Operations Center

Data Integrator Guide

September 2016

Contents

About This Guide

The *Data Integrator Guide* provides instructions for administering the Data Integrator adapters.

Audience

This guide is intended for system administrators who are responsible for setting up and administering Data Integrator adapters within the Operations Center environment. It is assumed that administrators are familiar with Operations Center and its user interface.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the *User Comments* feature at the bottom of each page of the online documentation.

Additional Documentation & Documentation Updates

This guide is part of the Operations Center documentation set. For the most recent version of the *Data Integrator Guide* and a complete list of publications supporting Operations Center, visit our Online Documentation Web Site at Operations Center online documentation.

The Operations Center documentation set is also available as PDF files on the installation CD or ISO; and is delivered as part of the online help accessible from multiple locations in Operations Center depending on the product component.

Additional Resources

We encourage you to use the following additional resources on the Web:

- NetIQ User Community (https://www.netiq.com/communities/): A Web-based community with a variety of discussion topics.

- NetIQ Support Knowledgebase (https://www.netiq.com/support/kb/?product%5B%5D=Operations_Center): A collection of in-depth technical articles.
- NetIQ Support Forums (https://forums.netiq.com/forumdisplay.php?26-Operations-Center): A Web location where product users can discuss NetIQ product functionality and advice with other product users.

Technical Support

You can learn more about the policies and procedures of NetIQ Technical Support by accessing its Technical Support Guide (https://www.netiq.com/Support/process.asp#_Maintenance_Programs_and).

Use these resources for support specific to Operations Center:

- Telephone in Canada and the United States: 1-800-858-4000
- Telephone outside the United States: 1-801-861-4000
- E-mail: support@netiq.com (support@netiq.com)
- Submit a Service Request: http://support.novell.com/contact/ (http://support.novell.com/contact/)

Documentation Conventions

In NetIQ documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path. The > symbol is also used to connect consecutive links in an element tree structure where you can either click a plus symbol (+) or double-click each element to expand them.

A trademark symbol (®, ™, etc.) denotes a NetIQ trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a forward slash to preserve case considerations in the UNIX* or Linux* operating systems.

1 Data Integrator Overview

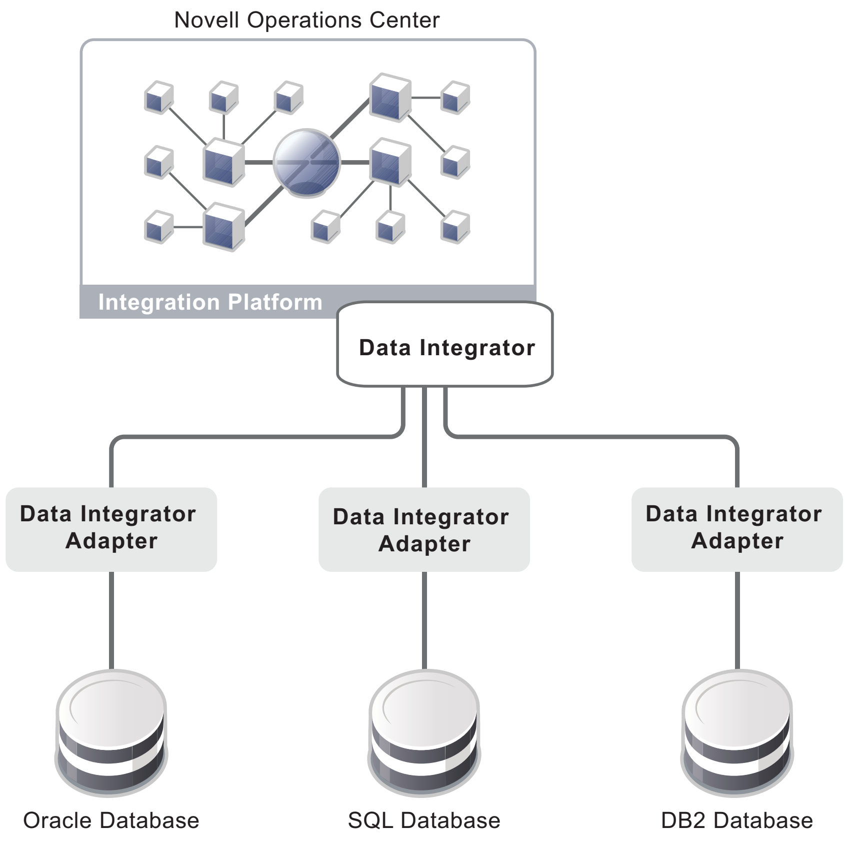
The Data Integrator is an optional component of the Operations Center platform and is licensed as a separate product. Use the Data Integrator to leverage business metrics from databases, including sales totals and help desk tickets, or analytics from business intelligence tools.

This guide explains how to define, configure, and deploy adapters using the Data Integrator. Use the Data Integrator environment to leverage and analyze database information within Operations Center. The Data Integrator adapter extracts data from specified sources and displays it in the Operations Center console.

- Section 1.1, “Overview,” on page 9
- Section 1.2, “How does Data Integrator Work?,” on page 11
- Section 1.3, “Development and Run Time Database Connections,” on page 12

1.1 Overview

Figure 1-1 Data Integrator Architecture



Operations Center offers rich integrations with virtually any source of technology data. The Data Integrator, which is an optional component of the Operations Center platform, extracts business metrics from databases, including sales totals and help desk tickets, and also extracts analytics from business intelligence tools.

The data extracted by a Data Integrator adapter is forwarded to the Operations Center platform for correlation, analysis and visualization, and can be further leveraged through Service Views and the dashboard.

Metrics from your data store can surface key performance indicators used by lines of business to measure their success. Using Operations Center to correlate these metrics, such as cross-referencing daily sales and help desk tickets, can provide additional information. This information can then populate real-time executive dashboards that identify the service performance of applications.

This is a true way of correlating business metrics with technology performance and availability. An increase in calls to a call center or a drop in revenue can be significant warnings of technology problems. Tying technology problems to their real business impact—such as the number of customers who complain—provides an accurate way to prioritize.

Analyzing this information over time can even help identify trends, such as a drop in sales totals when response time dips below a certain threshold. It can also assist in setting cost-effective performance thresholds for different geographies, customer groups and time periods.

- Section 1.1.1, “Secure Access to Relevant Data,” on page 10
- Section 1.1.2, “Point-and-Click Interface,” on page 10
- Section 1.1.3, “Share Adapter Definitions,” on page 10

1.1.1 Secure Access to Relevant Data

Instead of wading through large volumes of data, the Data Integrator extracts only the data needed to determine how technology is supporting the business. Also, the tool adheres to a database environment’s security settings, so users cannot extract data, unless they have permission to do so.

1.1.2 Point-and-Click Interface

A point-and-click interface enables quick access to database information. Data Integrator automatically discovers the available schemas in any relational SQL database, including DB2, Oracle, SQL Server, and Sybase. Just point and click to select fields or values for integration, then specify the time interval for capturing data, and how Operations Center should determine element or alarm severity.

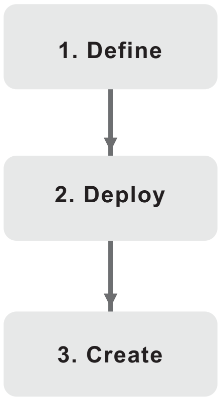
1.1.3 Share Adapter Definitions

The Data Integrator has a central management facility. Defined integrations can be stored and reused, which is valuable to organizations with many instances of the same database distributed throughout their environment. Each Data Integrator runtime license supports access to a single database schema.

1.2 How does Data Integrator Work?

Data Integrator is an optional component of the Operations Center platform and requires a license key.

The Data Integrator provides a development environment to define custom adapters that leverage information from strategic and business databases.



- Section 1.2.1, "Step 1. Define the Adapter Definition," on page 11
- Section 1.2.2, "Step 2. Deploy the Adapter Definition," on page 11
- Section 1.2.3, "Step 3. Create the Adapters," on page 12

1.2.1 Step 1. Define the Adapter Definition

The first step is to create an adapter definition. Define everything about the adapter type—the database it uses, the elements that are created, including their properties and icons, and which alarm information is available.

The Data Integrator Definition Editor enables pointing and clicking through the process of defining all aspects of the adapter definition. It reads the database schema and allows dragging and dropping table names and table columns directly from the database schema in order to construct the adapter definition.

Adapter definitions can be reused across organizations that have many instances of the same database distributed throughout their environment.

1.2.2 Step 2. Deploy the Adapter Definition

The next step is deploying the adapter. Operations Center makes this new adapter definition available as a run-time adapter. When a user deploys an adapter, the deployed definition is included in the *Adapter Type* drop-down list.

1.2.3 Step 3. Create the Adapters

To leverage the new adapter, Admin users create adapter instances just as they create any other Operations Center adapter. When started, the new adapter automatically leverages data from the defined database and builds out all the elements, properties, and alarms. Everything is available in the *Elements* hierarchy in the Operations Center console.

A deployed adapter definition can be edited and maintained based on changing needs or database changes. However, changes do not take effect until the adapter definition is redeployed, thereby protecting users against intermediate changes.

1.3 Development and Run Time Database Connections

The Data Integrator allows utilizing one database for development and another for adapter run time. The following sections provide more information on specifying development and run-time databases:

- Section 3.4, "Database Connections," on page 28
- Section 4.6, "Setting a Run-Time Database Connection," on page 39

Data Integrator is read-only when interacting with a database. It does not write to the database.

The Data Integrator supports JBDC drivers in general. Connection templates are provided for commonly used databases. For more information about supported database connections, see "JDBC Support in Data Integrator" in the *Operations Center Getting Started Guide*.

To deploy JDBC drivers for other databases, contact Support (https://www.netiq.com/support/) to have it signed, then save it to the `/`*`OperationsCenter_install_path`*`/html/client/deploy` directory. For more information, see Section 2.2.4, "Signing a JDBC Driver," on page 20.

2 Managing Adapter Definitions

The Data Integrator utilizes adapter definitions that instruct Operations Center how to create and populate an adapter based on data stored in a relational database.

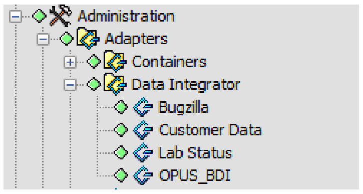
2.1 Understanding Adapters in Data Integrator

After deploying the adapter definition, you can create and start an adapter instance. A new branch of the *Elements* hierarchy displays, representing the data from the relational database. This new hierarchy consists of folders, elements, and alarm data.

Adapter definitions display in the *Explorer* pane under *Administration* > *Adapters* > *Data Integrator*. They are not available as Operations Center adapters until they are deployed.

Figure 2-1 illustrates adapter definitions displaying under the *Data Integrator* element:

Figure 2-1 *Explorer Pane*



The condition of an adapter definition alerts you to adverse changes. Table 2-1 identifies the possible conditions of an adapter definition:

Table 2-1 *Adapter Definition Conditions*

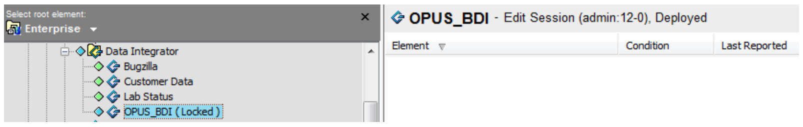
Status	Description
🟥 CRITICAL	The adapter definition is orphaned. The edit session abruptly ended without a proper save performed.
🟨 MINOR	The adapter definition is currently deployed.
🟦 INFO	The adapter definition is currently being edited by a user.
🟩 OK	The adapter definition is in an OK state. It is not deployed or opened by an editor.

The title bar also displays basic information about a selected adapter definition, including:

- Whether an edit session is open for the definition and the name of the user who is editing it
- Whether the definition is deployed as an adapter
- The number of additional users viewing the definition in a read-only state

For example, in Figure 2-2 an Admin user is editing the Support Issues definition:

Figure 2-2 *Explorer Pane and View Title Bar*



No additional users are viewing the definition and there are no current deployments.

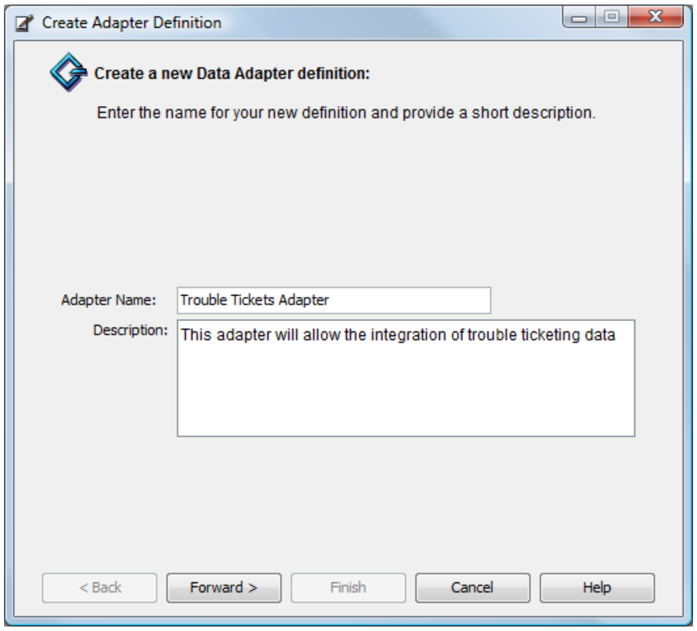
You can select an adapter definition element in the *Explorer* pane to view information in the title bar.

2.2 Creating Adapter Definitions

The first step in creating a new adapter definition is to set up basic information about the adapter and the database connection. The Create Adapter Definition wizard steps through these initial steps.

To create an adapter definition:

1. In the *Explorer* pane, expand the *Administration* root element > *Adapters*.
2. Right-click *Data Integrator*, then select *Create Definition* to open the Create Adapter Definition dialog box:



Create Adapter Definition

Create a new Data Adapter definition:

Enter the name for your new definition and provide a short description.

Adapter Name: Trouble Tickets Adapter

Description: This adapter will allow the integration of trouble ticketing data

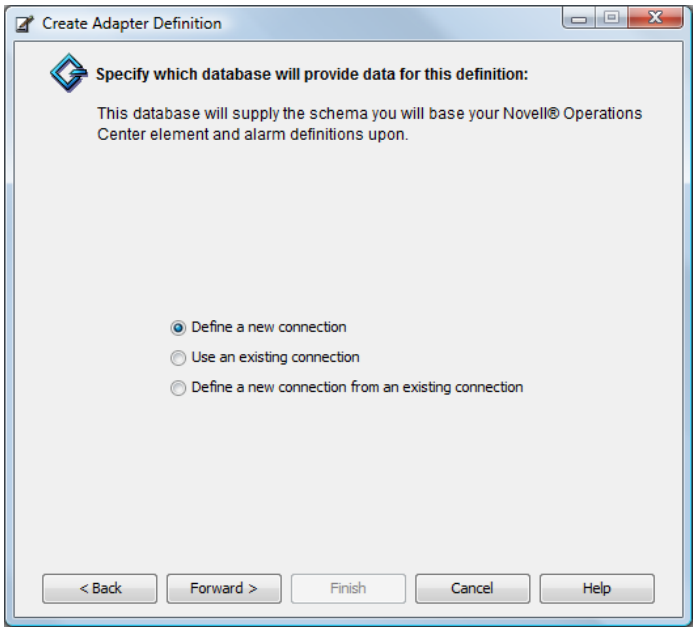
3 In the Create Adapter Definition dialog box, specify an Adapter Name and Description:

Adapter Name: The name of the new adapter that is created when the definition is deployed. When deployed, this is the name that the user selects from the *Adapter Type* drop-down list in the Create Adapter dialog box to create an adapter instance.

Description: Provides additional information about the definition. This description displays when the mouse cursor is hovered over the adapter definition name under the *Adapters* hierarchy.

4 Click *Forward* to continue.

5 Select the associated radio button to define a new connection, use an existing connection, or define a new connection from an existing connection:



Specify which database will provide data for this definition:

This database will supply the schema you will base your Novell® Operations Center element and alarm definitions upon.

[✓ Define a new connection]
Use an existing connection
Define a new connection from an existing connection

Define a new connection: Define a database connection that has not been used before with Data Integrator. Continue to step 5.

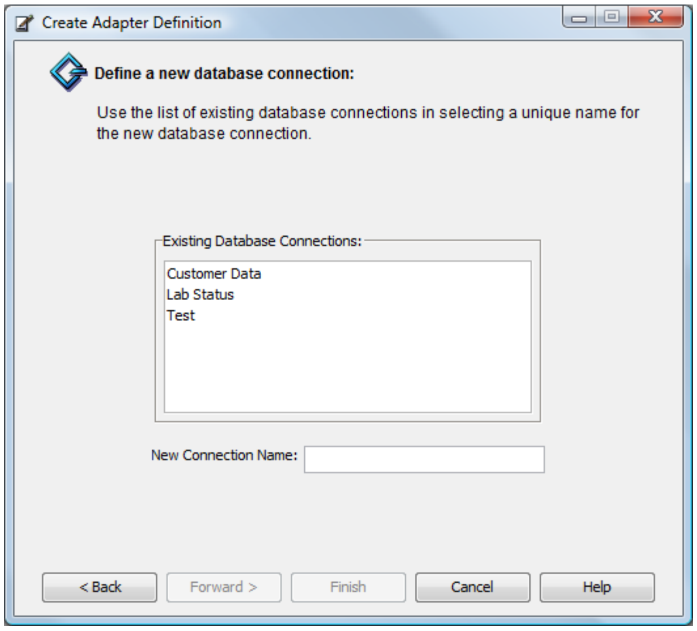
Use an existing connection: Select an existing connection to use. Skip ahead to step 6.

Define a new connection from an existing connection: Define a new database connection by modifying the properties of a current connection. Skip ahead to step 7.

