command after the session completes, and then stop the workflow:



Workflow Tasks

You can create the following types of tasks in the Workflow Manager:

- φ **Assignment.** Assigns a value to a workflow variable.
- φ **Command.** Specifies a shell command to run during the workflow.
- φ **Control.** Stops or aborts the workflow.
- φ **Decision.** Specifies a condition to evaluate.
- φ **Email.** Sends email during the workflow.
- φ **Event-Raise.** Notifies the Event-Wait task that an event has occurred.
- φ **Event-Wait.** Waits for an event to occur before executing the next task.
- φ **Session.** Runs a mapping you create in the Designer.
- φ **Timer.** Waits for a timed event to trigger.

Workflow Manager Windows

The Workflow Manager displays the following windows to help you create and organize workflows:

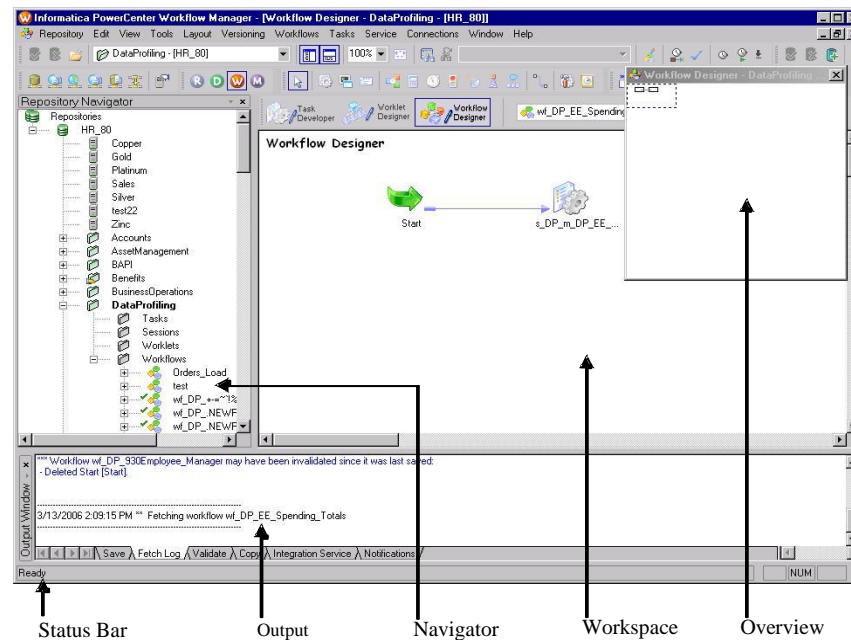
- φ **Navigator.** You can connect to and work in multiple repositories and folders. In the Navigator, the Workflow Manager displays a red icon over invalid objects.
- φ **Workspace.** You can create, edit, and view tasks, workflows, and worklets.
- φ **Output.** Contains tabs to display different types of output messages. The Output window contains the following tabs:
 - **Save.** Displays messages when you save a workflow, worklet, or task. The Save tab displays a validation summary when you save a workflow or a worklet.
 - **Fetch Log.** Displays messages when the Workflow Manager fetches objects from the repository.
 - **Validate.** Displays messages when you validate a workflow, worklet, or task.

- ❑ **Copy.** Displays messages when you copy repository objects.
- ❑ **Server.** Displays messages from the Integration Service.
- ❑ **Notifications.** Displays messages from the Repository Service.
- ❖ **Overview.** An optional window that lets you easily view large workflows in the workspace. Outlines the visible area in the workspace and highlights selected objects in color. Click View > Overview Window to display this window.

You can view a list of open windows and switch from one window to another in the Workflow Manager. To view the list of open windows, click Window > Windows.

The Workflow Manager also displays a status bar that shows the status of the operation you perform.

The following figure shows the Workflow Manager windows:



Setting the Date/Time Display Format

The Workflow Manager displays the date and time formats configured in the Windows Control Panel of the PowerCenter Client machine. To modify the date and time formats, display the Control Panel and open Regional Settings. Set the date and time formats on the Date and Time tabs.

Note: For the Timer task and schedule settings, the Workflow Manager displays date in short date format and the time in 24-hour format (HH:mm).

CHAPTER 23

Workflows and Worklets

This chapter includes the following topics:

- φ Overview
- φ Creating a Workflow
- φ Working with Worklets
- φ Working with Links

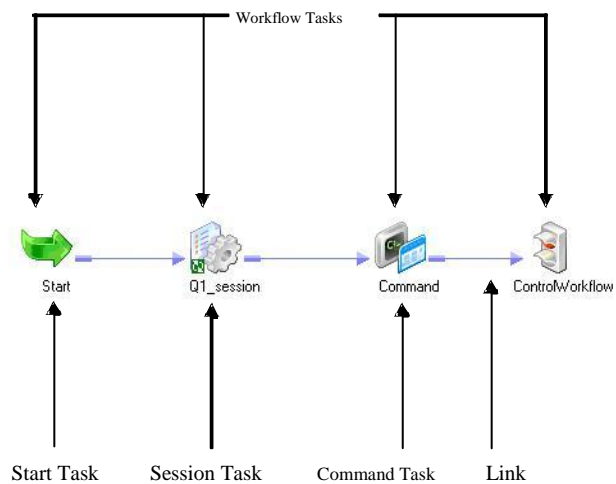
Overview

A workflow is a set of instructions that tells the Integration Service how to run tasks such as sessions, email notifications, and shell commands. After you create tasks in the Task Developer and Workflow Designer, you connect the tasks with links to create a workflow.

In the Workflow Designer, you can specify conditional links and use workflow variables to create branches in the workflow. The Workflow Manager also provides Event-Wait and Event-Raise tasks to control the sequence of task execution in the workflow. You can also create worklets and nest them inside the workflow.

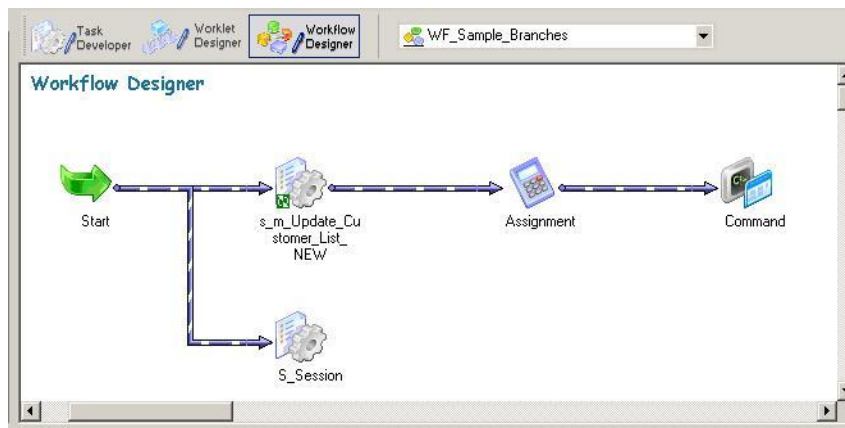
Every workflow contains a Start task, which represents the beginning of the workflow.

The following figure shows a sample workflow:



You can create workflows with branches to run tasks concurrently.