6 Click *Forward* to continue.

7 Do one of the following:

- If defining a new connection, specify a name for the database connection. Click *Forward*.
- If using an existing connection, select the connection you wish to use from the *Database* drop-down list. Click *Forward* and skip to step 9.
- If defining a new connection from an existing connection, do the following:
    1. Select the desired connection you wish to use from the *Existing Connection* drop-down list.
    2. In the *New Connection Name* field, specify the name of the new connection.
    3. Click *Forward* to continue.


Create Adapter Definition

Define a new database connection:

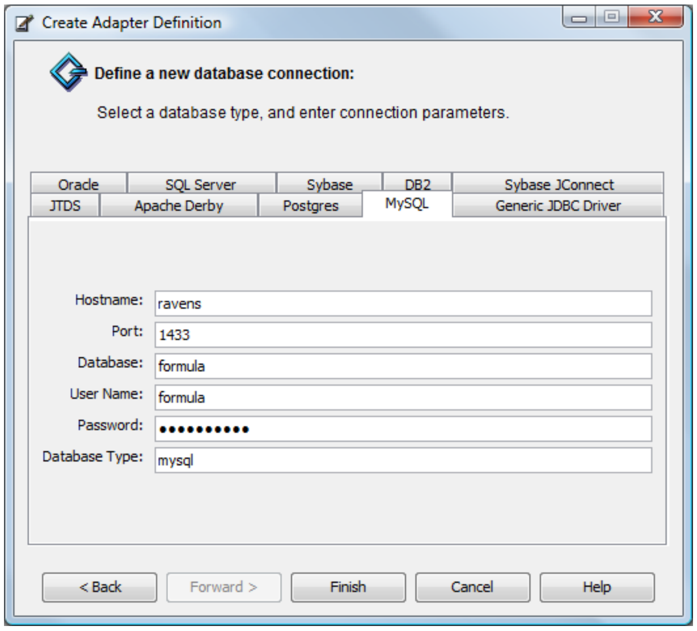
Use the list of existing database connections in selecting a unique name for the new database connection.

Existing Database Connections:

Customer Data
Lab Status
Test

New Connection Name:

8 If using an existing connection or creating a new connection from an existing connection, modify the properties as required for connection.


Create Adapter Definition

Define a new database connection:

Select a database type, and enter connection parameters.

Hostname:	ravens
Port:	1433
Database:	formula
User Name:	formula
Password:	••••••••••
Database Type:	mysql

Hostname: The database server name. Applies to all connection types except JDBC.

Port: The port number where the database listens for communications. Applies to all connection types except JDBC.

Database: The database name (when defining a SQL Server, Sybase, or DB2 database connection). Applies to all connection types except JDBC.

Server ID: The database name (when defining an Oracle database connection). Applies to all connection types except JDBC.

JDBC Driver: The name of the class used to initiate a JDBC driver. Typically, the class name is in a format similar to `com.product.Driver`. You can later hide this setting by not publishing the value as an adapter property or allow the driver to be overwritten at deployment time. Applies only to JDBC connections.

JDBC URL: Specifies how the driver connects to a JDBC database. Applies only to JDBC Connections.

Time Stamp Query: At runtime, Data Integrator periodically queries the JDBC database for availability. This query setting specifies how the database is queried to ensure it is available and servicing requests. Applies only to JDBC Connections.

User Name: The name of the user account.

Password: The password associated with the user name.

Windows Domain: The name of the Windows domain to use for Windows native authentication. Applies to JTDS and SQL Server connections.

For information about configuring the server for Windows Authentication, see Section 2.2.6, "Configuring the Server for Windows Authentication," on page 20.

Database Type: This value (`Oracle`, `SQL Server`, `Sybase`, `DB2`, and so on) is prefilled, based on the selected database tab. For information about using a nondefault database driver, see Section 2.2.5, "Using a Different Database Driver for a Data Source," on page 20.

If you are creating or modifying a mySQL connection, see Section 2.2.1, "Notes on Connecting to MySQL," on page 18 and Section 2.2.4, "Signing a JDBC Driver," on page 20.

If you are creating or modifying a JDBC connection, see Section 2.2.3, "Notes on Connecting to JDBC," on page 19 and Section 2.2.4, "Signing a JDBC Driver," on page 20.

If you are creating or modifying an Oracle RAC connection, see Section 2.2.2, "Notes on Connecting to Oracle RAC," on page 18 and Section 2.2.4, "Signing a JDBC Driver," on page 20.

9 Click *Finish* to complete the process and automatically open the Data Integrator Definition Editor.

A new adapter definition displays in the *Explorer* pane under *Administration* > *Adapters* > *Data Integrator*.

For basic information regarding using the Definition Editor and user preferences, see Chapter 3, "The Data Integrator Definition Editor," on page 25. This section also contains information about changing development data source settings.

For information regarding Groups, see Section 5.2, "Defining Groups," on page 42.

For information regarding Database Elements, see Section 5.4, "Defining Elements from Database Information," on page 46.

For information regarding Generators, see Section 5.3, "Defining Dynamic Groups (Generator Elements)," on page 44.

For information regarding alarms, see Section 5.5, "Creating Alarms," on page 49.

The following sections are referenced for additional information from some of the preceding steps:

- Section 2.2.1, "Notes on Connecting to MySQL," on page 18
- Section 2.2.2, "Notes on Connecting to Oracle RAC," on page 18
- Section 2.2.3, "Notes on Connecting to JDBC," on page 19
- Section 2.2.4, "Signing a JDBC Driver," on page 20
- Section 2.2.5, "Using a Different Database Driver for a Data Source," on page 20
- Section 2.2.6, "Configuring the Server for Windows Authentication," on page 20

2.2.1 Notes on Connecting to MySQL

The MySQL JDBC driver is Open Source GPL (General Public License). To use this driver, download it from http://dev.mysql.com/downloads/connector/j/3.1.html (http://dev.mysql.com/downloads/connector/j/3.1.html). The driver must be placed in the following two locations, and one copy of the driver must be signed for use in the Data Integrator:

- The Operations Center server directory: `/`*`OperationsCenter_install_path`*`/classes/ext`. Place the distributed driver as unsigned (unmodified).
- The Operations Center console deployment directory: `/`*`OperationsCenter_install_path`*`/html/client/deploy`. This copy must be signed by NetIQ for Operations Center. For more information, see Section 2.2.4, "Signing a JDBC Driver," on page 20.

2.2.2 Notes on Connecting to Oracle RAC

The following procedures describe various configuration information required to use Oracle RAC for either the design-time database or the adapter run-time database. While the use of the JDBC OCI provider is not required for the design-time connection, it is required for the run-time adapter.

To specify an Oracle RAC database to use while creating (design-time) the adapter definition:

1 Perform configuration steps as required for Oracle RAC on the Operations Center server.

While it is not necessary to have the OCI package installed, all connections require the `ojdbc6.jar` file to be manually deployed to the Operations Center server.

For instructions, see "Oracle RAC" in the *Operations Center Server Configuration Guide*.

2 Define the database using the *Generic JDBC Driver* tab, and do the following:

2a Specify the following value for the JDBC Driver:

```
oracle.jdbc.driver.OracleDriver
```

2b Do one of the following to specify the JDBC URL:

- To use the JDBC thin driver to connect to the Oracle RAC server, specify:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=host1_ipaddress)(PORT=host1_portnumber)) (ADDRESS=(PROTOCOL=TCP)
(HOST=host2_ipaddress)(PORT=host2_portnumber)) (LOAD_BALANCE=yes)
(CONNECT_DATA=(SERVER=DEDICATED) (SERVICE_NAME=service_domain_name)
(FAILOVER_MODE=(TYPE=SELECT)(METHOD=BASIC) (RETRIES=180)(DELAY=5))))
```

- If the OCI package is installed on the client machine, do one of the following:

To use the JDBC OCI connection (supports FCF and Transparent Application Failover (TAF), specify:

```
jdbc:oracle:oci:@racdb_configuration_name
```

or, if the connection is not defined in the `tsnnames.ora` file, specify:

```
jdbc:oracle:oci:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=host1_ipaddress)(PORT=host1_portnumber)) (ADDRESS=(PROTOCOL=TCP)
(HOST=host2_ipaddress)(PORT=host2_portnumber))
(LOAD_BALANCE=yes)(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=service_domain_name)(FAILOVER_MODE=(TYPE=SELECT)
(METHOD=BASIC)(RETRIES=180)(DELAY=5))))
```

2c Specify the following query command for the Time Stamp Query:

```
select systimestamp from dual
```

To specify an Oracle RAC database to use for the run-time adapter definition:

1 Perform configuration steps as required for Oracle RAC on the Operations Center server.

The default database properties of the run-time adapter must use the JDBC OCI provider.

For instructions, see "Oracle RAC" in the *Operations Center Server Configuration Guide*.

2 Define the database using the *Generic JDBC Driver* tab, and do the following:

2a Specify the following value for the JDBC Driver:

```
oracle.jdbc.driver.OracleDriver
```

2b Do one of the following to specify the JDBC URL:

- If the service definition for the RAC connection is defined in the `tsnnames.ora` file, specify:

  ```
  jdbc:oracle:oci:@racdb_configuration_name
  ```

- or, if the service definition for the RAC connection is not defined in the `tsnnames.ora` file, specify:

  ```
  jdbc:oracle:oci:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host1_ipadd
  ress)(PORT=host1_portnumber))(ADDRESS=(PROTOCOL=TCP)
  (HOST=host2_ipaddress)(PORT=host2_portnumber))(LOAD_BALANCE=yes)(CONNE
  CT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=service_domain_name)
  (FAILOVER_MODE=(TYPE=SELECT)(METHOD=BASIC)(RETRIES=180)(DELAY=5))))
  ```

2c Specify the following query command for the Time Stamp Query:

```
select systimestamp from dual
```

2.2.3 Notes on Connecting to JDBC

To install the generic JDBC driver, locate the appropriate driver on a product distribution disk or download it. The driver must be placed in the following two locations, and one copy of the driver must be signed for use in the Data Integrator:

- The Operations Center server directory: `/OperationsCenter_install_path/classes/ext`. Place the distributed driver as unsigned (unmodified).
- The Operations Center console deployment directory: `/OperationsCenter_install_path/html/client/deploy`. This copy must be signed by NetIQ for Operations Center. For more information, see Section 2.2.4, "Signing a JDBC Driver," on page 20.

2.2.4 Signing a JDBC Driver

The `/OperationsCenter_install_path/html/client/Readme.txt` file describes the subdirectories in the `/OperationsCenter_install_path/html/client` directory.

In the `deploy` directory, place patches or other `.jar` components that have been signed by Operations Center. This directory is not be renamed when upgrading Operations Center versions, so that its contents remain the same after upgrading.

Because of the tightened security model introduced by Oracle Java 7 Update 45, all .jar components, including those from third-parties, must be signed by NetIQ for Operations Center. Contact Support (https://www.netiq.com/support/) to have the content signed.

2.2.5 Using a Different Database Driver for a Data Source

If there are multiple database drivers available on a machine and you want to use a specific database driver, you can specify the exact driver to use in the *Database Type* connection parameter. However, if the database driver should be used in the run-time database connection, there is an additional step that must be completed.

To change the default database driver:

1 When defining or editing a database connection, in the *Database Type* field, replace the value with:

```
driver|url|timequery
```

where:

- *driver* is the driver's class name
- *url* is the JDBC URL string
- *timequery* (for Oracle) is select sysdate from dual

The pipe (|) characters are required.

2 If the driver is to be used for the run-time database setting, in the database tab for top-level settings, insert the same string into the *Database Type* field.

If the *Add to Adapter Page* check box is selected for the *Database Type*, the setting is seen as the default property value when using an instance of the adapter.

For more information about the database properties and run-time database settings, see Section 4.6, "Setting a Run-Time Database Connection," on page 39.

2.2.6 Configuring the Server for Windows Authentication

If you are using a JTDS or SQL Server connection with Windows Authentication, you must configure the server for Windows Authentication.

To configure the server for Windows Authentication:

1 Verify or configure the following requirements on the MSSQL Server:

- Verify the server is configured to use Windows Authentication.
- Verify the machine is on the domain.
- Add an associated domain user with appropriate permissions.
- If MSSQL is setup to use an instance name, you must turn on the SQL Server Browser Service. Then add the user to the instance's security group.

2 To configure Operations Center for Windows Single Sign-On (SSO), do the following:

2a Do one of the following to configure the Operations Center server:

- **On Unix:** No native libraries required, but you must provide user, password.
- **On Windows:**
  - Native libraries are optional. If you don't use the native library, you must provide user, password.
  - The domain user must be a member of the Windows Administrators group.

2b If running Operations Center as a service, the service must use the same domain user account as the database.

If you are receiving an `NT AUTHORITY\ANONYMOUS LOGON` exception, verify that the account used by the service and the database are the same.

2c Leave the *User* and *Password* properties blank (empty) for Data Integrator.

2d Leave the *User*, *Password*, and *Domain* adapter properties blank (empty).

2e Copy the /*OperationsCenter_install_path*/html/classes/win32/ntlmauth32.jar or /*OperationsCenter_install_path*/html/classes/win64/ntlmauth64.jar file into the /*OperationsCenter_install_path*/classes/ext directory.

2f Restart the Operations Center server.

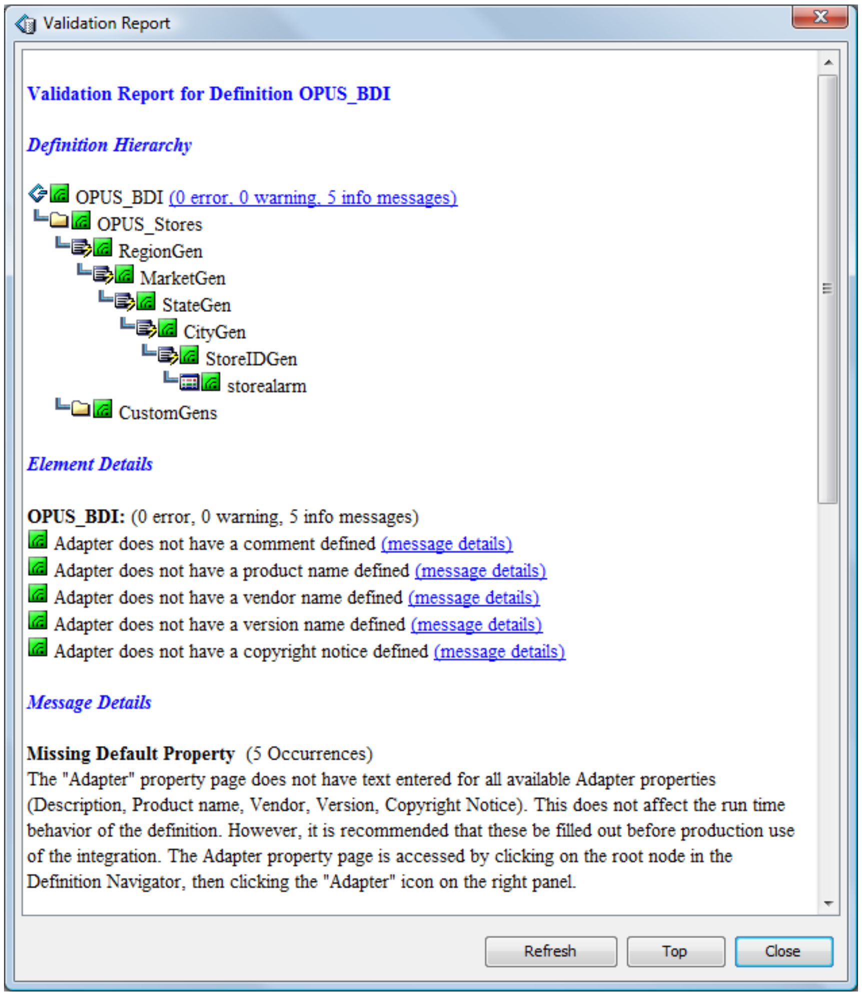
2.3 Validating an Adapter Definition

At any time, obtain a validation report about an adapter definition which reports on information about elements generated by the definition, identifies any problems with the definition, and statistics, such as the number of error messages associated with the definition and the number of elements and alarms defined.

To run a validation report on the adapter definition:

1 From the Data Integrator Definition Editor, click *Definition*, then select *Validate*.

The validation report displays:


Validation Report

Validation Report for Definition OPUS_BDI

Definition Hierarchy