The following figure shows a sample workflow with two branches:



When you create a workflow, select an Integration Service to run the workflow. You can start the workflow using the Workflow Manager, Workflow Monitor, or *pmcmd*.

Use the Workflow Monitor to see the progress of a workflow during its run. The Workflow Monitor can also show the history of a workflow.

Use the following guidelines when you develop a workflow:

1. **Create a workflow.** Create a workflow in the Workflow Designer or by using the Workflow Generation Wizard in the PowerCenter Designer.
2. **Add tasks to the workflow.** You might have already created tasks in the Task Developer. Or, you can add tasks to the workflow as you develop the workflow in the Workflow Designer.
3. **Connect tasks with links.** After you add tasks to the workflow, connect them with links to specify the order of execution in the workflow.
4. **Specify conditions for each link.** You can specify conditions on the links to create branches and dependencies.
5. **Validate workflow.** Validate the workflow in the Workflow Designer to identify errors.

6. **Save workflow.** When you save the workflow, the Workflow Manager validates the workflow and updates the repository.
7. **Run workflow.** In the workflow properties, select an Integration Service to run the workflow. Run the workflow from the Workflow Manager, Workflow Monitor, or *pmcmd*. You can monitor the workflow in the Workflow Monitor.

RELATED TOPICS:

- φ “Manually Starting a Workflow”
- φ “Workflow Monitor”
- φ “Workflow Properties Reference”

Creating a Workflow

A workflow must contain a Start task. The Start task represents the beginning of a workflow. When you create a workflow, the Workflow Designer creates a Start task and adds it to the workflow. You cannot delete the Start task.

After you create a workflow, you can add tasks to the workflow. The Workflow Manager includes tasks such as the Session, Command, and Email tasks.

Finally, you connect workflow tasks with links to specify the order of execution in the workflow. You can add conditions to links.

When you edit a workflow, the Repository Service updates the workflow information when you save the workflow. If a workflow is running when you make edits, the Integration Service uses the updated information the next time you run the workflow.

You can also create a workflow through the Workflow Wizard in the Workflow Manager or the Workflow Generation Wizard in the PowerCenter Designer.

Creating a Workflow Manually

Use the following procedure to create a workflow manually.

To create a workflow manually:

1. Open the Workflow Designer.
2. Click Workflows > Create.
3. Enter a name for the new workflow.
4. Click OK.

The Workflow Designer creates a Start task in the workflow.

Creating a Workflow Automatically

Use the following procedure to create a workflow automatically.

To create a workflow automatically:

1. Open the Workflow Designer. Close any open workflow.
2. Click the session button on the Tasks toolbar.
3. Click in the Workflow Designer workspace.

The Mappings dialog box appears.

4. Select a mapping to associate with the session and click OK.

The Create Workflow dialog box appears. The Workflow Designer names the workflow `wf_SessionName` by default. You can rename the workflow or change other workflow properties.

5. Click OK.

The Workflow Designer creates a workflow for the session.

Adding Tasks to Workflows

After you create a workflow, you add tasks you want to run in the workflow. You may already have created tasks in the Task Developer. Or, you may want to create tasks in the Workflow Designer as you develop the workflow.

If you have already created tasks in the Task Developer, add them to the workflow by dragging the tasks from the Navigator to the Workflow Designer workspace.

To create and add tasks as you develop the workflow, click **Tasks > Create** in the Workflow Designer. Or, use the Tasks toolbar to create and add tasks to the workflow. Click the button on the Tasks toolbar for the task you want to create. Click again in the Workflow Designer workspace to create and add the task.

Tasks you create in the Workflow Designer are non-reusable. Tasks you create in the Task Developer are reusable.

RELATED TOPICS:

- φ “Reusable Workflow Tasks”

Deleting a Workflow

You may decide to delete a workflow that you no longer use. When you delete a workflow, you delete all non-reusable tasks and reusable task instances associated with the workflow. Reusable tasks used in the workflow remain in the folder when you delete the workflow.

If you delete a workflow that is running, the Integration Service aborts the workflow. If you delete a workflow that is scheduled to run, the Integration Service removes the workflow from the schedule.

You can delete a workflow in the Navigator window, or you can delete the workflow currently displayed in the Workflow Designer workspace:

- φ To delete a workflow from the Navigator window, open the folder, select the workflow and press the Delete key.
- φ To delete a workflow currently displayed in the Workflow Designer workspace, click **Workflows > Delete**.

Using the Workflow Wizard

Use the Workflow Wizard to automate the process of creating sessions, adding sessions to a workflow, and linking sessions to create a workflow. The Workflow Wizard creates sessions from mappings and adds them to the workflow. It also creates a Start task and lets you schedule the workflow. You can add tasks and edit other workflow properties after the Workflow Wizard completes. If you want to create concurrent sessions, use the Workflow Designer to manually build a workflow.

Before you create a workflow, verify that the folder contains a valid mapping for the Session task.

Complete the following steps to build a workflow using the Workflow Wizard:

1. Assign a name and Integration Service to the workflow.
2. Create a session.
3. Schedule the workflow.

You can also use the Workflow Generation Wizard in the PowerCenter Designer to generate sessions and workflows.

Step 1. Assign a Name and Integration Service to the Workflow

In the first step of the Workflow Wizard, you add the name and description of the workflow and choose the Integration Service to run the workflow.

To create the workflow:

1. In the Workflow Manager, open the folder containing the mapping you want to use in the workflow.
2. Open the Workflow Designer.
3. Click Workflows > Wizard.

The Workflow Wizard appears.

4. Enter a name for the workflow.

The convention for naming workflows is *wf_WorkflowName*. For more information about naming conventions for repository objects, see *PowerCenter Getting Started*.

5. Enter a description for the workflow.
6. Select the Integration Service to run the workflow and click Next.

Step 2. Create a Session

In the second step of the Workflow Wizard, you create a session based on a mapping. You can add tasks later in the Workflow Designer workspace.

To create a session:

1. In the second step of the Workflow Wizard, select a valid mapping and click the right arrow button.

The Workflow Wizard creates a Session task in the right pane using the selected mapping and names it *s_MappingName* by default.

The following figure shows a mapping selected for a session:

2. You can select additional mappings to create more Session tasks in the workflow.

When you add multiple mappings to the list, the Workflow Wizard creates sequential sessions in the order you add them.

3. Use the arrow buttons to change the session order.

4. Specify whether the session should be reusable.

When you create a reusable session, use the session in other workflows.

5. Specify how you want the Integration Service to run the workflow.

You can specify that the Integration Service runs sessions only if previous sessions complete, or you can specify that the Integration Service always runs each session. When you select this option, it applies to all sessions you create using the Workflow Wizard.

Step 3. Schedule a Workflow

In the third step of the Workflow Wizard, you can schedule a workflow to run continuously, repeat at a given time or interval, or start manually. The Integration Service runs a workflow unless the prior workflow run fails.

When a workflow fails, the Integration Service removes the workflow from the schedule, and you must reschedule it. You can do this in the Workflow Manager or using *pmcmd*.

To schedule a workflow:

1. In the third step of the Workflow Wizard, configure the scheduling and run options.
2. Click Next.

The Workflow Wizard displays the settings for the workflow.