```
OPUS_BDI (0 error, 0 warning, 5 info messages)
└OPUS_Stores
  └RegionGen
    └MarketGen
      └StateGen
        └CityGen
          └StoreIDGen
            └storealarm
  └CustomGens
```

Element Details

OPUS_BDI: (0 error, 0 warning, 5 info messages)
Adapter does not have a comment defined (message details)
Adapter does not have a product name defined (message details)
Adapter does not have a vendor name defined (message details)
Adapter does not have a version name defined (message details)
Adapter does not have a copyright notice defined (message details)

Message Details

Missing Default Property (5 Occurrences)
The "Adapter" property page does not have text entered for all available Adapter properties (Description, Product name, Vendor, Version, Copyright Notice). This does not affect the run time behavior of the definition. However, it is recommended that these be filled out before production use of the integration. The Adapter property page is accessed by clicking on the root node in the Definition Navigator, then clicking the "Adapter" icon on the right panel.

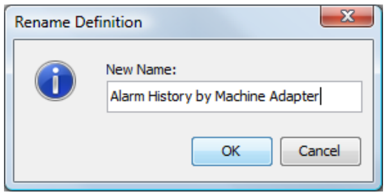
2.4 Renaming an Adapter Definition

All adapter definitions are located under *Administration* > *Adapters* > *Data Integrator*. It is possible to rename a definition.

Definitions that are deployed or are currently edited cannot be renamed. Close all editors and undeploy definitions before renaming an adapter definition.

To rename an adapter definition:

1. In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.
2. Right-click a definition, then select *Rename Definition* to open the Rename Definition dialog box:


Rename Definition

New Name: Alarm History by Machine Adapter

3 Enter a new name for the definition.

4 Click *OK* to apply the new name to the definition.

The definition is renamed (and resorted, if necessary) in the *Explorer* pane hierarchy.

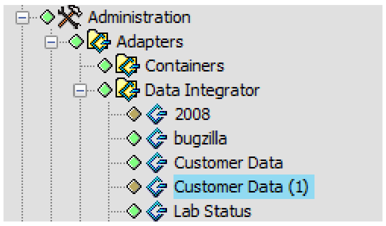
2.5 Duplicating an Adapter Definition

All adapter definitions are found under *Administration* > *Adapters* > *Data Integrator*. Copy a definition to create a definition with similar requirements.

To duplicate an adapter definition:

1 In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.

2 Right-click a definition, then select *Duplicate Definition*:



A new copy of the selected definition is created under *Administration* > *Adapters* > *Data Integrator* and is ready to be renamed and modified.

2.6 Editing an Adapter Definition

All adapter definitions are found under the *Administration* root element > *Adapters* > *Data Integrator*.

The first concurrent edit session has read/write access to a definition. Subsequent concurrent edit sessions have read-only access to the definition. Changes made to a definition from a read-only edit session are not saved.

To edit an adapter definition:

1 In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.

2 Right-click a definition, then select *Edit Definition*.

3 The Data Integrator Definition Editor opens.

For basic information regarding using the Definition Editor and user preferences and updating the development data source, see Chapter 3, “The Data Integrator Definition Editor,” on page 25.

For information regarding Groups, see Section 5.2, “Defining Groups,” on page 42.

For information regarding Database Elements, see Section 5.4, “Defining Elements from Database Information,” on page 46.

For information regarding Dynamic Groups, see Section 5.3, “Defining Dynamic Groups (Generator Elements),” on page 44.

For information regarding alarms, see Section 5.5, “Creating Alarms,” on page 49.

2.7 Deleting an Adapter Definition

All adapter definitions are found under the *Administration* > *Adapters* > *Data Integrator* tree. An adapter definition must be undeployed in order to delete it.

To delete an adapter definition:

1. In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.
2. Right-click a definition, then select *Delete Definition*.
3. Click *Yes* to confirm and delete the definition.

   The definition is removed from the *Administration* > *Adapters* > *Data Integrator* hierarchy.
4. Close all editors and undeploy the definitions before trying to delete them.

   Definitions that are deployed or are currently edited cannot be deleted.

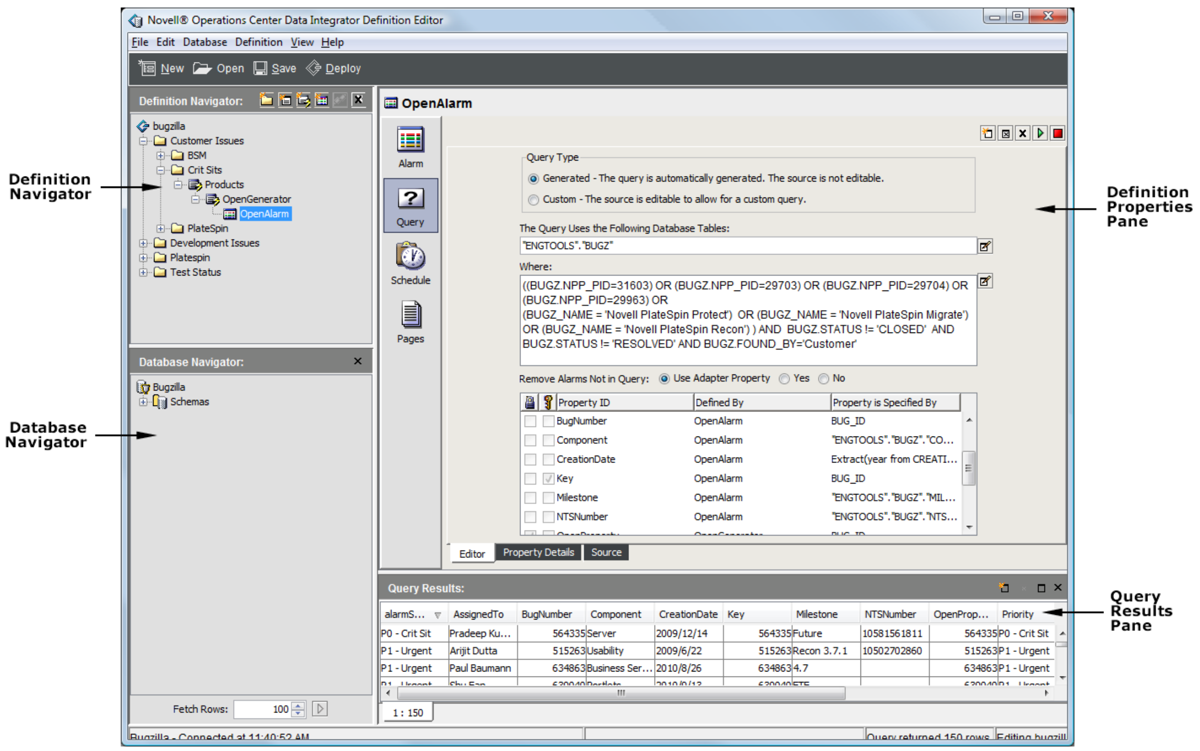
3 The Data Integrator Definition Editor

The Data Integrator Definition Editor is used to create and edit adapter definitions, as well as to query and extract information from database tables and create elements and alarms in Operations Center.

3.1 Understanding the Data Integrator Definition Editor

Figure 3-1 shows the Data Integrator Definition Editor and its four panes:

***Figure 3-1** Data Integrator Definition Editor*



Query Type

(●) Generated - The query is automatically generated. The source is not editable.
( ) Custom - The source is editable to allow for a custom query.

The Query Uses the Following Database Tables:

"ENGTOOLS"."BUGZ"

Where:

```
((BUGZ.NPP_PID=31603) OR (BUGZ.NPP_PID=29703) OR (BUGZ.NPP_PID=29704) OR
(BUGZ.NPP_PID=29963) OR
(BUGZ_NAME = 'Novell PlateSpin Protect')  OR (BUGZ_NAME = 'Novell PlateSpin Migrate')
OR (BUGZ_NAME = 'Novell PlateSpin Recon') ) AND  BUGZ.STATUS != 'CLOSED'  AND
BUGZ.STATUS != 'RESOLVED' AND BUGZ.FOUND_BY='Customer'
```

Remove Alarms Not in Query: [✓ Use Adapter Property] Yes No

		Property ID	Defined By	Property is Specified By
		BugNumber	OpenAlarm	BUG_ID
		Component	OpenAlarm	"ENGTOOLS"."BUGZ"."CO…
		CreationDate	OpenAlarm	Extract(year from CREATI…
	✓	Key	OpenAlarm	BUG_ID
		Milestone	OpenAlarm	"ENGTOOLS"."BUGZ"."MIL…
		NTSNumber	OpenAlarm	"ENGTOOLS"."BUGZ"."NTS…

alarmS…	AssignedTo	BugNumber	Component	CreationDate	Key	Milestone	NTSNumber	OpenProp…	Priority
P0 - Crit Sit	Pradeep Ku…	564335	Server	2009/12/14	564335	Future	10581561811	564335	P0 - Crit Sit
P1 - Urgent	Arijit Dutta	515263	Usability	2009/6/22	515263	Recon 3.7.1	10502702860	515263	P1 - Urgent
P1 - Urgent	Paul Baumann	634863	Business Ser…	2010/8/26	634863	4.7		634863	P1 - Urgent

The Definition Editor is comprised of these panes:

- **Definition Navigator:** Use to create new groups, elements, generators, or alarm definitions. It provides a hierarchical view of elements.
- **Database Navigator:** The database schema is automatically read into the Data Integrator Editor, and the *Database Navigator* pane provides a hierarchical list of views, database tables and columns for the development database. Use the drag and drop feature to drag a table or column to the *Definition Properties* pane to define new queries or properties assignments.
- **Definition Properties:** Use to specify basic information about an adapter definition, including adapter icons, run-time adapter database connections and properties, and scripts that run based on adapter activity.
- **Query Results:** The adapter definition uses database queries to create elements and issue alarms. The *Query Results* pane displays the output or an executed query.

You can use the buttons at the top of the Definition Editor dialog box to perform the following functions:

- **New:** Creates a new adapter definition.
- **Open:** Opens a specified adapter definition in the Editor.
- **Save:** Saves changes made to the definition. The Definition Editor has an auto-save feature, however you can click *Save* at any time.
- **Deploy:** To implement changes made in the Definition Editor, click *Deploy*. The adapter is restarted with the definition changes.

3.2 Opening the Definition Editor

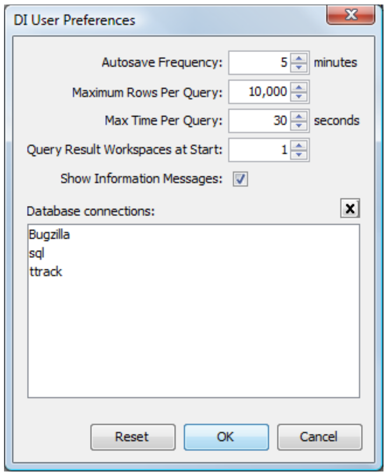
1. In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.
2. Right-click an adapter definition, then select *Edit Definition* to open the Definition Editor.

   The Definition Editor attempts to connect to the database. You can view the connection status in the bottom left of the Editor dialog box.

3.3 Configuring User Preferences

The Data Integrator User Preferences dialog box enables selecting various settings, including an auto-save feature and default maximum settings for queries.

Figure 3-2 *Data Integrator User Preferences Dialog*



Autosave Frequency: 5 minutes
Maximum Rows Per Query: 10,000
Max Time Per Query: 30 seconds
Query Result Workspaces at Start: 1
Show Information Messages: ✓
Database connections:
Bugzilla
sql
ttrack

User preferences set auto-save frequency and query defaults and maintain database connections.

Set the following preferences for the Data Integrator Definition Editor:

- Section 3.3.1, "Setting an Auto Save Frequency," on page 27
- Section 3.3.2, "Setting the Maximum Number of Table Rows Per Query," on page 27
- Section 3.3.3, "Setting the Maximum Time Allowed Per Query," on page 28
- Section 3.3.4, "Viewing Informational Messages When Using the Definition Editor," on page 28
- Section 3.3.5, "Deleting a Database Connection," on page 28

3.3.1 Setting an Auto Save Frequency

1. In the Definition Editor, click *Edit* > *User Preferences*.
2. Use the *Autosave Frequency* spinner buttons to select the interval in minutes for the automatic save feature to run.

3.3.2 Setting the Maximum Number of Table Rows Per Query

1. In the Definition Editor, click *Edit* > *User Preferences.*
2. Use the *Maximum Rows Per Query* spinner buttons to select the number of rows that are allowed per query.

   Enter -1 to allow an unlimited number of rows per query.

3.3.3 Setting the Maximum Time Allowed Per Query

1. In the Definition Editor, click *Edit* > *User Preferences.*
2. Use the *Max Time Per Query* spinner buttons to select the time in seconds that are allowed to elapse per query.

   Enter -1 to allow an unlimited amount of time.

3.3.4 Viewing Informational Messages When Using the Definition Editor

1. In the Definition Editor, click *Edit* > *User Preferences.*
2. Select the *Show Informational Messages* check box to display alerts in specific cases.

   For example, an informational message might be displayed in certain instances, such as when trying to edit a definition that is read-only or when a database connection must be defined.
3. (Optional) Deselect the *Show Informational Messages* check box to disable the message feature.

3.3.5 Deleting a Database Connection

1. In the Definition Editor, click *Edit* > *User Preferences.*

   All database connection profiles are listed in the *Database Connections* list.
2. Select a database connection.
3. Click *Delete* to delete the database connection.

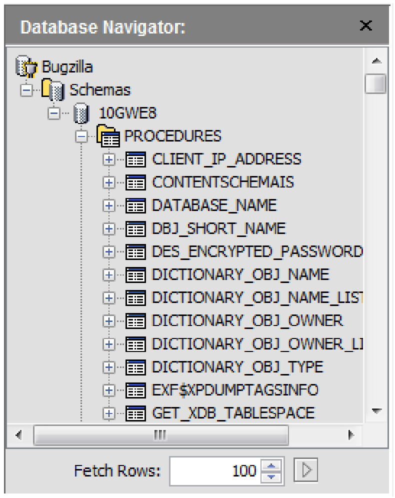
3.4 Database Connections

The adapter definition requires defining a database connection to access the database schema. Because the Data Integrator is a tool for the development environment, the development database connection could differ from the production database accessed at run-time. You can define two different database connections. For more information, see Section 4.6, “Setting a Run-Time Database Connection,” on page 39.

If you do not create and assign a schedule to the database element or alarm definition, the DEFAULT_SCHEDULE is automatically used.

If the current database connection is deleted for a definition currently being edited, then the current connection remains in effect. The next time the definition is edited, a new database connection must be defined.

Figure 3-3 *Database Navigator Pane*



Fetch Rows: 100

View and use database schema to create queries for extracting element and alarm information into Operations Center.

The following sections cover defining and testing a development database:

- Section 3.4.1, “Viewing a Database Schema,” on page 29
- Section 3.4.2, “Running a Sample Query,” on page 29
- Section 3.4.3, “Changing the Development Data Source,” on page 30
- Section 3.4.4, “Testing the Development Database Connection,” on page 32

3.4.1 Viewing a Database Schema

During the adapter definition creation process, a development database is defined as you specify basic details in the Create Adapter Definition wizard. Data Integrator automatically discovers and displays the database schema in the *Database Navigator* pane.

Click the plus (+) signs next to the database elements to open and explore the tables and columns of the database schema.

3.4.2 Running a Sample Query

The *Database Navigator* pane provides a quick way to view sample data. Click a table, then click *Run* (the green arrow) in the *Fetch Rows* section. Table data displays in the *Query Results* pane.

3.4.3 Changing the Development Data Source

The development data source is specified as a step in the Create Adapter Definition wizard (Section 2.2, “Creating Adapter Definitions,” on page 14).

After you have started to define and build out all the necessary element definitions for an adapter definition, then changing databases is restricted to the same database type. For example, if an adapter definition relies on an Oracle database, you can modify the adapter definition to use a different Oracle database with the same table names, and so on. However, you cannot change to a different database type, such as Sybase.

WARNING: If it is necessary to change the database type, close all Data Integrator Editor dialog boxes and undeploy the definition before trying to change the database type.

To change the development data source, do one the following:

- “Changing the Development Data Source Using a New Database Connection” on page 30
- “Changing the Development Data Source Using an Existing Database Connection” on page 31
- “Changing the Development Data Source Using a New Connection Created from an Existing One” on page 31
- “Changing Settings for the Development Data Source” on page 31
- “Refreshing the Connection to the Development Data Source” on page 31

Changing the Development Data Source Using a New Database Connection

1. In the Definition Editor, click *Database* > *Open Connection* to open the *Update Business Data Integration* wizard.
2. Select the *Define a New Connection* radio button, then click *Forward*.
3. Specify the new name in the *Database Name* field, then click *Forward* to continue.
4. Specify the database parameters using the appropriate database tab:

   Hostname: The database server name. Applies to all connection types except JDBC.

   Port: The port number where the database listens for communications. Applies to all connection types except JDBC.

   Database: The database name (when defining a SQL Server, Sybase, or DB2 database connection). Applies to all connection types except JDBC.

   Server ID: The database name (when defining an Oracle database connection). Applies to all connection types except JDBC.

   JDBC Driver: The name of the class used to initiate a JDBC driver. Typically, the class name is in a format similar to `com.product.Driver`. You can hide this setting by not publishing the value as an adapter property or allow the driver to be overwritten at deployment time. Applies only to JDBC connections.

   JDBC URL: Specifies how the driver connects to a JDBC database. Applies only to JDBC Connections.

   Time Stamp Query: At runtime, Data Integrator periodically queries the JDBC database for availability. This query setting specifies how the database is queried to ensure it is available and servicing requests. Applies only to JDBC Connections.

   User Name: The name of the user account.

Password: The password associated with the user name.

Windows Domain: The name of the Windows domain to use for Windows native authentication. Applies to JTDS and SQL Server connections.

For information about configuring the server for Windows Authentication, see Section 2.2.6, “Configuring the Server for Windows Authentication,” on page 20.

Database Type: This value (`Oracle`, `SQL Server`, `Sybase`, `DB2`, and so on) is prefilled based on the selected database tab. For information about using a nondefault database driver, see Section 2.2.5, “Using a Different Database Driver for a Data Source,” on page 20.

Changing the Development Data Source Using an Existing Database Connection

1. In the Definition Editor, click *Database* > *Open Connection* to open the *Update Business Data Integration* wizard.
2. Select the *Use an Existing Connection* radio button, then click *Forward*.
3. Click the *Connection* drop-down list, then select an existing database connection.
4. Click *Forward* to review or edit database settings.

Changing the Development Data Source Using a New Connection Created from an Existing One

1. In the Definition Editor, click *Database* > *Open Connection* to open the *Update Business Data Integration* wizard.
2. Select the *Use a New Connection from an Existing Connection* radio button, then click *Forward*.
3. Click the *Existing Connection* drop-down list, then select an existing connection.
4. Enter the name for the new connection in the *New Connection Name* field, then click *Forward*.
5. Edit the prefilled parameters to customize the new connection.
6. Click *Finish* to save the new settings and use the new database.

Changing Settings for the Development Data Source

1. In the Definition Editor, click *Database* > *Open Connection* to open the *Update Business Data Integration* wizard.
2. Click the *Use an Existing Connection* radio button, then click *Forward*.
3. Click the *Connection* drop-down list, then select the current database connection.
4. Click *Forward* to review or edit database settings.
5. Click *Finish* to save the new settings and use the new database.

Refreshing the Connection to the Development Data Source

1. In the Definition Editor, click *Database* > *Refresh Connection*.

   The editor attempts to reestablish the database connection.

3.4.4 Testing the Development Database Connection

1. In the Definition Editor, click *Database* > *Test Connection*.

   A confirmation message displays when the test is complete.
2. Click *OK* to close the message box.

4 Specifying Adapter Definition Properties

For each adapter definition, there are basic properties and run-time database information that must be defined. This includes specifying user-defined adapter properties.

4.1 Specifying Adapter Definition Properties

When setting up basic and necessary information for the adapter definition, the requirements are organized in five main tabs:

- **Adapter:** Specify basic descriptive information for the adapter. For more information, see Section 4.2, “Adapter Definition Properties,” on page 33.
- **Icon:** Specify an icon for the adapter from the Operations Center icon library or select a custom icon. For more information, see Section 4.3, “Assigning an Adapter Icon,” on page 35.
- **Properties:** Specify defaults for the run-time adapter properties, including the alarm columns displayed in the Operations Center console, and the adapter hierarchy file and stylesheet file. For more information, see Section 4.4, “Specifying Run-Time Adapter Properties,” on page 36.
- **Scripts:** Set up defaults for scripts that should run based on adapter activity. For more information, see Section 4.5, “Defining Script Triggers,” on page 38.
- **Database:** Specify the database connection that is used by the adapter at run-time. For more information, see Section 4.6, “Setting a Run-Time Database Connection,” on page 39.

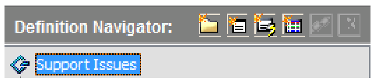
If you change adapter properties after deploying an adapter, it is necessary to stop and delete the adapter instance, undeploy the adapter, then redeploy the adapter and create an adapter instance.

4.2 Adapter Definition Properties

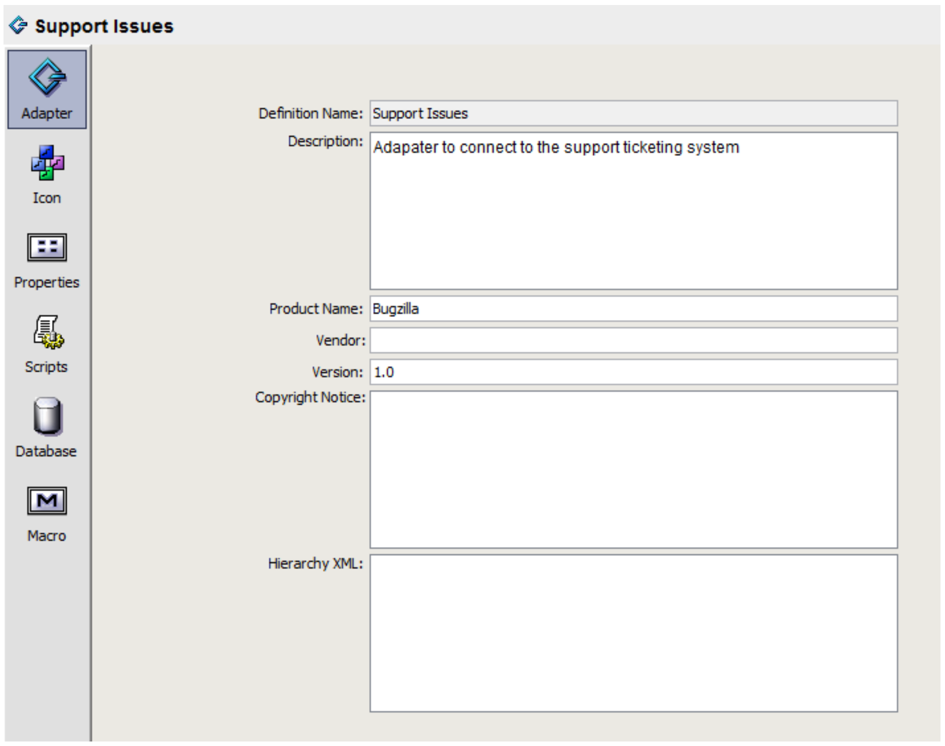
Systems integrators or OEMs can use Data Integrator to create customized adapters for deployment to multiple clients. Information in the adapter properties can describe the deployable product. For internal use, the adapter properties could be used as part of an organization’s change control mechanism.

To specify basic properties for the adapter definition:

1 Select the adapter definition root element in the *Definition Navigator* pane.



2 The *Adapter* icon is selected by default and the *Definition Properties* pane shows basic properties for the adapter definition.



Definition Name: Support Issues
Description: Adapater to connect to the support ticketing system
Product Name: Bugzilla
Vendor:
Version: 1.0
Copyright Notice:
Hierarchy XML:

3 Fill in the fields:

Definition Name: The name of the deployed adapter.

Description: A brief text description of the adapter definition.

Product Name: The name of the related product used to gather the data that is access by the adapter.

Vendor: The vendor name of the specified product.

Version: The version number of the specified product.

Copyright Notice: The related copyright information for the adapter definition.

4 Enter XML in the *Hierarchy XML* field.

The Hierarchy XML text area provides support for MODL (Managed Objects Definition Language) `<param>` tags, custom properties, and other tags that are valid with a MODL `<generator>` tag. You should know the XML syntax for what you want to accomplish.

MODL is an XML-based markup language used to create HierarchyFiles for Operations Center. The HierarchyFile reflects both the nature of information received from a management system and the processing logic of the Operations Center system.

For more information about MODL and the HierarchyFile, see "Using the HierarchyFile" in the *Operations Center Adapter and Integration Guide*.

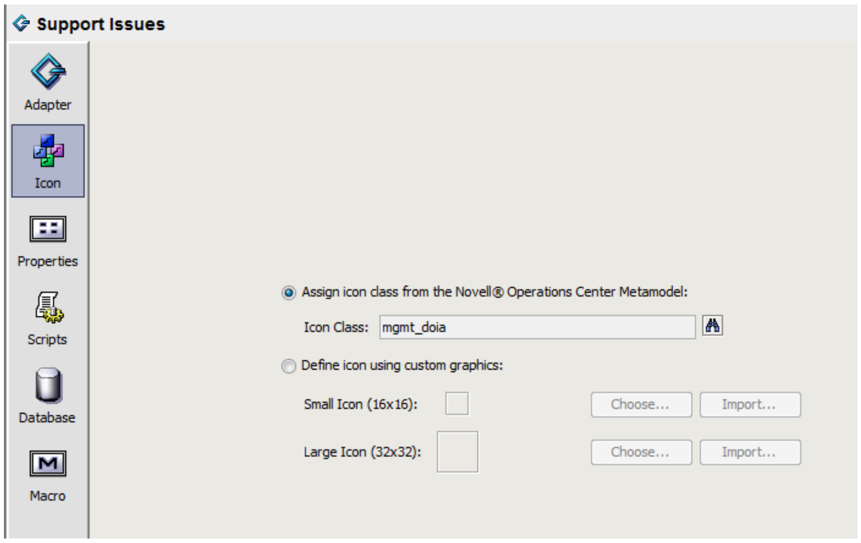
4.3 Assigning an Adapter Icon

The icon specified at the adapter definition level is associated with the adapter element displayed under *Adapters* in the Operations Center console. Select an adapter-related icon from the *Operations Center Metamodel* icon class browser.

To specify an icon for an adapter:

1. Select the root element in the *Definition Navigator* pane.

   If you have just created the adapter, it is already selected.
2. In the right pane, click *Icon* to define icon properties:



Assign icon class from the Novell® Operations Center Metamodel: [✓]

Icon Class: mgmt_doia

Define icon using custom graphics:

Small Icon (16x16):

Large Icon (32x32):

3. Select a radio button:

   Assign icon class from the Operations Center Metamodel: Uses an icon class from the Operations Center Metamodel browser.

   Next step: Click *Browse Class*, then select an icon class.

   Define icon using custom graphics: Enables specifying custom small and large icon graphics.

   Next step: Click *Choose* or *Import* to select the graphic file for *Small Icon* and *Large Icon*. Imported graphics must be in a GIF or JPEG format. Click *Choose* only if you have previously saved an imported graphic.

TIP: When defining custom graphics, set *Small Icon* to 16 x 16 pixels for best viewing results. Set *Large Icon* to 32 x 32 pixels for best viewing results.

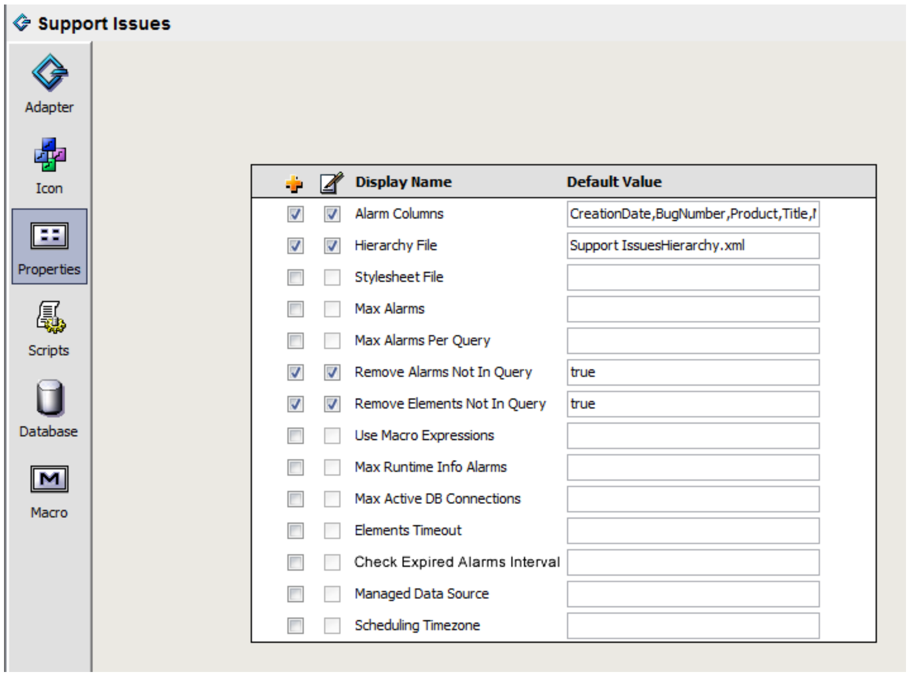
4.4 Specifying Run-Time Adapter Properties

Set default adapter property values through the adapter definition. You can also activate and display/hide specific properties.

IMPORTANT: Do not manually edit the hierarchy for Data Integrator definitions. Always use the Data Integrator Editor. Assume a defined Data Integrator adapter has a custom `Hierarchy.XML` file specified in the adapter properties. If changes are made to the Data Integrator definition and the definition is redeployed, Data Integrator overwrites the existing hierarchy file.

To define run-time adapter properties:

1. Select the adapter definition element in the *Definition Navigator* pane.
2. In the right pane, click the *Properties* icon to define run-time adapter properties:



		Display Name	Default Value
✓	✓	Alarm Columns	CreationDate,BugNumber,Product,Title,
✓	✓	Hierarchy File	Support IssuesHierarchy.xml
☐	☐	Stylesheet File	
☐	☐	Max Alarms	
☐	☐	Max Alarms Per Query	
✓	✓	Remove Alarms Not In Query	true
✓	✓	Remove Elements Not In Query	true
☐	☐	Use Macro Expressions	
☐	☐	Max Runtime Info Alarms	
☐	☐	Max Active DB Connections	
☐	☐	Elements Timeout	
☐	☐	Check Expired Alarms Interval	
☐	☐	Managed Data Source	
☐	☐	Scheduling Timezone	

3. Select the (*Activate Property*) check box to enable the property for the adapter.
4. Select the (*Add Property*) check box beside the property to display the property in the run-time adapter properties.
5. Enter default values for the properties where applicable:

   Alarm Columns: The default alarm columns displayed in the *Alarms* view. Specify using the Data Integrator property names or labels.

   To customize the alarm column name used in Operations Center when deployed, specify as `aliasname=propertyName`. Column aliasing is not available on the following alarms columns: `Severity`, `Element`, `Date/Time`, and `ID`.

   Hierarchy File: A file in the `/`*`OperationsCenter_install_path`*`/database` directory that contains an XML description of the element hierarchy to build below the adapter element.

   Stylesheet File: The file in the `/`*`OperationsCenter_install_path`*`/database` directory that applies to the HierarchyFile as a style markup to produce the final output.

Max Alarms: The maximum number of alarms allowed for the adapter. After reaching the maximum number, the adapter ages out the oldest alarms based on the values set. Enter -1 to allow an unlimited number of alarms for the run-time adapter.

Max Alarms Per Query: The maximum number of alarms allowed per query. After reaching the maximum number, the adapter ages out the oldest alarms based on the values set. Enter -1 to allow an unlimited number of alarms for the run-time adapter.

Remove Alarms Not In Query: Determines what happens to alarms not selected by a query. Set to True to automatically delete alarms in the run-time adapter after deleting them from the database.

Remove Elements Not In Query: Determines what happens to elements not selected by a query. Set to True to automatically delete elements in the run-time adapter after deleting them from the database. This property cannot be used in conjunction with delta queries.

All child objects of a deleted element are deleted, even if data is available for the child elements. If subsequent child queries for the element return values after the parent element has been removed, then a dummy parent element is created for those instances.

Use Macro Expressions: If True, enables macro expression query preprocessing. It is disabled by default. Macro expression query preprocessing is available in Data Integrator for added flexibility in creating query text. For more information, see Section 6.7, “Using Macro Expression Query Preprocessing,” on page 69.

Max Runtime Info Alarms: The maximum number of runtime alarms. Enter 0 to disable runtime alarms for the run-time adapter. Alarms are generated for such actions as:

- Running and finishing schedules
- Starting and completing queries
- Having an error condition that is internal to the adapter, such as a query failed

Max Active DB Connections: The maximum number of concurrent JDBC connections allowed open at one time. After reaching the limit, it blocks additional queries until another query ends and frees a connection.

In addition to the maximum JDBC connections specified, an additional connection is automatically established for standard health check processes. For example, if *Max Active DB Connections* is set to 1, there will be 2 connections established.

Elements Timeout: The length of time, in seconds, to age out elements. If no open alarms exist and the element’s condition does not change in n seconds, and the element has no children, then the element disappears. The element redisplays if another alarm is generated. The default is 300 seconds.

- **AgeOutTime < 0:** Never age out.
- **AgeOutTime = 0:** Age out immediately.
- **AgeOutTime > 0:** Age out after specified time expires.

Check Expired Alarms Interval: The length of time, in minutes, to wait to check for expired alarms. When an expired alarm is found, based on it’s *alarmDateToExpire* value, it is deleted from the server. Map the *alarmDateToExpire* property to an appropriate database field in the Alarm Query definition (see Section 6.2.2, “Mapping Element or Alarm Properties,” on page 60).

Managed Data Source: Determines how to handle elements and alarms if database connection is lost. Set to True to remove alarms and change element conditions to UNKNOWN (the integration “hollows”).

Set to False to leave unchanged alarm and element status.

Scheduling Timezone: Specifies a time zone to use for an adapter's scheduled queries. Valid entries are:

- US/*zone_name* (Eastern, Pacific, Central, Mountain)
- Europe/*city_name* (such as London)
- GMT *hour* (such as 06:00, 07:00)
- UTC (coordinated universal time)

Scheduling Timezone does not apply to schedules that are selected to use UTC time on a query's Scheduling page. Those schedules are still relative to UTC. For more information, see Section 6.10, "Scheduling Queries," on page 73.

If this property is not enabled or is blank, the Operations Center server's time zone is used for scheduled queries.

4.5 Defining Script Triggers

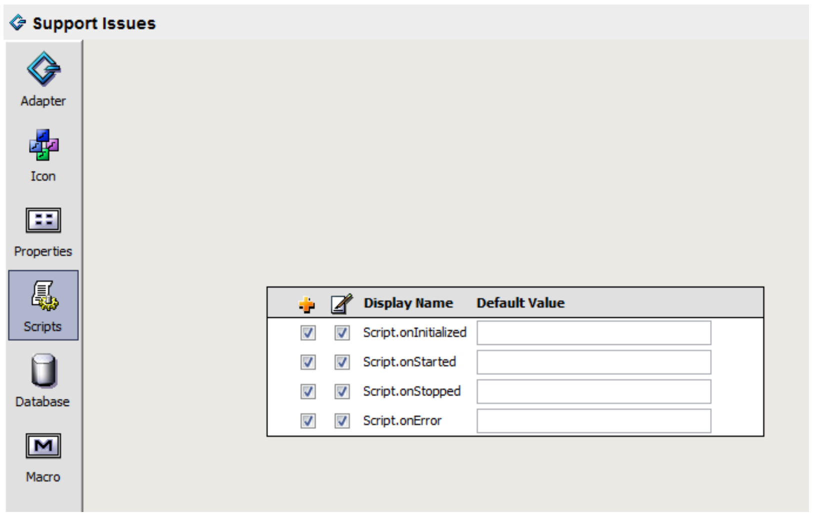
Various scripts might be prespecified for the user through the adapter definition. This provides default values for creating an adapter with the deployed adapter definition. Specify script triggers as visible or hidden to the user.

Script triggers are available as adapter properties upon adapter creation and later as editable values.

To define script triggers:

1 Select the definition root element in the *Definition Navigator* pane.

2 In the right pane, click the *Scripts* icon:



✚	✎	Display Name	Default Value
✓	✓	Script.onInitialized	
✓	✓	Script.onStarted	
✓	✓	Script.onStopped	
✓	✓	Script.onError	

3 For each script trigger, select the associated ✚ (*Activate Property*) check box.

4 (Optional) Select the *Add to Adapter Property Page* ✎ check box to display the script property in the run-time adapter properties.

If it is deselected, the property is active, but not visible to users.

5 Enter a script file for the *Default Value*.

4.6 Setting a Run-Time Database Connection

The database information set for the run-time adapter could be different from the database schema used to create this adapter definition. Because the Data Integrator tool is a development tool, it is likely that a development database is used to set up all definitions for the adapter. But, for run-time adapters, point users to a production database (of the same database type with the same set of tables). Specify database settings as visible or hidden to the user.

The Data Integrator never writes to the database. It sets a read-only JDBC connection and performs SQL select statements.

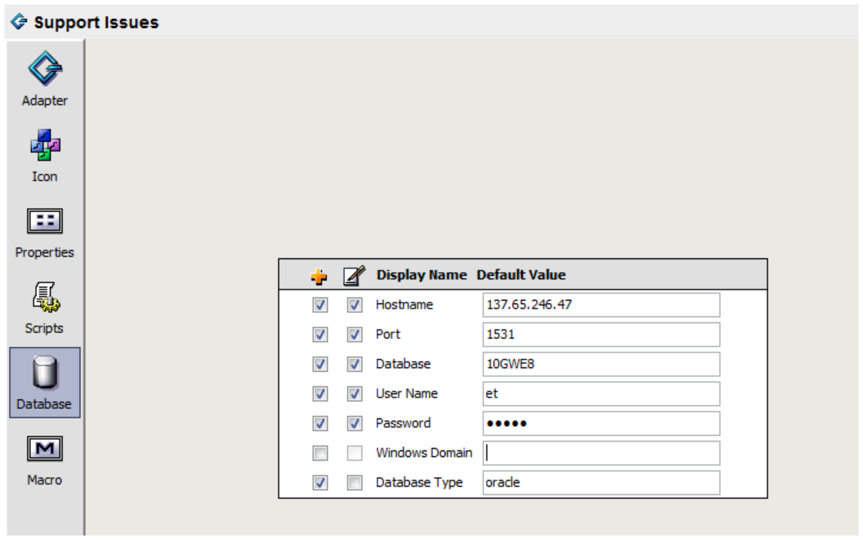
WARNING: The run-time database must be the same type and have the same schema as the development data source.

To set up run-time database information:

1 Select the root element in the *Definition Navigator* pane.

If the adapter was just created, it is already selected.

2 Click the *Database* icon:



		Display Name	Default Value
✓	✓	Hostname	137.65.246.47
✓	✓	Port	1531
✓	✓	Database	10GWE8
✓	✓	User Name	et
✓	✓	Password	●●●●●
		Windows Domain	
✓		Database Type	oracle

3 Define database values for the deployed adapter at run-time:

Hostname: The database server name.

Port: The port number where the database listens for communications.

Database: The database name (when defining a SQL Server, Sybase, or DB2 database connection).

Server ID: The database name (when defining an Oracle database connection).

User Name: The name of the user account.

Password: The password associated with the user name.

Windows Domain: The name of the Windows domain to use for Windows native authentication. Applies to JTDS and SQL Server connections only.

Database Type: The type of database (Oracle, SQL Server, Sybase, and so on).

5 Defining Groups, Elements and Alarms

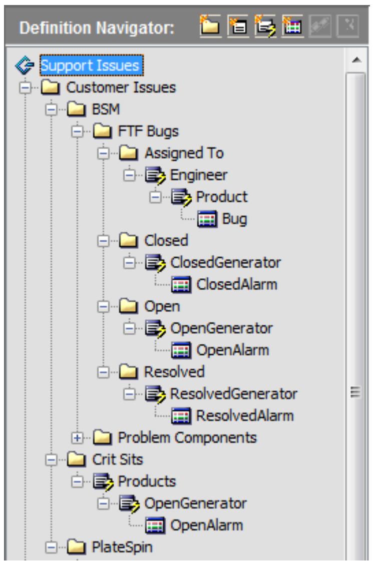
After defining the properties of the adapter definition, the next step is to organize and define the structure of incoming database information using groups, elements, and alarms. During the process, set up queries, which inform the adapter how to extract information from a database to create the elements and alarms, as well as their properties.

5.1 Understanding Groups, Elements and Alarms

Groups, elements, and alarms display in a hierarchy in the *Definition Navigator* pane of the Data Integrator Definition Editor.

Figure 5-1 illustrates element and alarm definitions displaying for the *Support Issues* adapter definition:

Figure 5-1 *Definition Navigator Pane*



In addition to the basic folder (*Group Element*), Table 5-1 explains the three definitions that are used to create the element structure for the adapter and feed it additional data through element alarms:

***Table 5-1** Types of Element and Alarm Definitions*

Icon	Type	Creates…
📁	Group	A single folder in the Operations Center hierarchy tree when an adapter is created from the deployed definition.
▤	Database Element	One or more elements (generated by a database query) in the Operations Center hierarchy when an adapter is created from the deployed definition. Use this element definition when a more complex query is needed to filter the data to create specific elements. These elements have property information defined about them and can have custom property pages.
▤	Dynamic Groups (Generator)	One or more elements (generated directly from a database table column) in the Operations Center hierarchy tree when an adapter is created from the deployed definition. Use this element definition type to create an element directly representing each of the column values. There is no other information or properties defined for these elements.
▤	Alarm	Alarms or events (generated by a database query) for the associated elements in the Operations Center hierarchy when an adapter is created from the deployed definition.

In the *Definition Navigator* pane, each definition created for the adapter definition is displayed with the associated icon for the definition type. Browsing the adapter definition hierarchy provides an overview of the types of elements and associated alarms.

5.2 Defining Groups

Group Element definitions create static folder objects within the adapter element hierarchy. These folders typically act as parent containers to Database Elements or Defining Dynamic Groups (Generator Elements).

When setting up a Group Element Definition, the requirements are organized into two main tabs:

- **Group:** Specify basic descriptive information for the folder.
- **Icon:** Specify an icon for the group element from the Operations Center Metamodel icon class browser or select a custom icon. For more information, see Section 5.7, “Assigning Icons to Groups, Dynamic Groups, and Database Elements,” on page 53.

To define groups, following the instructions in these sections:

- Section 5.2.1, “Creating a Group,” on page 42
- Section 5.2.2, “Specifying Basic Properties for a Group,” on page 43
- Section 5.2.3, “Specifying a Default Algorithm,” on page 44

5.2.1 Creating a Group

1. In the *Definition Navigator* pane, select the parent element of the new group.
2. Click 📁 (*New Group*).

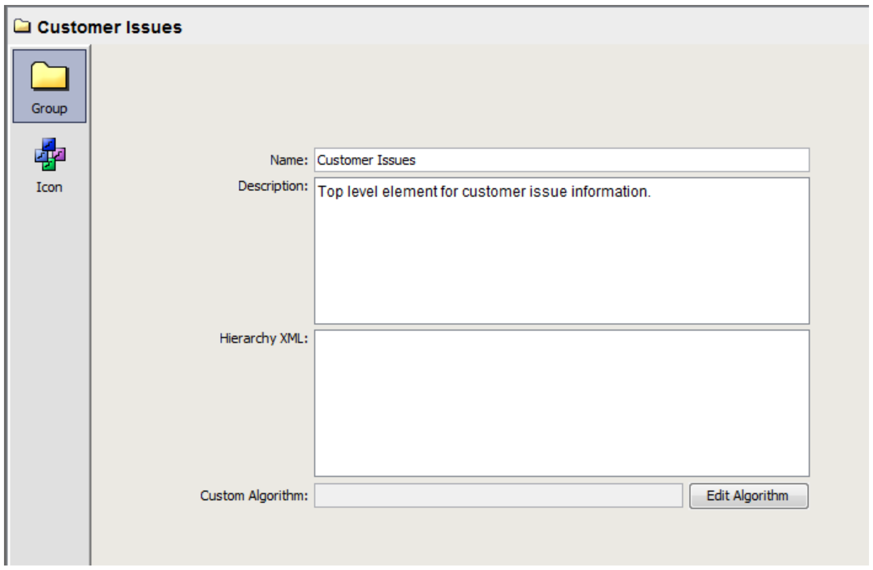
   A group folder is created under the selected element.

5.2.2 Specifying Basic Properties for a Group

1 Select the *Group* element in the *Definition Navigator* pane.

If you just created the group, it is already selected.

In the right pane, *Group* is selected by default and the *Definition Properties* pane shows basic properties for the group:


Customer Issues

Name: Customer Issues

Description: Top level element for customer issue information.

Hierarchy XML:

Custom Algorithm:

2 Specify a name and description for the group.

3 Enter XML in the *Hierarchy XML* field.

The Hierarchy XML text area provides support for MODL (Managed Objects Definition Language) `<param>` tags, custom properties, and other tags that are valid with a MODL `<generator>` tag. You should know the XML syntax for what you want to accomplish.

MODL is an XML-based markup language used to create HierarchyFiles for Operations Center. The HierarchyFile reflects both the nature of information received from a management system and the processing logic of the Operations Center system.

For more information about MODL and the HierarchyFile, see "Using the HierarchyFile"in the *Operations Center Adapter and Integration Guide*.

4 Specify a default algorithm for the group in the *Custom Algorithm* field.

For more information, see the set of steps below.

5 Specify an icon.

5.2.3 Specifying a Default Algorithm

To specify a default algorithm for the group in the *Custom Algorithm* field:

1. In the group *Definition Properties* pane, click *Edit Algorithm* to open the Edit Algorithm dialog box.
2. In the Edit Algorithm dialog box, select the *Use the Default Algorithm for This Element* radio button to specify no custom algorithm.

   None displays in the *Custom Algorithm* field in the group *Definition Properties* pane.
3. To set a custom algorithm, select the *Select an Alternate Algorithm* radio button, then select an algorithm type from the drop-down list.
4. Click *OK* to close the Edit Algorithm dialog box.

   For information on algorithms, see *Using Algorithms to Calculate Element State* in the *Operations Center Server Configuration Guide*.

5.3 Defining Dynamic Groups (Generator Elements)

Dynamic groups (generator elements) are created directly from database columns. Use Generator elements to represent data from the database when no additional information about them is available or required.

For example, define a Generator element for every ticket number in a trouble ticket table when there is little need for additional alarm properties or element information.

As database information changes and is re-queried by the run-time adapter, these elements are deleted or added as column data changes.

When setting up a Generated elements definition, the requirements are organized into two main tabs:

- **Generator:** Specify basic descriptive information for the Generator element definition. Specify the database field that pulls unique values to produce each element.
- **Icon:** Specify an icon for the Generator elements from the Operations Center Metamodel icon class browser or select custom icon. For more information, see Section 5.7, “Assigning Icons to Groups, Dynamic Groups, and Database Elements,” on page 53.

To define dynamic groups, following the instructions in these sections:

- Section 5.3.1, “Creating a Generator,” on page 44
- Section 5.3.2, “Specifying Basic Properties for a Generator Element,” on page 45
- Section 5.3.3, “Specifying a Default Algorithm,” on page 46

5.3.1 Creating a Generator

1. In the *Definition Navigator* pane, select the parent element of the new generator.
2. Click (*Create a Generated Element*).

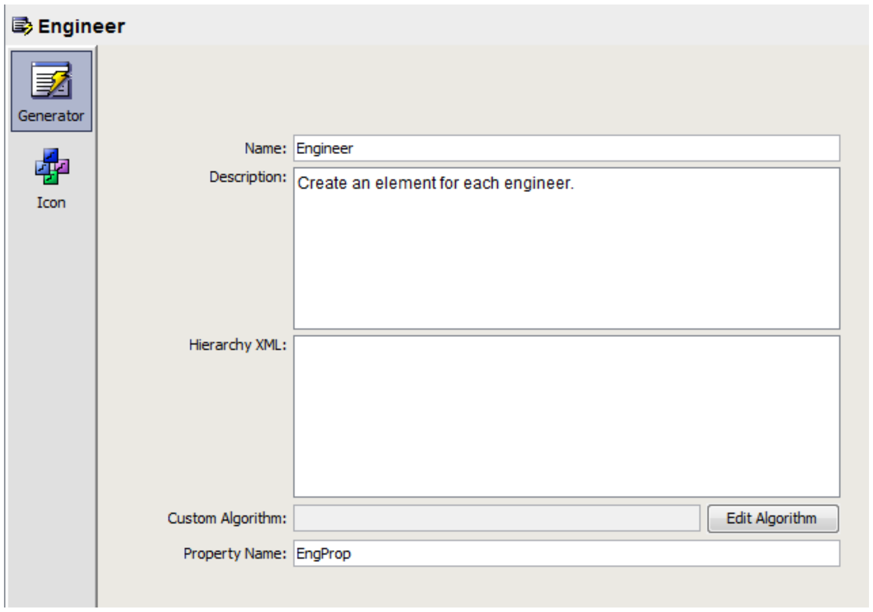
   A Generator element definition is created under the element in the *Definition Navigator* pane.

5.3.2 Specifying Basic Properties for a Generator Element

It is possible to create a parent/child hierarchy directly from the database columns. For example, values from the first column can create parent elements, while the unique values from another column create the child elements.

To specify basic properties for a Generator element:

1. Select the Generator element in the *Definition Navigator* pane.
2. In the left pane, *Generator* is selected by default and the *Definition Properties* pane shows basic properties for the definition:



Name: Engineer
Description: Create an element for each engineer.
Hierarchy XML:
Custom Algorithm:
Property Name: EngProp

3. Use the *Fetch Rows* feature at the bottom of the *Database Navigator* pane to run a quick sampling of database information and verify that the correct table column is used to drive a Generator element.
4. Specify the *Name* and *Description* for the definition (used only in the Definition Editor for the definition name).
5. Enter XML in the *Hierarchy XML* field.

   The Hierarchy XML text area provides support for MODL (Managed Objects Definition Language) `<param>` tags, custom properties, and other tags that are valid with a MODL `<generator>` tag. You should know the XML syntax for what you want to accomplish.

   MODL is an XML-based markup language used to create HierarchyFiles for Operations Center. The HierarchyFile reflects both the nature of information received from a management system and the processing logic of the Operations Center system.

   For more information about MODL and the HierarchyFile, see “Using the HierarchyFile”in the *Operations Center Adapter and Integration Guide*.
6. Specify a default algorithm for the group in the *Custom Algorithm* field.

   For more information, see the following set of steps.
7. In the *Property Name* field, specify the database column whose values will drive the creation of these elements.

8 Specify a database column name or drag and drop a column name from the *Database Navigator* field to the *Property Name* field.

   The query Property ID set for an alarm or database element definition that is a child definition of a Generator element must match exactly the Property Name specified for the Generator element. If these properties do not match, the elements are not created at run-time.

9 Specify an icon.

5.3.3 Specifying a Default Algorithm

To specify a default algorithm for the group in the *Custom Algorithm* field:

1 In the group *Definition Properties* pane, click *Edit Algorithm* to open the Edit Algorithm dialog box.

2 In the Edit Algorithm dialog box, select the *Use the Default Algorithm for This Element* radio button to specify no custom algorithm.

   None displays in the *Custom Algorithm* field in the group *Definition Properties* pane.

3 To set a custom algorithm, select the *Select an Alternate Algorithm* radio button, then select an algorithm type from the drop-down list.

4 Click *OK* to close the Edit Algorithm dialog box.

   For information on algorithms, see *Using Algorithms to Calculate Element State* in the *Operations Center Server Configuration Guide*.

5.4 Defining Elements from Database Information

Database element definitions tell the deployed adapter definition how and where to create elements that are based on information pulled from a database. Use the DBElement feature to create robust relationship modeling.

- Section 5.4.1, "Understanding Element Organization," on page 46
- Section 5.4.2, "Creating an Element for Incoming Database Information," on page 47
- Section 5.4.3, "Specifying Basic Properties," on page 47
- Section 5.4.4, "Specifying a Default Algorithm," on page 48
- Section 5.4.5, "Implementing the Definition," on page 48
- Section 5.4.6, "Ensuring Adapter Element State Propagation," on page 49

5.4.1 Understanding Element Organization

When creating an element definition, the requirements are organized in five main tabs:

- **Element:** Specify basic descriptive information for the element.
- **Icon:** Specify an icon to represent the element. Make a selection from the Operations Center Metamodel icon class browser or specify a custom icon.

  For more information, see Section 5.7, "Assigning Icons to Groups, Dynamic Groups, and Database Elements," on page 53.

- **Query:** Define a query to gather and filter database information to create the element and its properties.

  For more information, see Section 6.1, "Defining a Query," on page 57 and Section 6.9, "Testing Queries," on page 73.
- **Schedule:** Set up scheduling requirements for running the element query and checking for new information at adapter runtime.

  For more information, see Section 6.10, "Scheduling Queries," on page 73.
- **Pages:** (Optional) Create custom property pages for the element.

  For more information, see Section 5.9, "Defining Custom Property Pages," on page 54.

If multiple element layers are created by different database queries that run on different schedules, the adapter might need to create a temporary parent element for the child information until the query for the parent element runs and updates. This occurs in cases where the database query might be run more frequently for the child elements because information affecting the parents is changed less often.

5.4.2 Creating an Element for Incoming Database Information

1. In the *Definition Navigator* pane, select the parent element of the new element.
2. Right-click the parent element, then select *New Database Element*, or click (*Create a Database Element*).

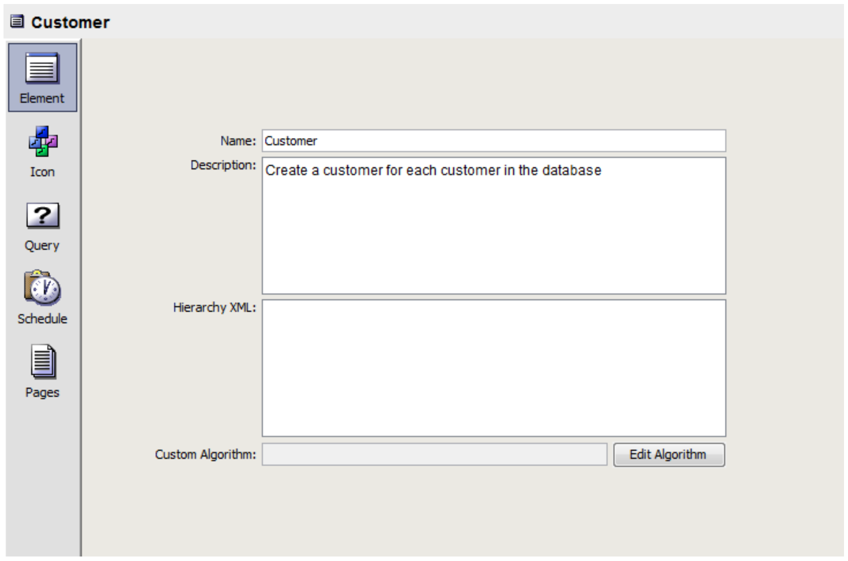
   An element is created under the parent element.

5.4.3 Specifying Basic Properties

To specify basic properties for database elements:

1. Click the new element in the *Definition Navigator* pane.
2. In the right pane, the *Element* icon is selected by default.

   The *Definition Properties* pane shows basic properties for the database element:



Name: Customer
Description: Create a customer for each customer in the database
Hierarchy XML:
Custom Algorithm:

3. Specify the Name and Description for the database element.

   This information is for definition purposes and is not displayed when the adapter is deployed and created in Operations Center.

   To simplify the element distinguished name definition hierarchy, see Section 6.2.5, "Using the HierarchyKey Property to Simplify Element Naming," on page 64.

4. Enter XML in the *Hierarchy XML* field.

   The Hierarchy XML text area provides support for MODL (Managed Objects Definition Language) `<param>` tags, custom properties, and other tags that are valid with a MODL `<generator>` tag. You should know the XML syntax for what you want to accomplish.

   MODL is an XML-based markup language used to create HierarchyFiles for Operations Center. The HierarchyFile reflects both the nature of information received from a management system and the processing logic of the Operations Center system.

   For more information about MODL and the HierarchyFile, see "Using the HierarchyFile" in the *Operations Center Adapter and Integration Guide*.

5. Specify a default algorithm for the group in the *Custom Algorithm* field.

   For more information, see the steps below.

5.4.4 Specifying a Default Algorithm

To specify a default algorithm in the *Custom Algorithm* field:

1. In the *Definition Properties* pane, click *Edit Algorithm* to open the Edit Algorithm dialog box.
2. In the Edit Algorithm dialog box, select the *Use the Default Algorithm for This Element* radio button to specify no custom algorithm.

   None displays in the *Custom Algorithm* field in the group *Definition Properties* pane.

3. To set a custom algorithm, select the *Select an Alternate Algorithm* radio button, then select an algorithm type from the drop-down list.
4. Click *OK* to close the Edit Algorithm dialog box.

   For information on algorithms, see *Using Algorithms to Calculate Element State* in the *Operations Center Server Configuration Guide*.

5.4.5 Implementing the Definition

Follow these steps to implement the element definition:

1. Specify an icon.

   For instructions, see Section 4.3, "Assigning an Adapter Icon," on page 35.

2. Define and test a query.

   For instructions, see Section 6.1, "Defining a Query," on page 57 and Section 6.9, "Testing Queries," on page 73.

3. Select a schedule for the query.

   For instructions, see Section 6.10, "Scheduling Queries," on page 73.

4. (Optional) Define property pages for the database element and define the properties they contain at adapter runtime.

   For instructions, see Section 5.9, "Defining Custom Property Pages," on page 54.

5.4.6 Ensuring Adapter Element State Propagation

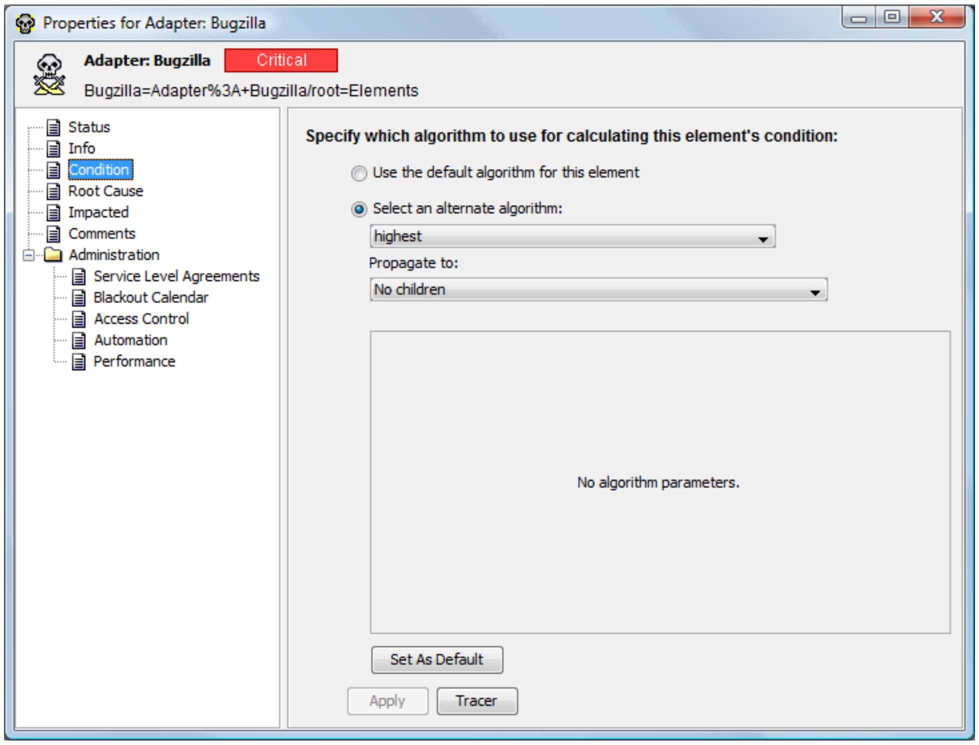
In the Data Integrator adapter element hierarchy, there is an option to have the child element state propagate up the hierarchy and change the state of its parent, which is the Data Integrator adapter element.

For example, if the adapter element is UNKNOWN, but its child is OK, you might want the OK state to propagate up to the parent adapter element.

To ensure state propagation from child to parent, start the Data Integrator adapter and set the top-most element to the highest algorithm. When the elements are recalculated, the parent element takes the highest child condition.

To set the element to the highest algorithm:

1 In the Operations Center *Explorer* pane, under *Elements*, right-click the adapter root element, then select *Properties* to open the Status property page:



Specify which algorithm to use for calculating this element's condition:

Use the default algorithm for this element

[✓ Select an alternate algorithm:]

highest

Propagate to:

No children

No algorithm parameters.

2 In the left pane, click *Condition* to update the property page.

3 Select the *Select an Alternate Algorithm* radio button.

4 Select *Highest* from the drop-down list.

5 Click *Apply*.

5.5 Creating Alarms

Alarm definitions tell the deployed adapter definition how and where to list information as alarms.

- Section 5.5.1, “Understanding Alarm Requirements Organization,” on page 50
- Section 5.5.2, “Creating an Alarm Definition,” on page 50

- Section 5.5.3, “Specifying Basic Properties,” on page 51
- Section 5.5.4, “Next Actions to Perform,” on page 51

5.5.1 Understanding Alarm Requirements Organization

When defining an alarm definition, the requirements are organized in four main tabs:

- **Alarm:** Specify basic descriptive information for the alarm.
- **Query:** Define a query to gather and filter database information to create the alarm properties. For more information, see Section 6.1, “Defining a Query,” on page 57 and Section 6.9, “Testing Queries,” on page 73.
- **Schedule:** Set up scheduling requirements for running the definition’s query and checking for new information at adapter runtime. For more information, see Section 6.10, “Scheduling Queries,” on page 73.
- **Pages:** Create alarm property pages and define the properties they contain at adapter runtime. For more information, see Section 5.9, “Defining Custom Property Pages,” on page 54.

5.5.2 Creating an Alarm Definition

To create an alarm definition:

1. In the *Definition Navigator* pane, select the parent element of the new alarm definition.
2. Right-click the parent element, then select *New Alarm* or click (*Create an Alarm Definition*).

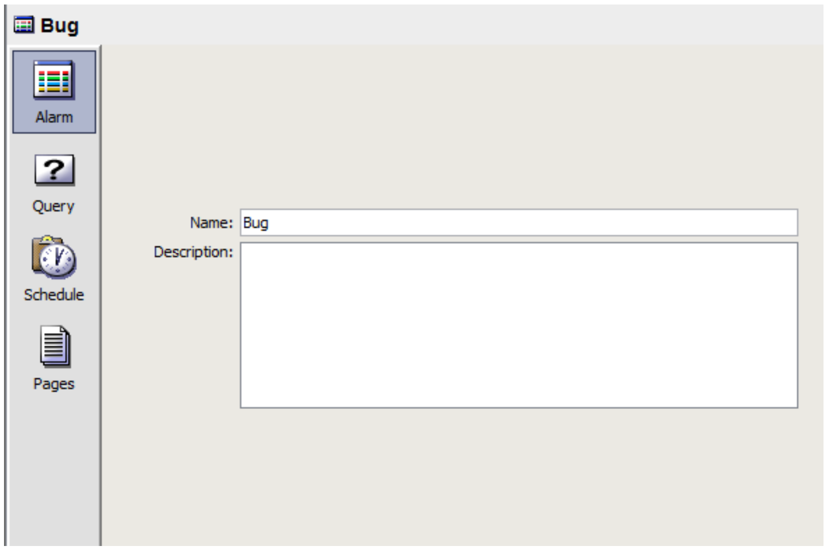
   An alarm definition is created under the element.

5.5.3 Specifying Basic Properties

To specify basic properties for an Alarm Definition:

1. Select the alarm definition in the *Definition Navigator* pane.
2. In the right pane, the *Alarm* icon is selected by default.

   The *Definition Properties* pane shows basic properties for the alarm definition:



Name: Bug
Description:

3. Specify the Name and Description for the Alarm Definition (this information is used only by the Definition Editor).

5.5.4 Next Actions to Perform

1. Specify an icon.

   For instructions, see Section 4.3, "Assigning an Adapter Icon," on page 35.
2. Define and test a query.

   For instructions, see Section 6.1, "Defining a Query," on page 57 and Section 6.9, "Testing Queries," on page 73.
3. Pick a schedule for the query.

   For instructions, see Section 6.10, "Scheduling Queries," on page 73.
4. Define property pages.

   For instructions, see Section 5.9, "Defining Custom Property Pages," on page 54.

5.6 Creating Relationships

Relationship definitions can only be created under a Database Element and tell the deployed adapter definition how and where relationships are created between elements.

5.6.1 Understanding Relationships Requirements Organization

When defining a relationship definition, the requirements are organized in three main tabs:

- **Relationship:** Specify basic descriptive information and target area for the relationship.
- **Query:** Define a query to gather and filter database information to create the relationship connection. For more information, see Section 6.1, “Defining a Query,” on page 57 and Section 6.9, “Testing Queries,” on page 73.
- **Schedule:** Set up scheduling requirements for running the definition’s query and checking for new information at adapter runtime. For more information, see Section 6.10, “Scheduling Queries,” on page 73.

5.6.2 Creating a Relationship Definition

To create a relationship definition:

1. In the *Definition Navigator* pane, select the Database Element that is the parent element of the new relationship definition.
2. Right-click the Database Element, then select *New Relationship* or click (*Create a Relationship Definition*).

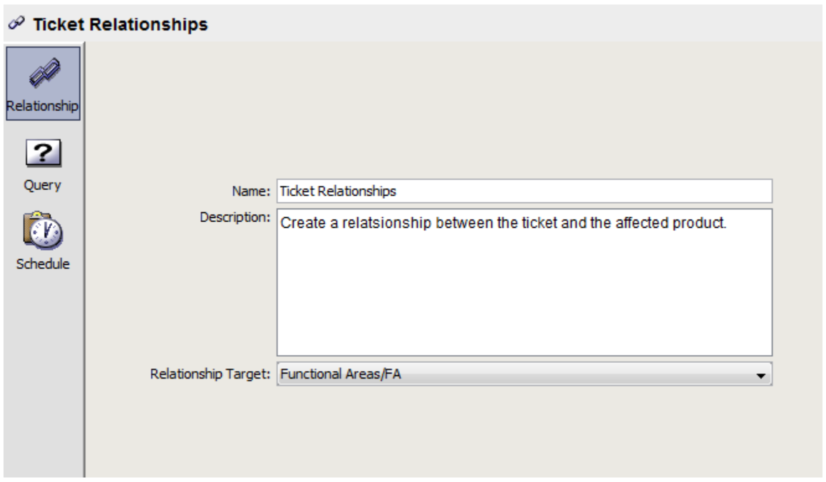
   A relationship definition is created under the element.

5.6.3 Specifying Basic Properties

To specify basic properties for a Relationship Definition:

1. Select the relationship definition in the *Definition Navigator* pane.
2. In the right pane, the *Relationship* icon is selected by default.

   The *Definition Properties* pane shows basic properties for the alarm definition:



Name: Ticket Relationships
Description: Create a relatsionship between the ticket and the affected product.
Relationship Target: Functional Areas/FA

3. Specify the Name and Description for the Relationship Definition (this information is used only by the Definition Editor).
4. Select a target area by selecting a folder and database element combination from the *Relationship Target* drop-down list.

5.6.4 Next Actions to Perform

1 Define and test a query.

For instructions, see Section 6.1, “Defining a Query,” on page 57 and Section 6.9, “Testing Queries,” on page 73.

2 Pick a schedule for the query.

For instructions, see Section 6.10, “Scheduling Queries,” on page 73.

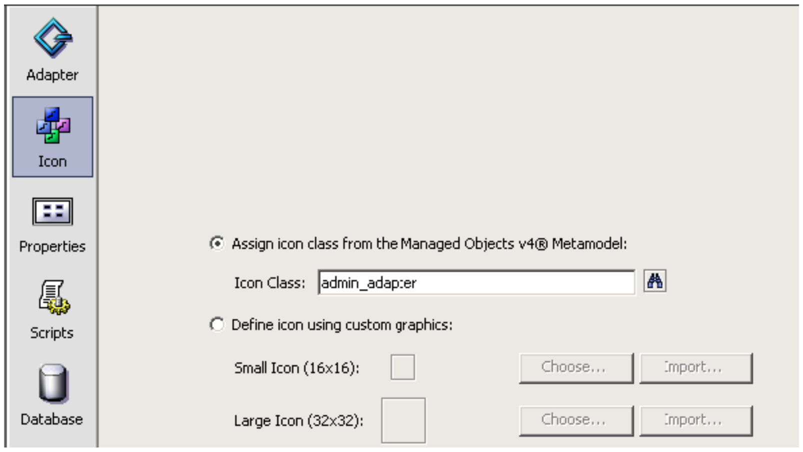
5.7 Assigning Icons to Groups, Dynamic Groups, and Database Elements

For all element-related (Group, Database Element, and Generator) definitions, there is the opportunity to specify a unique icon that is leveraged at adapter runtime. The icon can be selected directly from the Operations Center Icon Library, or custom icons can be used.

To specify an icon for a group or element definition:

1 Select the element in the *Definition Navigator* pane.

2 In the right pane, click *Icon* to define icon properties:



[✓ Assign icon class from the Managed Objects v4® Metamodel:]

Icon Class: admin_adapter

Define icon using custom graphics:

Small Icon (16x16):

Large Icon (32x32):

3 Select one of the following radio buttons:

Assign icon class from the Operations Center Metamodel: Uses an icon class from the Operations Center Metamodel browser.

Next step: Click *Browse Class*, then select an icon class. You can also specify the class name.

Define icon using custom graphics: Enables specifying custom small and large icon graphics.

Next step: Click *Choose* or *Import* to select the graphic file for *Small Icon* and *Large Icon*. Imported graphics must be in a GIF or JPEG format. Click *Choose* only if you have previously saved an imported graphic.

TIP: When defining custom graphics, set *Small Icon* to 16 x 16 pixels for best viewing results. Set *Large Icon* to 32 x 32 pixels for best viewing results.

5.8 Working with Groups, Elements, and Alarms

Cut, *Copy*, and *Paste* options are available for groups, elements, and alarms. Copying and cutting includes all children of a selected item.

Do any of the following:

- To copy, right-click a group, element, or alarm in the *Definition Navigator* pane, then select *Copy*.

  A copy symbol ( ) displays next to the item.
- To cut, right-click a group, element, or alarm in the *Definition Navigator* pane, then select *Cut*.

  The item is placed on the Clipboard and is ready to be pasted. A cut symbol displays next to the item.
- To paste, right-click a group, element, or alarm, then select *Paste*.

  The item in the Clipboard is pasted as a child of the selected group, element, or alarm.

5.9 Defining Custom Property Pages

Database Element and Alarm definitions have options to define one or more custom property pages. Properties available in the *Query Results* pane's *Properties* tab are also available for inclusion in an element or alarm custom property pages.

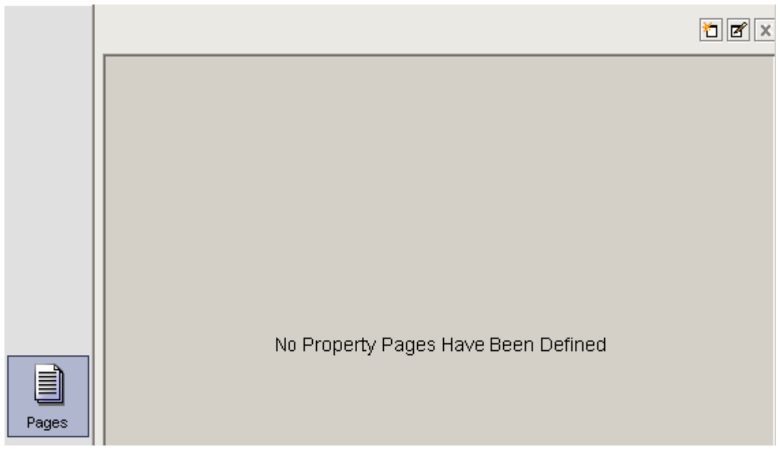
- Section 5.9.1, "Defining a Custom Property Page," on page 54
- Section 5.9.2, "Renaming a Property Page," on page 55
- Section 5.9.3, "Adding Properties to the Page," on page 55
- Section 5.9.4, "Reordering the Properties," on page 56
- Section 5.9.5, "Deleting a Property," on page 56
- Section 5.9.6, "Deleting a Property Page," on page 56

5.9.1 Defining a Custom Property Page

To define a custom property page:

1 In the *Definition Navigator* pane, select an element or alarm definition.

2 In the right pane, click the *Pages* icon:



If no pages have been defined, the tab pane is empty.

3 Click (*Add a Property Page*).

A new page displays in the *Pages* tab.

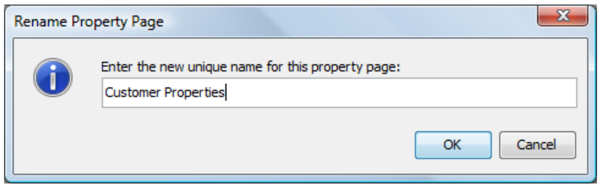
You might want to rename the new page.

5.9.2 Renaming a Property Page

To rename a custom property page:

1 Click the associated tab for the property page to display the *Property Page* tab.

2 Click (*Rename Property Page*) to open the Rename Property Page dialog box:



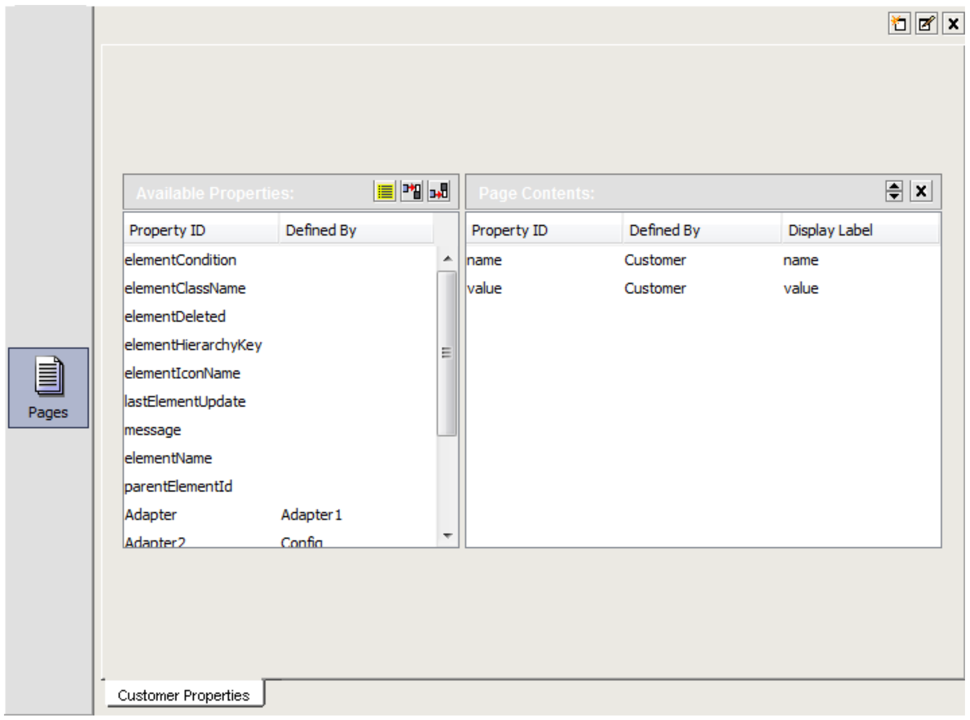
3 Specify the new name in the field.

4 Click *OK* to rename the property page.

5.9.3 Adding Properties to the Page

To add properties to a page:

1 Select the property IDs in the *Available Properties* section of the table:



Property ID	Defined By
elementCondition	
elementClassName	
elementDeleted	
elementHierarchyKey	
elementIconName	
lastElementUpdate	
message	
elementName	
parentElementId	
Adapter	Adapter1
Adapter2	Config

Property ID	Defined By	Display Label
name	Customer	name
value	Customer	value

2 Do one of the following:
   - To select all properties listed, click (*Select All*).

     All properties are highlighted.
   - To select multiple noncontiguous properties, hold the Ctrl keyboard button while clicking the properties.
   - To select multiple contiguous properties, select the first property, then hold the Shift keyboard button while clicking the last property.

     All properties in between the first and last properties are highlighted.

3 Click (*Insert Above*) or (*Insert Below*) to add the selected properties to the *Page Contents* section of the table.

4 Click the value in the associated row of the *Display Label* column and specify the name to display in the property page.

5.9.4 Reordering the Properties

To reorder the properties on a property page:

1 Select the property in the *Page Contents* section of the table.

2 To move the property up in the list, click (*Move Up*).

3 To move the property down in the list, click (*Move Down*).

5.9.5 Deleting a Property

To delete a property from the property page:

1 Select the property in the *Page Contents* section of the table.

2 In the *Page Contents* header, click (*Delete*) to remove the unwanted property.

   The property is removed.

5.9.6 Deleting a Property Page

To delete a property page:

1 Click the property page’s tab to open it inside the *Pages* tab.

2 In the upper right corner of the tab, click (*Delete*) to delete the selected page.

   The property page is removed.

6 Defining Queries and Properties

The adapter uses database queries to create elements or issue alarms. A component of the query is to map database information to element or alarm properties.

6.1 Defining a Query

The first step is to define the basic query that instructs the adapter to pull information from a specific database table or set of tables.

TIP: To retrieve data and seed the adapter upon startup or restart, define an initial query that runs once, then define a second query to retrieve new data or changes after the initial query. Otherwise, the adapter definition reads the entire database every time the adapter is started or restarted, potentially causing performance issues.

To define a query:

1. Select an element or alarm in the *Definition Navigator* pane.
2. In the right pane, click the *Query* icon to update the *Query Results* pane.

   Use the *Editor* tab to define the basic parameters of the query.

3 Under *Query Type*, do one of the following.

- To automatically generate the query source code, click the *Generated* radio button.

  The source code is then not editable in the *Source* tab.
- To enter a custom query where the query source code is editable, click the *Custom* radio button.

  The source code is editable in the *Source* tab, but the *Database Tables* and *Where* text areas are not editable in the *Editor* tab.

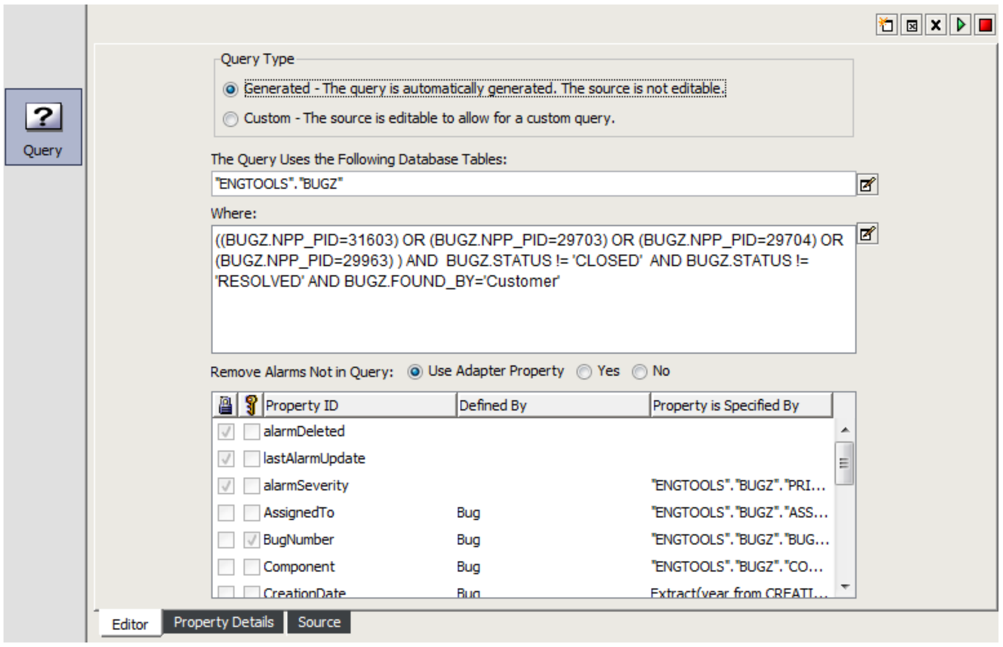
4 Do any of the following to specify the database tables:

- Enter the database tables in *The Query Uses the Following Database Tables* text area.

  Use a comma to separate database table names.
- Drag and drop the database tables and columns directly from the *Database Navigator* pane
- Click (*Edit Tables*) to open an edit dialog box for additional typing space.

5 Depending on the Query Type, do one of the following:

- In the Editor tab, enter a Where clause in the *Where* text area or click (*Edit Where*) to open an edit dialog box for more space to enter the Where clause:



Query Type
[✓ Generated - The query is automatically generated. The source is not editable.]
Custom - The source is editable to allow for a custom query.

The Query Uses the Following Database Tables:
"ENGTOOLS"."BUGZ"

Where:
((BUGZ.NPP_PID=31603) OR (BUGZ.NPP_PID=29703) OR (BUGZ.NPP_PID=29704) OR (BUGZ.NPP_PID=29963)) AND BUGZ.STATUS != 'CLOSED' AND BUGZ.STATUS != 'RESOLVED' AND BUGZ.FOUND_BY='Customer'

Remove Alarms Not in Query: [✓ Use Adapter Property] Yes No

		Property ID	Defined By	Property is Specified By
✓		alarmDeleted		
✓		lastAlarmUpdate		
✓		alarmSeverity		"ENGTOOLS"."BUGZ"."PRI...
		AssignedTo	Bug	"ENGTOOLS"."BUGZ"."ASS...
	✓	BugNumber	Bug	"ENGTOOLS"."BUGZ"."BUG...
		Component	Bug	"ENGTOOLS"."BUGZ"."CO...
		CreationDate	Bug	Extract(year from CREATI...

- If defining a custom query, switch to the *Source* tab to edit the source code directly.

If no lock mode is set in the Data Integrator Query (for example, `select * from` *`table`* `WITH (ROWLOCK)` when using SQL Server), the select statements use the default locking strategy for the database.

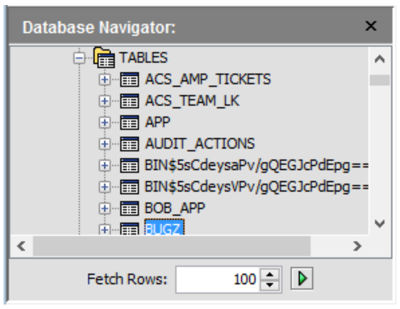
6 To determine whether to remove all alarms that are not found in a query, select one of the following *Remove Alarms Not in Query* radio buttons:

- **Use Adapter Property:** Select to use the current value for the adapter property. For more information, see Section 4.4, "Specifying Run-Time Adapter Properties," on page 36.
- **Yes:** Select to remove all alarms not listed in the subsequent query.
- **No:** Select to keep all alarms regardless of the query results.

7 Switch to the *Property Details* tab to define how each property value is extracted from the database using the query.

For instructions on defining properties, see Section 6.2, “Defining Element or Alarm Properties,” on page 59.

8 Use the *FetchRows* tab at the bottom of the *Database Navigator* pane to run a quick sampling from a database table.



6.2 Defining Element or Alarm Properties

After defining the query, map database information to element or alarm properties that display in the Operations Center *Summary* or *Alarms* view.

- Section 6.2.1, “Understanding the Key Property,” on page 59
- Section 6.2.2, “Mapping Element or Alarm Properties,” on page 60
- Section 6.2.3, “Understanding Computed Properties,” on page 62
- Section 6.2.4, “Understanding Alarm Severity and Element Condition Property Data Types,” on page 63
- Section 6.2.5, “Using the HierarchyKey Property to Simplify Element Naming,” on page 64
- Section 6.2.6, “Deleting a Property,” on page 66

6.2.1 Understanding the Key Property

When defining a property, it is possible to define a Key property that ensures unique elements or alarms are brought into Operations Center. For example, define the trouble ticket ID property as the Key property so that alarms that are brought into Operations Center have unique IDs corresponding to trouble ticket IDs in the database.

Defining unique, persistent alarm IDs ensures that in situations where the adapter stops and restarts, the system does not reread and create duplicate alarms when processing the data for alarm history or for use in SLA calculations. Stored alarm values are updated, if necessary.

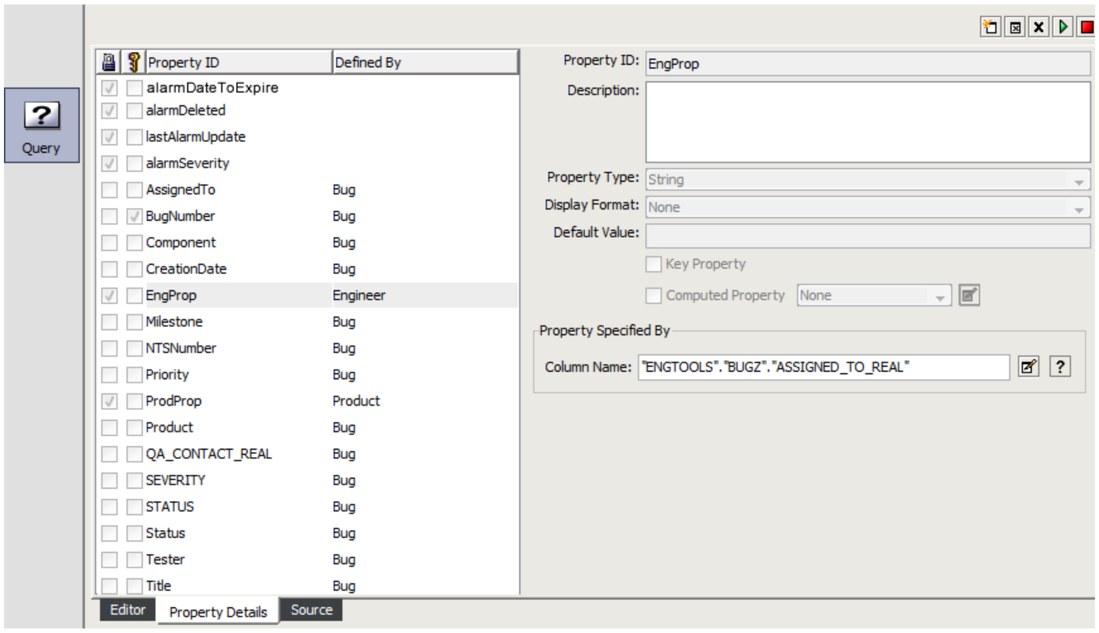
Defining the key property is explained in Section 6.2.2, “Mapping Element or Alarm Properties,” on page 60.

6.2.2 Mapping Element or Alarm Properties

To map element and alarm properties:

1. Click the *Property Details* tab at the bottom of the *Query Results* pane.

   Existing properties display on the left side of the screen:



		Property ID	Defined By
✓		alarmDateToExpire	
✓		alarmDeleted	
✓		lastAlarmUpdate	
✓		alarmSeverity	
		AssignedTo	Bug
	✓	BugNumber	Bug
		Component	Bug
		CreationDate	Bug
✓		EngProp	Engineer
		Milestone	Bug
		NTSNumber	Bug
		Priority	Bug
✓		ProdProp	Product
		Product	Bug
		QA_CONTACT_REAL	Bug
		SEVERITY	Bug
		STATUS	Bug
		Status	Bug
		Tester	Bug
		Title	Bug

Property ID: EngProp
Description:
Property Type: String
Display Format: None
Default Value:
Key Property
Computed Property None
Property Specified By
Column Name: "ENGTOOLS"."BUGZ"."ASSIGNED_TO_REAL"

2. To add a property, click (*Add Property*).

   A new property displays in the *Property ID List* and is open in the tab.

3. To specify a name for the property, do the following:

   - Enter a unique name in the *Property ID* field. Do not use spaces.

     For instructions on manually configuring alarm column to include spaces, see Section 9.1, “Manually Configuring Alarm Column Names,” on page 83.

   - To automatically create parent elements in a parent/child relationship from database information, enter `parentElementID` in the *Property ID* field.

   - To automatically create child elements (under the *parentElementID* elements) without identifying the property as a *Key Property*, enter `elementName` in the *PropertyID* field.

     If the `elementName` property ID is not used, then the property selected as *Key Property* is used to generate child elements.

4. Specify a description for the property in the *Description* field.

5. Click the *Property Type* drop-down list, then select one of the following types:

   Alarm Severity: Displays data as an alarm severity value (`Critical`, `Major`, `Minor`, and so on).

   For more information, see Section 6.2.4, “Understanding Alarm Severity and Element Condition Property Data Types,” on page 63.

   Boolean: Displays data as a True or False value.

   Byte: Displays data as a whole number value ranging from -128 to 127.

   Character: Displays data as any symbol value that requires one byte of storage.

   Double: Displays data as a more precise numerical value ranging from 4.9E-324 to 1.7E+308.

Element Condition: Displays data as an element condition value.

For more information, see Section 6.2.4, "Understanding Alarm Severity and Element Condition Property Data Types," on page 63.

Float: Displays data as a numerical value with a decimal ranging from 1.4E-45 to 3.4E+38.

Integer: Displays data as a whole number value.

Long: Displays data as a numerical value between -2,147,483,648 and 2,147,483,648.

Short: Displays data as a numerical value between -32,768 and 32,767.

String: Displays data as a series of characters manipulated as a group.

Time Stamp: A date and time value.

6 Click the *Display Format* drop-down list, then select a format.

The options are based on the selected *Property Type* and can include:

Composed Text: Displays data as static text while including a table column value within the text. Enclose the table column in brackets. For example: `text [table.column] text`. This is used with the String property type.

Date: Displays data as a standard date format.

DateTime: Displays data as a standard date format, including the time.

IPAddress: Displays data as an IP address syntax.

Image: Displays data as the name and address of an image.

Multiline Text: Displays data as multiline text.

None: Selected as the default when a display format is not applicable.

Password: Displays data as a password in encoded format.

Percentage: Displays data as a number as a percentage.

Script text: Displays data as script code.

Text Field: Displays data as a text field

Time: Displays data as a time format.

URL: Displays data as the URL as a hypertext link.

Click the hyperlink to open the URL in a new window. Note the URL displays only in the Property page; it does not display as a column in the *Alarms* view.

If this property type (or URL Embedded Text) is imported into the following portlets, it possible to click the URL link and open a new window using the specified URL:

- Alarm
- Alarm Properties
- Element Status
- Hotlist Properties
- Information
- Element Properties

For details on each of these portlets, see "Configuring Operations Center Portlets" in the *Operations Center Dashboard Guide*.

URL Embedded Text: Displays data as The URL within a text string is highlighted as a hypertext link. Used when the “http” string is embedded within a property value.

Click the hyperlink to open the URL in a new window.

Requirements for the URL format:

- URLs must be at the beginning or end of a string or surrounded by spaces.
- The required URL format is:

  {space or beginning of string}{alphanumeric characters}://*{URL characters}{space or end of string}{URL characters}*

  URLs cannot contain a double quote, less than, or greater than characters.

URL Page: Displays data as the URL page.

7 To specify a default value for the property that is used if the value is null, enter the value in the *Default Value* field. For example, for a column representing *City*, the default value might be `No city provided`.

8 If the property should be used to ensure unique elements or alarms are extracted from the database, select the *Key Property* check box. For more information, see Section 6.2.1, “Understanding the Key Property,” on page 59.

9 If the property is to be computed, select the *Computed Property* option and specify the type.

For information on computational types, see Section 6.2.3, “Understanding Computed Properties,” on page 62.

10 (Element Properties Only) If performance and scalability issues are of concern, consider selecting the *Fetch Property Value on Demand* check box to load property values only when a user clicks the associated property sheet.

11 Link the property to a database column by doing any of the following:

- Enter the column name in the *Column Name* field.
- Drag and drop a column from the *Database Navigator* pane into the *Column Name* field.
- Click (*Edit Value*) to open an edit dialog box for more space to enter the *Column Name* value.

12 For additional information on valid entries for generated vs. custom queries, click the *Help* button.

6.2.3 Understanding Computed Properties

A computation can be used to define an element or alarm property. The values are assigned based on a column value across all database rows read by the query after the adapter starts.

Various computation types can be selected when defining an element or alarm property:

- **First:** Returns the value of the first row found for the specified table column.
- **First With Value:** Returns the value of the first row having a non-null value for the specified table column.
- **Highest:** Returns the highest value found for the specified table column.
- **Last:** Returns the value found in the last row for the specified table column.
- **Last With Value:** Returns the value of the last row having a non-null value for the specified table column.
- **Lowest:** Returns the value of the row having the lowest value for the specified table column.
- **None:** Performs no calculation.

- **Regular Expression:** Returns a value calculated using the regular expression entered. Click *Edit Computed Property Parameter* to enter a regular expression.
- **Text Date Highest:** Attempts to convert the text into data and return the highest value. Available for text date columns.
- **Text Date Lowest:** Attempts to convert the text into data and return the lowest value. Available for text date columns.
- **Text Numeric Highest:** Attempts to convert the text into a number and return the highest value. Available for text numeric columns.
- **Text Numeric Lowest:** Attempts to convert the text into a number and return the lowest value. Available for text numeric columns.

Notes concerning these computation types:

- First, First With Value, Last and Last With Value are very useful if an *ORDER BY* column has been specified. For example, in the Query Page, Editor tab, Where: text field: ID > 0 ORDER BY ID ASC. In this case First, First With Value, Last, and Last With Value can act similar to Highest and Lowest.
- Text Date Highest and Text Date Lowest are useful if a date is encoded as text instead of an underlying database date type. If two text values 4/5/2006 and Jan 1 2006, are sorted using the Highest computed property, they are ordered 4/5/2006, then Jan 1 2006 based on the underlying text type sort order. Based on the Text Date Highest computed property, they are correctly ordered Jan 1 2006, then 4/5/2006.

For instructions on defining element or alarm properties from a query, see Section 6.2.2, “Mapping Element or Alarm Properties,” on page 60.

6.2.4 Understanding Alarm Severity and Element Condition Property Data Types

If the property type is set to *Alarm Severity* or *Element Condition*, select the appropriate data type from the *Data Type* drop-down list, then set the minimum and maximum values in the *Value Mapping* table.

The *Data Type* options include:

Byte: Sets the property to a whole number value ranging from -128 to 127. Both *Min Value* and *Max Value* are inclusive.

Double: Sets the property to a more precise numerical value ranging from 4.9E-324 to 1.7E+308. In this case, *Min Value* specified is inclusive (>=) while *Max Value* specified is exclusive (<) to help avoid potential gaps with decimal numbering values.

Float: Sets the property to a numerical value with a decimal part ranging from 1.4E-45 to 3.4E+38. In this case, *Min Value* specified is inclusive (>=) while *Max Value* specified is exclusive (<) to help avoid potential gaps with decimal numbering values.

Integer: Sets the property to a whole number value. Both *Min Value* and *Max Value* are inclusive.

Long: Sets the property to a numerical value between -2,147,483,648 and 2,147,483,648. Both *Min Value* and *Max Value* are inclusive.

Short: Sets the property to a numerical value between -32,768 and 32,767. Both *Min Value* and *Max Value* are inclusive.

String: Sets the property to a series of characters manipulated as a group. A regular expression can be used to specify the *Column Value* when the *Data Type* is set to `String` but it must be enclosed within forward slashes(/).

For example, a regular expression such as `^.*\b(NEW|ASSIGNED|NEEDINFO|REOPENED)\b.*$` must be defined as `/^.*\b(NEW|ASSIGNED|NEEDINFO|REOPENED)\b.*$/`

6.2.5 Using the HierarchyKey Property to Simplify Element Naming

- "Using the HierarchyKey property" on page 64
- "Understanding the HierarchyKey Property" on page 64

Using the HierarchyKey property

Use the HierarchyKey property to simplify the element name section in distinguished names.

For more information on the HierarchyKey property, see "Understanding the HierarchyKey Property" on page 64.

Understanding the HierarchyKey Property

HierarchyKey can generate simpler element DNames than those DNames that result from using a natural database key to manage a set of elements.

As a simple example, assume there are a table of servers and a table of applications. Each table contains a unique integer (or long, UUID, and so on) key, and a name. Assume `serverId` or `appId` is selected as the *DBElement* key, and `serverName` or `appName` is selected as the element name.

Table 6-1 and Table 6-2 show how the example tables of servers and applications are defined and the database key is assigned.

Table 6-1 *Servers Table*

serverID	serverName
1	Server1
2	Server2

Table 6-2 *Applications Table*

appID	appName
3	Application 1
4	Application 2

The DBElement types are modeled so that each appears under a separate Data Integrator group; one named “Servers” and one named “Applications”:

```
Servers
   ServerDBElement
      serverId: marked as key property
      serverName: elementName property
Applications
   AppDBElement
      appId: marked as key property
      appName: elementName property
```

In Operations Center, the element hierarchy looks similar to this:

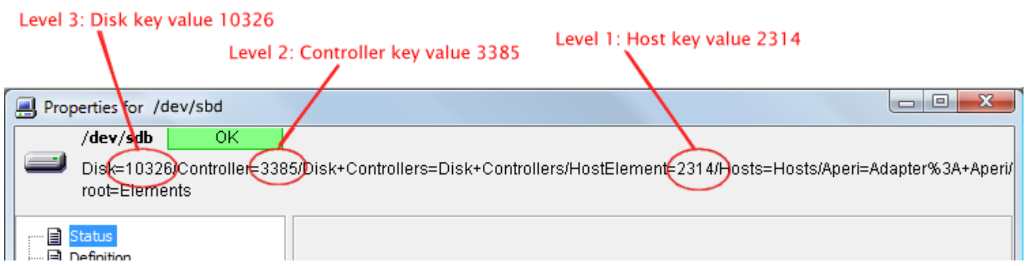
```
MyAdapter
   Servers
      Server1
      Server2
   Applications
      Application 1
      Application 2
```

Data Integrator uses the DBElement key property for each entity as the hierarchy key. This results in a proper uniqueness in the hierarchy, but makes the distinguished names practically unusable for a correlation. Continuing with the example, the resulting distinguished names are:

- myBDI=MyAdapter/root=Elements
- ServerDBElement=1/Servers=Servers/myBDI=MyAdapter/root=Elements
- ServerDBElement=2/Servers=Servers/myBDI=MyAdapter/root=Elements
- AppDBElement=3/Applications=Applications/myBDI=MyAdapter/root=Elements
- AppDBElement=4/Applications=Applications/myBDI=MyAdapter/root=Elements

Note that 1, 2, 3, and 4 in the hierarchy correspond to Server1, Server2, Application 1, and Application 2 respectively in the element names. Figure 6-1 illustrates the three-level nesting of names:

***Figure 6-1** Properties for an Element*


Level 3: Disk key value 10326
Level 2: Controller key value 3385
Level 1: Host key value 2314
Properties for /dev/sbd
/dev/sdb
OK
Status
Definition

Disk=10326/Controller=3385/Disk+Controllers=Disk+Controllers/HostElement=2314/Hosts=Hosts/Aperi=Adapter%3A+Aperi/root=Elements

A problem occurs if you need to perform element correlation, such as creating a SCM definition that correlates to elements named similar to the one shown in the previous figure. It is nearly impossible to properly correlate such named elements from external data sources. Many users wind up changing the DBElement keys to an element name. This solution can work, except when there is a need to model relationships in the application-to-server mapping that use a database mapping of many-to-many, in commonly occurring situations when one server is used by multiple applications.

In these cases, you can use the simpler `hierarchyKey` property to provide the name of the element segment in the distinguished name. Using `hierarchyKey` places the DBElement in the resulting element hierarchy using a string-based name, instead of the common case of a numeric key value or other value that is difficult to correlate.

6.2.6 Deleting a Property

To delete a property from a query:

1. Click the *Property Details* subtab at the bottom of the *Query Results* pane.
2. Select a property in the *Property ID List* pane.
3. Click ⊠ (*Delete Selected Properties*) to remove the property definition.

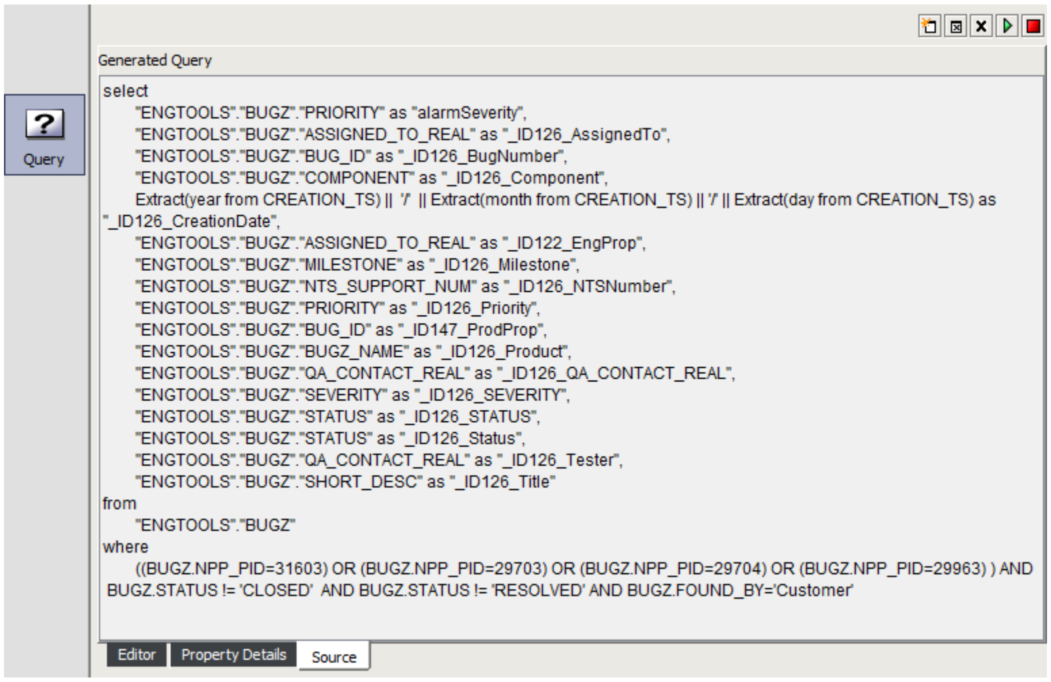
6.3 Understanding the Source View

The Source subtab, found at the bottom of the *Query Results* pane, enables viewing the actual query statement that was created by the settings established in the *Editor* and *Property Details* tabs.

If the query is a Custom query, the *Source* tab can be used to edit the query source code directly.

Figure 6-2 illustrates the *Source* subtab enabling viewing and editing of the source code for the query:

Figure 6-2 *Definition Editor Query Pane*