3. Verify the workflow settings and click Finish. To edit settings, click Back.

The completed workflow opens in the Workflow Designer workspace. From the workspace, you can add tasks, create concurrent sessions, add conditions to links, or modify properties.

Working with Worklets

A worklet is an object representing a set of tasks created to reuse a set of workflow logic in multiple workflows. You can create a worklet in the Worklet Designer.

To run a worklet, include the worklet in a workflow. The workflow that contains the worklet is called the parent workflow. When the Integration Service runs a worklet, it expands the worklet to run tasks and evaluate links within the worklet. It writes information about worklet execution in the workflow log.

Suspending Worklets

When you choose Suspend on Error for the parent workflow, the Integration Service also suspends the worklet if a task in the worklet fails. When a task in the worklet fails, the Integration Service stops executing the failed task and other tasks in its path. If no other task is running in the worklet, the worklet status is “Suspended.” If one or more tasks are still running in the worklet, the worklet status is “Suspending.” The Integration Service suspends the parent workflow when the status of the worklet is “Suspended” or “Suspending.”

For more information about suspending workflows, see the *PowerCenter Advanced Workflow Guide*.

Developing a Worklet

To develop a worklet, you must first create a worklet. After you create a worklet, configure worklet properties and add tasks to the worklet. You can create reusable worklets in the Worklet Designer. You can also create non-reusable worklets in the Workflow Designer as you develop the workflow.

Creating a Reusable Worklet

Create reusable worklets in the Worklet Designer. You can view a list of reusable worklets in the Navigator Worklets node.

To create a reusable worklet:

1. In the Worklet Designer, click Worklet > Create.

The Create Worklet dialog box appears.

2. Enter a name for the worklet.
3. If you are adding the worklet to a workflow that is enabled for concurrent execution, enable the worklet for concurrent execution.
4. Click OK.

The Worklet Designer creates a Start task in the worklet.

Creating a Non-Reusable Worklet

You can create a non-reusable worklet in the Workflow Designer as you develop the workflow. Non-reusable worklets only exist in the workflow. You cannot use a non-reusable worklet in another workflow. After you create the worklet in the Workflow Designer, open the worklet to edit it in the Worklet Designer.

You can promote non-reusable worklets to reusable worklets by selecting the Make Reusable option in the worklet properties. To rename a non-reusable worklet, open the worklet properties in the Workflow Designer.

To create a non-reusable worklet:

1. In the Workflow Designer, open a workflow.
2. Click Tasks > Create.
3. For the Task type, select Worklet.
4. Enter a name for the task.
5. Click Create.

The Workflow Designer creates the worklet and adds it to the workspace.

6. Click Done.

Configuring Worklet Properties

When you use a worklet in a workflow, you can configure the same set of general task settings on the General tab as any other task. For example, you can make a worklet reusable, disable a worklet, configure the input link to the worklet, or fail the parent workflow based on the worklet.

In addition to general task settings, you can configure the following worklet properties:

- φ **Worklet variables.** Use worklet variables to reference values and record information. You use worklet variables the same way you use workflow variables. You can assign a workflow variable to a worklet variable to override its initial value. For more information about worklet variables, see the *PowerCenter Advanced Workflow Guide*.
- φ **Events.** To use the Event-Wait and Event-Raise tasks in the worklet, you must first declare an event in the worklet properties.
- φ **Metadata extension.** Extend the metadata stored in the repository by associating information with repository objects.

Adding Tasks in Worklets

After you create a worklet, add tasks by opening the worklet in the Worklet Designer. A worklet must contain a Start task. The Start task represents the beginning of a worklet. When you create a worklet, the Worklet Designer creates a Start task for you.

To add tasks to a non-reusable worklet:

1. Create a non-reusable worklet in the Workflow Designer workspace.
2. Right-click the worklet and choose Open Worklet.

The Worklet Designer opens so you can add tasks in the worklet.

3. Add tasks in the worklet by using the Tasks toolbar or click Tasks > Create in the Worklet Designer.
4. Connect tasks with links.

Declaring Events in Worklets

Use Event-Wait and Event-Raise tasks in a worklet like you would use workflows. To use the Event-Raise task, you first declare a user-defined event in the worklet. Events in one instance of a worklet do not affect events in other instances of the worklet. You cannot specify worklet events in the Event tasks in the parent workflow.

Nesting Worklets

You can nest a worklet within another worklet. When you run a workflow containing nested worklets, the Integration Service runs the nested worklet from within the parent worklet. You can group several worklets together by function or simplify the design of a complex workflow when you nest worklets.

You might choose to nest worklets to load data to fact and dimension tables. Create a nested worklet to load fact and dimension data into a staging area. Then, create a nested worklet to load the fact and dimension data from the staging area to the data warehouse.

You might choose to nest worklets to simplify the design of a complex workflow. Nest worklets that can be grouped together within one worklet. To nest an existing reusable worklet, click Tasks > Insert Worklet. To create a non-reusable nested worklet, click Tasks > Create, and select worklet.

Working with Links

Use links to connect each task in a workflow or worklet. You can specify conditions with links to create branches. The Workflow Manager does not allow you to use links to create loops. Each link in the workflow or worklet can run only once.

To link two tasks:

1. In the Tasks toolbar, click the Link Tasks button.

Link Tasks
Button

2. In the workspace, click the first task you want to connect and drag it to the second task.
3. A link appears between the two tasks.

If you want to link multiple tasks concurrently, you may not want to connect each link manually.

To link tasks concurrently:

1. In the workspace, click the first task you want to connect.
2. Ctrl-click all other tasks you want to connect.
Note: Do not use Ctrl+A or Edit > Select All to choose tasks.
3. Click Tasks > Link Concurrent.

A link appears between the first task you selected and each task you added. The first task you selected links to each task concurrently.

If you have a number of tasks that you want to link sequentially, you may not wish to connect each link manually.

To link tasks sequentially:

1. In the workspace, click the first task you want to connect.
2. Ctrl-click the next task you want to connect. Continue to add tasks in the order you want them to run.
3. Click Tasks > Link Sequential.

Links appear in sequential order between the first task and each subsequent task you added.

Specifying Link Conditions

After you create links between tasks, you can specify conditions for each link to determine the order of execution in the workflow. If you do not specify conditions for each link, the Integration Service runs the next task in the workflow or worklet by default.

Use predefined or user-defined workflow and worklet variables in the link condition. If the link condition evaluates to True, the Integration Service runs the next task in the workflow or worklet. If the link condition

evaluates to False, the Integration Service does not run the next task.

You can view results of link evaluation during workflow runs in the workflow log file.

Example of Link Conditions

Use link conditions to specify the order of execution or to create branches. For example, you have two Session tasks in the workflow, s_STORES_CA and s_STORES_AZ. You want the Integration Service to run the second Session task only if the first Session task has no target failed rows.

To accomplish this, you can set the following link condition between the sessions so that the s_STORES_AZ runs only if the number of failed target rows for S_STORES_CA is zero:

```
$s_STORES_CA.TgtFailedRows = 0
```

After you specify the link condition in the Expression Editor, the Workflow Manager validates the link condition and displays it next to the link in the workflow or worklet.

To specify a condition for a link:

1. In the Workflow Designer or Worklet Designer workspace, double-click the link you want to specify.

-or-

Right-click the link and choose Edit. The Expression Editor appears.

2. In the Expression Editor, enter the link condition.

The Expression Editor provides predefined workflow and worklet variables, user-defined workflow and worklet variables, variable functions, and boolean and arithmetic operators.

3. Validate the expression using the Validate button.

The Workflow Manager displays validation results in the Output window.

Tip: Drag the end point of a link to move it from one task to another without losing the link condition.

Viewing Links in a Workflow or Worklet

When you edit a workflow or worklet, you can view the forward or backward link paths to other tasks. You can highlight paths to see links in the workflow branch from the Start task to the last task in the branch.

Note: You can configure the color the Workflow Manager uses to display links. To configure the color for links, click Tools > Options > Format, and choose the Link Selection option.

To view link paths:

1. In the Workflow Designer or Worklet Designer workspace, right-click a task and choose Highlight Path.
2. Select Forward Path, Backward Path, or Both.

The Workflow Manager highlights all links in the branch you select.

CHAPTER 24

Sessions

This chapter includes the following topics:

- φ Overview
- φ Creating a Session Task
- φ Editing a Session

Overview

A session is a set of instructions that tells the Integration Service how and when to move data from sources to targets. A session is a type of task, similar to other tasks available in the Workflow Manager. In the Workflow Manager, you configure a session by creating a Session task. To run a session, you must first create a workflow to contain the Session task.

When you create a Session task, enter general information such as the session name, session schedule, and the Integration Service to run the session. You can select options to run pre-session shell commands, send On-Success or On-Failure email, and use FTP to transfer source and target files.

Configure the session to override parameters established in the mapping, such as source and target location, source and target type, error tracing levels, and transformation attributes. You can also configure the session to collect performance details for the session and store them in the PowerCenter repository. You might view performance details for a session to tune the session.

You can run as many sessions in a workflow as you need. You can run the Session tasks sequentially or concurrently, depending on the requirement.

The Integration Service creates several files and in-memory caches depending on the transformations and options used in the session.

Creating a Session Task

You create a Session task for each mapping that you want the Integration Service to run. The Integration Service uses the instructions configured in the session to move data from sources to targets.

You can create a reusable Session task in the Task Developer. You can also create non-reusable Session tasks in the Workflow Designer as you develop the workflow. After you create the session, you can edit the session properties at any time.

Note: Before you create a Session task, you must configure the Workflow Manager to communicate with databases and the Integration Service. You must assign appropriate permissions for any database, FTP, or external loader connections you configure.

Steps to Create a Session Task

Create the Session task in the Task Developer or the Workflow Designer. Session tasks created in the Task Developer are reusable.

To create a Session task:

1. In the Workflow Designer, click Tasks > Create.
2. Select Session Task for the task type.
3. Enter a name for the Session task.
4. Click Create.
5. Select the mapping you want to use in the Session task and click OK.

6. Click Done.

Editing a Session

After you create a session, you can edit it. For example, you might need to adjust the buffer and cache sizes, modify the update strategy, or clear a variable value saved in the repository.

Double-click the Session task to open the session properties. The session has the following tabs, and each of those tabs has multiple settings:

- φ **General tab.** Enter session name, mapping name, and description for the Session task, assign resources, and configure additional task options.
- φ **Properties tab.** Enter session log information, test load settings, and performance configuration.
- φ **Config Object tab.** Enter advanced settings, log options, and error handling configuration.
- φ **Mapping tab.** Enter source and target information, override transformation properties, and configure the session for partitioning.
- φ **Components tab.** Configure pre- or post-session shell commands and emails.
- φ **Metadata Extension tab.** Configure metadata extension options.

You can edit session properties at any time. The repository updates the session properties immediately.

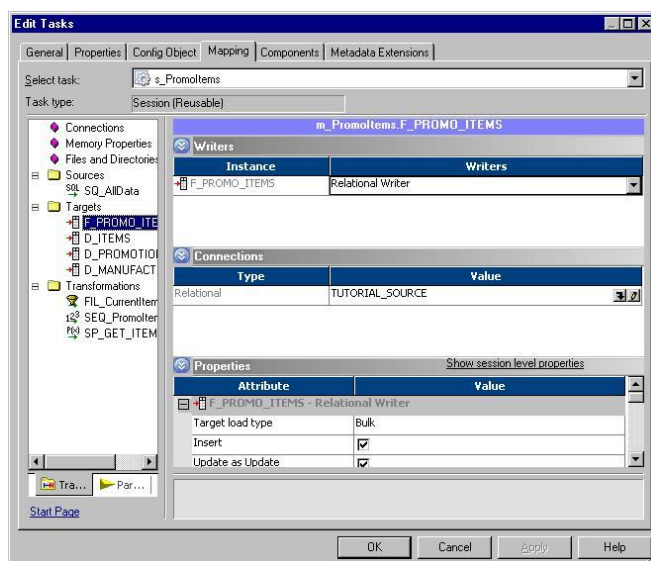
If the session is running when you edit the session, the repository updates the session when the session completes. If the mapping changes, the Workflow Manager might issue a warning that the session is invalid. The Workflow Manager then lets you continue editing the session properties. After you edit the session properties, the Integration Service validates the session and reschedules the session.

Applying Attributes to All Instances

When you edit the session properties, you can apply source, target, and transformation settings to all instances of the same type in the session. You can also apply settings to all partitions in a pipeline. You can apply reader or writer settings, connection settings, and properties settings.

For example, you might need to change a relational connection from a test to a production database for all the target instances in a session. You can change the connection value for one target in a session and apply the connection to the other relational target objects.

The following figure shows the writers, connections, and properties settings for a target instance in a session:



CHAPTER 25

Tasks

This chapter includes the following topics:

- φ Overview
- φ Creating a Task
- φ Configuring Tasks
- φ Working with the Assignment Task
- φ Working with the Command Task
- φ Working with the Control Task
- φ Working with the Decision Task
- φ Working with the Event Task
- φ Working with the Timer Task

Overview

The Workflow Manager contains many types of tasks to help you build workflows and worklets. You can create reusable tasks in the Task Developer. Or, create and add tasks in the Workflow or Worklet Designer as you develop the workflow.

The following table summarizes workflow tasks available in Workflow Manager:

Task Name	Tool	Reusable	Description
Assignment	Workflow Designer Worklet Designer	No	Assigns a value to a workflow variable.
Command	Task Developer Workflow Designer Worklet Designer	Yes	Specifies shell commands to run during the workflow. You can choose to run the Command task if the previous task in the workflow completes Stops or aborts the workflow.
Control	Workflow Designer Worklet Designer	No	Specifies a condition to evaluate in the workflow. Use the Decision task to create branches in a workflow.
Decision	Workflow Designer Worklet Designer	No	

Task Name	Tool	Reusable	Description
Email	Task Developer Workflow Designer Worklet Designer	Yes	Sends email during the workflow.
Event-Raise	Workflow Designer Worklet Designer	No	Represents the location of a user-defined event. The Event-Raise task triggers the user-defined event when the Integration Service runs the Event-Raise task
Event-Wait	Workflow Designer Worklet Designer	No	Waits for a user-defined or a predefined event to occur. Once the event occurs, the Integration Service
Session	Task Developer Workflow Designer Worklet Designer	Yes	completes the rest of the workflow. Set of instructions to run a mapping. Waits for a specified period of time to run the next task.
Timer	Workflow Designer Worklet Designer	No	

The Workflow Manager validates tasks attributes and links. If a task is invalid, the workflow becomes invalid. Workflows containing invalid sessions may still be valid.

Creating a Task

You can create tasks in the Task Developer, or you can create them in the Workflow Designer or the Worklet Designer as you develop the workflow or worklet. Tasks you create in the Task Developer are reusable. Tasks you create in the Workflow Designer and Worklet Designer are non-reusable by default.

RELATED TOPICS:

φ “Reusable Workflow Tasks”

Creating a Task in the Task Developer

You can create the following types of task in the Task Developer:

- φ Command
- φ Session
- φ Email

To create a task in the Task Developer:

1. In the Task Developer, click Tasks > Create.
2. Select the task type you want to create, Command, Session, or Email.
3. Enter a name for the task.
4. For session tasks, select the mapping you want to associate with the session.
5. Click Create.

The Task Developer creates the workflow task.

6. Click Done to close the Create Task dialog box.

Creating a Task in the Workflow or Worklet Designer

You can create and add tasks in the Workflow Designer or Worklet Designer as you develop the workflow or worklet. You can create any type of task in the Workflow Designer or Worklet Designer. Tasks you create in the Workflow Designer or Worklet Designer are non-reusable. Edit the General tab of the task properties to promote a non-reusable task to a reusable task.

To create tasks in the Workflow Designer or Worklet Designer:

1. In the Workflow Designer or Worklet Designer, open a workflow or worklet.
2. Click Tasks > Create.
3. Select the type of task you want to create.
4. Enter a name for the task.
5. Click Create.

The Workflow Designer or Worklet Designer creates the task and adds it to the workspace.

6. Click Done.

You can also use the Tasks toolbar to create and add tasks to the workflow. Click the button on the Tasks toolbar for the task you want to create. Click again in the Workflow Designer or Worklet Designer workspace to create and add the task. The Workflow Designer or Worklet Designer creates the task with a default task name when you use the Tasks toolbar.

Configuring Tasks

After you create the task, you can configure general task options on the General tab. For each task instance in the workflow, you can configure how the Integration Service runs the task and the other objects associated with the selected task. You can also disable the task so you can run rest of the workflow without the selected task.

When you use a task in the workflow, you can edit the task in the Workflow Designer and configure the following task options in the General tab:

- ⌘ **Fail parent if this task fails.** Choose to fail the workflow or worklet containing the task if the task fails.
- ⌘ **Fail parent if this task does not run.** Choose to fail the workflow or worklet containing the task if the task does not run.
- ⌘ **Disable this task.** Choose to disable the task so you can run the rest of the workflow without the task.
- ⌘ **Treat input link as AND or OR.** Choose to have the Integration Service run the task when all or one of the input link conditions evaluates to True.

Reusable Workflow Tasks

Workflows can contain reusable task instances and non-reusable tasks. Non-reusable tasks exist within a single workflow. Reusable tasks can be used in multiple workflows in the same folder.

You can create any task as non-reusable or reusable. Tasks you create in the Task Developer are reusable. Tasks you create in the Workflow Designer are non-reusable by default. However, you can edit the general properties of a task to promote it to a reusable task.

The Workflow Manager stores each reusable task separate from the workflows that use the task. You can view a list of reusable tasks in the Tasks node in the Navigator window. You can see a list of all reusable Session tasks in the Sessions node in the Navigator window.

Promoting a Non-Reusable Workflow Task

You can promote a non-reusable workflow task to a reusable task. Reusable tasks must have unique names within the repository. When you promote a non-reusable task, the repository checks for naming conflicts. If a reusable task with the same name already exists, the repository appends a number to the reusable task name to make it unique. The repository applies the appended name to the checked-out version and to the latest checked-in version of the reusable task.

To promote a non-reusable workflow task:

1. In the Workflow Designer, double-click the task you want to make reusable.
2. In the General tab of the Edit Task dialog box, select the Make Reusable option.
3. When prompted whether you are sure you want to promote the task, click Yes.
4. Click OK.

The newly promoted task appears in the list of reusable tasks in the Tasks node in the Navigator window.

Instances and Inherited Changes

When you add a reusable task to a workflow, you add an instance of the task. The definition of the task exists outside the workflow, while an instance of the task exists in the workflow.

You can edit the task instance in the Workflow Designer. Changes you make in the task instance exist only in the workflow. The task definition remains unchanged in the Task Developer.

When you make changes to a reusable task definition in the Task Developer, the changes reflect in the instance of the task in the workflow if you have not edited the instance.

Reverting Changes in Reusable Tasks Instances

When you edit an instance of a reusable task in the workflow, you can revert back to the settings in the task definition. When you change settings in the task instance, the Revert button appears. The Revert button appears after you override task properties. You cannot use the Revert button for settings that are read-only or locked by another user.

AND or OR Input Links

For each task, you can choose to treat the input link as an AND link or an OR link. When a task has one input link, the Integration Service processes the task when the previous object completes and the link condition evaluates to True. If you have multiple links going into one task, you can choose to have an AND input link so that the Integration Service runs the task when all the link conditions evaluate to True. Or, you can choose to have an OR input link so that the Integration Service runs the task as soon as any link condition evaluates to True.

To set the type of input links, double-click the task to open the Edit Tasks dialog box. Select AND or OR for the input link type.

RELATED TOPICS:

φ “Working with Links”

Disabling Tasks

In the Workflow Designer, you can disable a workflow task so that the Integration Service runs the workflow without the disabled task. The status of a disabled task is DISABLED. Disable a task in the workflow by selecting the Disable This Task option in the Edit Tasks dialog box.

Failing Parent Workflow or Worklet

You can choose to fail the workflow or worklet if a task fails or does not run. The workflow or worklet that contains the task instance is called the parent. A task might not run when the input condition for the task evaluates to False.

To fail the parent workflow or worklet if the task fails, double-click the task and select the Fail Parent If This Task Fails option in the General tab. When you select this option and a task fails, it does not prevent the other tasks in the workflow or worklet from running. Instead, the Integration Service marks the status of the workflow or worklet as failed. If you have a session nested within multiple worklets, you must select the Fail Parent If This Task Fails option for each worklet instance to see the failure at the workflow level.

To fail the parent workflow or worklet if the task does not run, double-click the task and select the Fail Parent If This Task Does Not Run option in the General tab. When you choose this option, the Integration Service fails the parent workflow if a task did not run.

Note: The Integration Service does not fail the parent workflow if you disable a task.

Working with the Assignment Task

You can assign a value to a user-defined workflow variable with the Assignment task. To use an Assignment task in the workflow, first create and add the Assignment task to the workflow. Then configure the Assignment task to assign values or expressions to user-defined variables. After you assign a value to a variable using the Assignment task, the Integration Service uses the assigned value for the variable during the remainder of the workflow.

You must create a variable before you can assign values to it. You cannot assign values to predefined workflow variables.

To create an Assignment task:

1. In the Workflow Designer, click Tasks > Create.
2. Select Assignment Task for the task type.
3. Enter a name for the Assignment task. Click Create. Then click Done.

The Workflow Designer creates and adds the Assignment task to the workflow.

4. Double-click the Assignment task to open the Edit Task dialog box.
5. On the Expressions tab, click Add to add an assignment.
6. Click the Open button in the User Defined Variables field.
7. Select the variable for which you want to assign a value. Click OK.
8. Click the Edit button in the Expression field to open the Expression Editor.

The Expression Editor shows predefined workflow variables, user-defined workflow variables, variable functions, and boolean and arithmetic operators.

9. Enter the value or expression you want to assign.

For example, if you want to assign the value 500 to the user-defined variable \$\$custno1, enter the number 500 in the Expression Editor.

10. Click Validate.

Validate the expression before you close the Expression Editor.

11. Repeat steps 6 to 8 to add more variable assignments.

Use the up and down arrows in the Expressions tab to change the order of the variable assignments.

Working with the Command Task

You can specify one or more shell commands to run during the workflow with the Command task. For example, you can specify shell commands in the Command task to delete reject files, copy a file, or archive target files.

Use a Command task in the following ways:

- φ **Standalone Command task.** Use a Command task anywhere in the workflow or worklet to run shell commands.
- φ **Pre- and post-session shell command.** You can call a Command task as the pre- or post-session shell command for a Session task.

Use any valid UNIX command or shell script for UNIX servers, or any valid DOS or batch file for Windows servers. For example, you might use a shell command to copy a file from one directory to another. For a Windows server you would use the following shell command to copy the SALES_ ADJ file from the source directory, L, to the target, H:

```
copy L:\sales\sales_adj H:\marketing\
```

For a UNIX server, you would use the following command to perform a similar operation:

```
cp sales/sales_adj marketing/
```

Each shell command runs in the same environment as the Integration Service. Environment settings in one shell command script do not carry over to other scripts. To run all shell commands in the same environment, call a single shell script that invokes other scripts.

RELATED TOPICS:

- φ “Using Pre- and Post-Session Shell Commands”

Using Parameters and Variables

You can use parameters and variables in standalone Command tasks and pre- and post-session shell commands.

For example, you might use a service process variable instead of hard-coding a directory name.

You can use the following parameters and variables in commands:

- φ **Standalone Command tasks.** You can use service, service process, workflow, and worklet variables in standalone Command tasks. You cannot use session parameters, mapping parameters, or mapping variables in standalone Command tasks. The Integration Service does not expand these types of parameters and variables in standalone Command tasks.
- φ **Pre- and post-session shell commands.** You can use any parameter or variable type that you can define in the parameter file.

Assigning Resources

You can assign resources to Command task instances in the Worklet or Workflow Designer. You might want to assign resources to a Command task if you assign the workflow to an Integration Service associated with a grid. When you assign a resource to a Command task and the Integration Service is configured to check resources, the Load Balancer dispatches the task to a node that has the resource available. A task fails if the Load Balancer cannot find a node where the required resource is available.

Creating a Command Task

Complete the following steps to create a Command task.

To create a Command task:

1. In the Workflow Designer or the Task Developer, click Task > Create.
2. Select Command Task for the task type.
3. Enter a name for the Command task. Click Create. Then click Done.
4. Double-click the Command task in the workspace to open the Edit Tasks dialog box.
5. In the Commands tab, click the Add button to add a command.
6. In the Name field, enter a name for the new command.
7. In the Command field, click the Edit button to open the Command Editor.
8. Enter the command you want to run. Enter one command in the Command Editor. You can use service, service process, workflow, and worklet variables in the command.
9. Click OK to close the Command Editor.
10. Repeat steps 4 to 9 to add more commands in the task.
11. Optionally, click the General tab in the Edit Tasks dialog to assign resources to the Command task.
12. Click OK.

If you specify non-reusable shell commands for a session, you can promote the non-reusable shell commands to a reusable Command task.

RELATED TOPICS:

φ “Creating a Reusable Command Task from Pre- or Post-Session Commands”

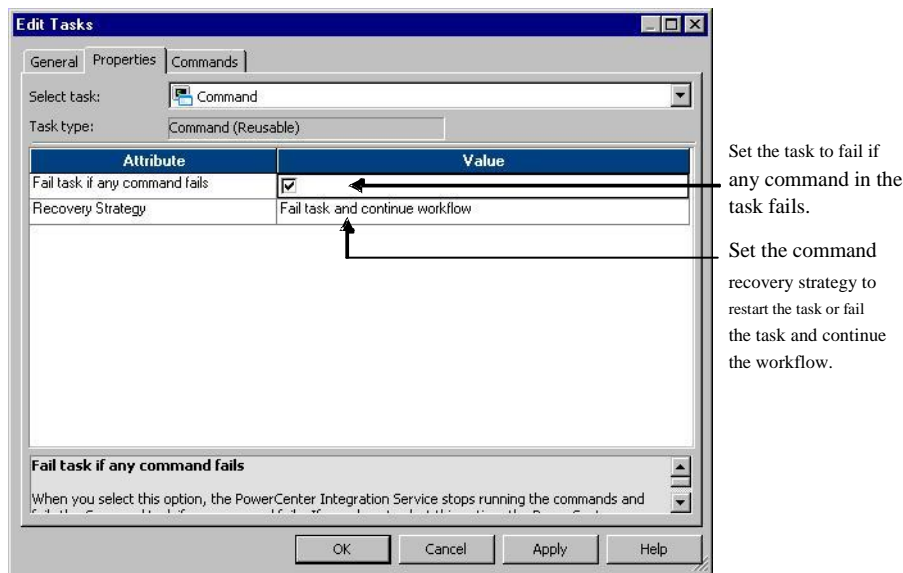
Executing Commands in the Command Task

The Integration Service runs shell commands in the order you specify them. If the Load Balancer has more Command tasks to dispatch than the Integration Service can run at the time, the Load Balancer places the tasks it cannot run in a queue. When the Integration Service becomes available, the Load Balancer dispatches tasks from the queue in the order determined by the workflow service level.

You can choose to run a command only if the previous command completed successfully. Or, you can choose to run all commands in the Command task, regardless of the result of the previous command. If you configure multiple commands in a Command task to run on UNIX, each command runs in a separate shell.

If you choose to run a command only if the previous command completes successfully, the Integration Service stops running the rest of the commands and fails the task when one of the commands in the Command task fails. If you do not choose this option, the Integration Service runs all the commands in the Command task and treats the task as completed, even if a command fails. If you want the Integration Service to perform the next command only if the previous command completes successfully, select Fail Task if Any Command Fails in the Properties tab of the Command task.

The following figure shows the Fail Task if Any Command Fails option:



You can choose a recovery strategy for the task. The recovery strategy determines how the Integration Service recovers the task when you configure workflow recovery and the task fails. You can configure the task to restart or you can configure the task to fail and continue running the workflow. For more information, see the *PowerCenter Advanced Workflow Guide*.

Log Files and Command Tasks

When the Integration Service processes a Command task, it creates temporary files in \$PMTempDir. It writes temporary process files to \$PMTempDir before it writes them to the log files. After it writes the process files to the log files, it deletes them from \$PMTempDir. If the Integration Service shuts down before it deletes the process files, you must delete them manually. The process file names begin with is.process.

Working with the Control Task

Use the Control task to stop, abort, or fail the top-level workflow or the parent workflow based on an input link condition. A parent workflow or worklet is the workflow or worklet that contains the Control task.

To create a Control task:

1. In the Workflow Designer, click Tasks > Create.
2. Select Control Task for the task type.
3. Enter a name for the Control task. Click Create. Then click Done.

The Workflow Manager creates and adds the Control task to the workflow.

4. Double-click the Control task in the workspace to open it.

5. Configure the following control options on the Properties tab:

Control Option	Description
Fail Me	Marks the Control task as “Failed.” The Integration Service fails the Control task if you choose this option. If you choose Fail Me in the Properties tab and choose Fail Parent If This Task Fails in the General tab, the Integration Service fails the parent workflow.
Fail Parent	Marks the status of the workflow or worklet that contains the Control task as failed after the workflow or worklet completes.
Stop Parent	Stops the workflow or worklet that contains the Control task.
Abort Parent	Aborts the workflow or worklet that contains the Control task.
Fail Top-Level Workflow	Fails the workflow that is running.
Stop Top-Level Workflow	Stops the workflow that is running.
Abort Top-Level Workflow	Aborts the workflow that is running.

Working with the Decision Task

You can enter a condition that determines the execution of the workflow, similar to a link condition with the Decision task. The Decision task has a predefined variable called `$Decision_task_name.condition` that represents the result of the decision condition. The Integration Service evaluates the condition in the Decision task and sets the predefined condition variable to True (1) or False (0).

You can specify one decision condition per Decision task. After the Integration Service evaluates the Decision task, use the predefined condition variable in other expressions in the workflow to help you develop the workflow.

Depending on the workflow, you might use link conditions instead of a Decision task. However, the Decision task simplifies the workflow. If you do not specify a condition in the Decision task, the Integration Service evaluates the Decision task to True.

RELATED TOPICS:

φ “Working with Links”

Using the Decision Task

Use the Decision task instead of multiple link conditions in a workflow. Instead of specifying multiple link conditions, use the predefined condition variable in a Decision task to simplify link conditions.

Example

For example, you have a Command task that depends on the status of the three sessions in the workflow. You want the Integration Service to run the Command task when any of the three sessions fails. To accomplish this, use a Decision task with the following decision condition:

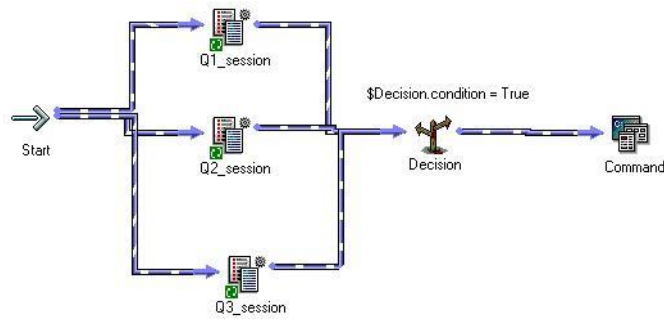
```
$Q1_session.status = FAILED OR $Q2_session.status = FAILED OR $Q3_session.status = FAILED
```

You can then use the predefined condition variable in the input link condition of the Command task. Configure the input link with the following link condition:

```
$Decision.condition = True
```

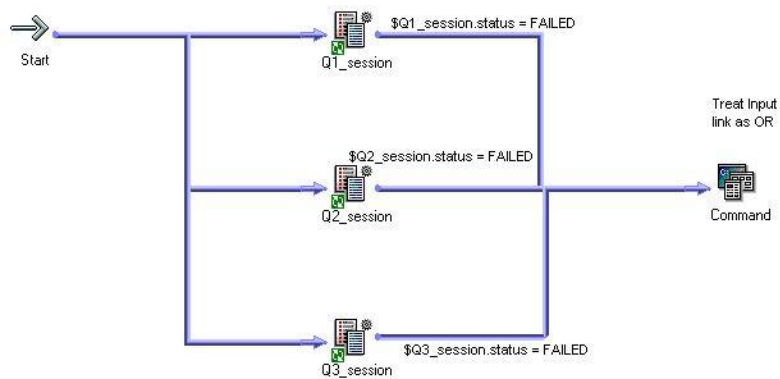
Figure 5-1 shows the sample workflow using a Decision task:

Figure 5-1. Sample Workflow Using a Decision Task



You can configure the same logic in the workflow without the Decision task. Without the Decision task, you need to use three link conditions and treat the input links to the Command task as OR links.

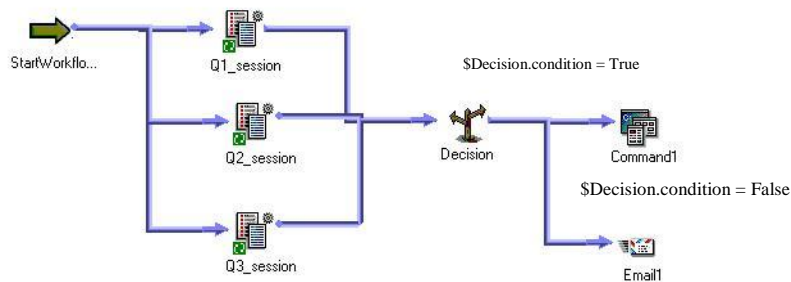
The following figure shows the sample workflow without the Decision task:



You can further expand the sample workflow in Figure 5-1. In Figure 5-1, the Integration Service runs the Command task if any of the three Session tasks fails. Suppose now you want the Integration Service to also run an Email task if all three Session tasks succeed.

To do this, add an Email task and use the decision condition variable in the link condition.

The following figure shows the expanded sample workflow using a Decision task:



Creating a Decision Task

Complete the following steps to create a Decision task.

To create a Decision task:

1. In the Workflow Designer, click Tasks > Create.
2. Select Decision Task for the task type.
3. Enter a name for the Decision task. Click Create. Then click Done.
The Workflow Designer creates and adds the Decision task to the workspace.
4. Double-click the Decision task to open it.
5. Click the Open button in the Value field to open the Expression Editor.
6. In the Expression Editor, enter the condition you want the Integration Service to evaluate.
Validate the expression before you close the Expression Editor.
7. Click OK.

Working with the Event Task

You can define events in the workflow to specify the sequence of task execution. The event is triggered based on the completion of the sequence of tasks. Use the following tasks to help you use events in the workflow:

- φ **Event-Raise task.** Event-Raise task represents a user-defined event. When the Integration Service runs the Event-Raise task, the Event-Raise task triggers the event. Use the Event-Raise task with the Event-Wait task to define events.
- φ **Event-Wait task.** The Event-Wait task waits for an event to occur. Once the event triggers, the Integration Service continues executing the rest of the workflow.

To coordinate the execution of the workflow, you may specify the following types of events for the Event-Wait and Event-Raise tasks:

- φ **Predefined event.** A predefined event is a file-watch event. For predefined events, use an Event-Wait task to instruct the Integration Service to wait for the specified indicator file to appear before continuing with the rest of the workflow. When the Integration Service locates the indicator file, it starts the next task in the workflow.
- φ **User-defined event.** A user-defined event is a sequence of tasks in the workflow. Use an Event-Raise task to specify the location of the user-defined event in the workflow. A user-defined event is sequence of tasks in the branch from the Start task leading to the Event-Raise task.

When all the tasks in the branch from the Start task to the Event-Raise task complete, the Event-Raise task triggers the event. The Event-Wait task waits for the Event-Raise task to trigger the event before continuing with the rest of the tasks in its branch.

Working with Event-Raise Tasks

The Event-Raise task represents the location of a user-defined event. A user-defined event is the sequence of tasks in the branch from the Start task to the Event-Raise task. When the Integration Service runs the Event-Raise task, the Event-Raise task triggers the user-defined event.

To use an Event-Raise task, you must first declare the user-defined event. Then, create an Event-Raise task in the workflow to represent the location of the user-defined event you just declared. In the Event-Raise task properties, specify the name of a user-defined event.

Declaring a User-Defined Event

Complete the following steps to declare a name for a user-defined event.

To declare a user-defined event:

1. In the Workflow Designer, click Workflow > Edit.
2. Select the Events tab in the Edit Workflow dialog box.
3. Click the Add button to add an event name.
Event name is not case sensitive.
4. Click OK.

Using the Event-Raise Task for a User-Defined Event

After you declare a user-defined event, use the Event-Raise task to represent the location of the event and to trigger the event.

To use an Event-Raise task:

1. In the Workflow Designer workspace, create an Event-Raise task and place it in the workflow to represent the user-defined event you want to trigger.
A user-defined event is the sequence of tasks in the branch from the Start task to the Event-Raise task.
2. Double-click the Event-Raise task to open it.
3. On the Properties tab, click the Open button in the Value field to open the Events Browser for user-defined events.
4. Choose an event in the Events Browser.
5. Click OK twice.

Working with Event-Wait Tasks

The Event-Wait task waits for a predefined event or a user-defined event. A predefined event is a file-watch event. When you use the Event-Wait task to wait for a predefined event, you specify an indicator file for the Integration Service to watch. The Integration Service waits for the indicator file to appear. Once the indicator file appears, the Integration Service continues running tasks after the Event-Wait task.

You can assign resources to Event-Wait tasks that wait for predefined events. You may want to assign a resource to a predefined Event-Wait task if you are running on a grid and the indicator file appears on a specific node or in a specific directory. When you assign a resource to a predefined Event-Wait task and the Integration Service is configured to check resources, the Load Balancer distributes the task to a node where the required resource is available. For more information about assigning resources to tasks, see the *PowerCenter Advanced Workflow Guide*. For more information about configuring the Integration Service to check resources, see the *PowerCenter Administrator Guide*.

Note: If you use the Event-Raise task to trigger the event when you wait for a predefined event, you may not be able to successfully recover the workflow.

You can also use the Event-Wait task to wait for a user-defined event. To use the Event-Wait task for a user-

defined event, specify the name of the user-defined event in the Event-Wait task properties. The Integration Service waits for the Event-Raise task to trigger the user-defined event. Once the user-defined event is triggered, the Integration Service continues running tasks after the Event-Wait task.

Waiting for User-Defined Events

Use the Event-Wait task to wait for a user-defined event. A user-defined event is triggered by the Event-Raise task. To wait for a user-defined event, you must first use an Event-Raise task to trigger the user-defined event.

To wait for a user-defined event:

1. In the workflow, create an Event-Wait task and double-click the Event-Wait task to open it.
2. In the Events tab of the task, select User-Defined.
3. Click the Event button to open the Events Browser dialog box.
4. Select a user-defined event for the Integration Service to wait.
5. Click OK twice.

Waiting for Predefined Events

To use a predefined event, you need a shell command, script, or batch file to create an indicator file. The file must be created or sent to a directory that the Integration Service can access. The file can be any format recognized by the Integration Service operating system. You can choose to have the Integration Service delete the indicator file after it detects the file, or you can manually delete the indicator file. The Integration Service marks the status of the Event-Wait task as failed if it cannot delete the indicator file.

When you specify the indicator file in the Event-Wait task, enter the directory in which the file appears and the name of the indicator file. You must provide the absolute path for the file. If you specify the file name and not the directory, the Integration Service looks for the indicator file in the following directory:

- φ On Windows, the Integration Service looks for the file in the system directory. For example, on Windows 2000, the system directory is `c:\winnt\system32`.
- φ On UNIX, the Integration Service looks for the indicator file in the current working directory for the Integration Service process. On UNIX this directory is `/server/bin`.

You can enter the actual name of the file or use process variables to specify the location of the file. You can also use user-defined workflow and worklet variables to specify the file name and location. For example, create a workflow variable, `$$MyFileWatchFile`, for the indicator file name and location, and set `$$MyFileWatchFile` to the file name and location in the parameter file.

The Integration Service writes the time the file appears in the workflow log.

Note: Do not use a source or target file name as the indicator file name because you may accidentally delete a source or target file. Or, the Integration Service may try to delete the file before the session finishes writing to the target.

To wait for a predefined event in the workflow:

1. Create an Event-Wait task and double-click the Event-Wait task to open it.
2. In the Events tab of the task, select Predefined.
3. Enter the path of the indicator file.
4. If you want the Integration Service to delete the indicator file after it detects the file, select the Delete Filewatch File option in the Properties tab.
5. Click OK.

Enabling Past Events

By default, the Event-Wait task waits for the Event-Raise task to trigger the event. By default, the Event-Wait task does not check if the event already occurred. You can select the Enable Past Events option so that the Integration Service verifies that the event has already occurred.

When you select Enable Past Events, the Integration Service continues executing the next tasks if the event already occurred.

Select the Enable Past Events option in the Properties tab of the Event-Wait task.

Working with the Timer Task

You can specify the period of time to wait before the Integration Service runs the next task in the workflow with the Timer task. You can choose to start the next task in the workflow at a specified time and date. You can also choose to wait a period of time after the start time of another task, workflow, or worklet before starting the next task.

The Timer task has the following types of settings:

- φ **Absolute time.** You specify the time that the Integration Service starts running the next task in the workflow. You may specify the date and time, or you can choose a user-defined workflow variable to specify the time.
- φ **Relative time.** You instruct the Integration Service to wait for a specified period of time after the Timer task, the parent workflow, or the top-level workflow starts.

For example, a workflow contains two sessions. You want the Integration Service wait 10 minutes after the first session completes before it runs the second session. Use a Timer task after the first session. In the Relative Time setting of the Timer task, specify ten minutes from the start time of the Timer task. Use a Timer task anywhere in the workflow after the Start task.