```
select
     "ENGTOOLS"."BUGZ"."PRIORITY" as "alarmSeverity",
     "ENGTOOLS"."BUGZ"."ASSIGNED_TO_REAL" as "_ID126_AssignedTo",
     "ENGTOOLS"."BUGZ"."BUG_ID" as "_ID126_BugNumber",
     "ENGTOOLS"."BUGZ"."COMPONENT" as "_ID126_Component",
     Extract(year from CREATION_TS) || '/' || Extract(month from CREATION_TS) || '/' || Extract(day from CREATION_TS) as
"_ID126_CreationDate",
     "ENGTOOLS"."BUGZ"."ASSIGNED_TO_REAL" as "_ID122_EngProp",
     "ENGTOOLS"."BUGZ"."MILESTONE" as "_ID126_Milestone",
     "ENGTOOLS"."BUGZ"."NTS_SUPPORT_NUM" as "_ID126_NTSNumber",
     "ENGTOOLS"."BUGZ"."PRIORITY" as "_ID126_Priority",
     "ENGTOOLS"."BUGZ"."BUG_ID" as "_ID147_ProdProp",
     "ENGTOOLS"."BUGZ"."BUGZ_NAME" as "_ID126_Product",
     "ENGTOOLS"."BUGZ"."QA_CONTACT_REAL" as "_ID126_QA_CONTACT_REAL",
     "ENGTOOLS"."BUGZ"."SEVERITY" as "_ID126_SEVERITY",
     "ENGTOOLS"."BUGZ"."STATUS" as "_ID126_STATUS",
     "ENGTOOLS"."BUGZ"."STATUS" as "_ID126_Status",
     "ENGTOOLS"."BUGZ"."QA_CONTACT_REAL" as "_ID126_Tester",
     "ENGTOOLS"."BUGZ"."SHORT_DESC" as "_ID126_Title"
from
     "ENGTOOLS"."BUGZ"
where
     ((BUGZ.NPP_PID=31603) OR (BUGZ.NPP_PID=29703) OR (BUGZ.NPP_PID=29704) OR (BUGZ.NPP_PID=29963) ) AND
 BUGZ.STATUS != 'CLOSED'  AND BUGZ.STATUS != 'RESOLVED' AND BUGZ.FOUND_BY='Customer'
```

6.4 Understanding Property Text Substitution

Data Integrator queries support the use of property text substitution that can be utilized in the Where statement. This allows the substitution of values from a property into a query. For example:

```
LASTUPDATE > &sq;{CP_HIGHEST_LASTUPDATE}&sq; or INTERNALID >
{CP_HIGHEST_INTERNALID::-1}
```

In this example, the properties `CP_HIGHEST_LASTUPDATE` and `CP_HIGHEST_INTERNALID` are substituted in the query. The initial substitution uses the empty string for `CP_HIGHEST_LASTUPDATE` and `-1` for `CP_HIGHEST_INTERNAL ID` and provides a resulting initial query with the clause of `LASTUPDATE > '' or INTERNALID > -1`.

Subsequent queries substitute the actual values of the two computed properties, based on their computed property type, such as `LASTUPDATE > '2/12/2006 11:12 AM' or INTERNLID > 827`.

The general format for a text substitution is:

{*substitution_specification*::*initial_value*}

or

{*substitution_specification*}

The second format uses an initial value of the empty string, as no initial value is specifically given. Do not use spaces in the *substitution_specification*. Only use spaces in the initial value if spaces are needed. If the computed property value must be single-quoted in the query, then surround the substitution with `&sq;` (or `&singlequote`), which inserts the single quotes (').

To use the format {...} literally in the query, surround the braces with single quotes. For example: `NAME > '{BOB}'` makes no substitution because `{BOB}` is surrounded by single quotes. The {*substitution_specification*} format is based on the JAVA MessageFormat class.

6.5 Setting up a Query using a Stored Procedure

It is possible to query a database using stored procedures rather than building a query using the steps described previously.

To define a query from a stored procedure:

1. Select an element in the *Definition Navigator* pane.
2. In the right pane, click the *Query* icon to open the *Editor* tab.
3. Select the *Custom* radio button for the *Query Type*.
4. Click the *Source* tab.
5. In the *Custom Query* pane, specify the execute statement for the procedure invocation.

   Oracle stored procedures use “out” parameters to return values and must be handled in a special way. The execute statement must be:

   ```
   call procedure_name(?)
   ```

   where *procedure_name* is the name of the stored procedure. The question mark represents the out cursor parameter to use for Data Integrator. If the procedure has multiple parameters, fill them in normally; just use a question mark for the out parameter.
6. Click the *Property Details* tab.
7. Create properties for procedure results columns.

   To map result columns to properties, specify the name of the result column for each property in the *Column Name* field.

6.6 Setting up a Delta Query

Data Integrator provides a delta query feature where the adapter checks for new alarms without returning all alarms each time.

- Section 6.6.1, “Defining a Delta Query,” on page 68
- Section 6.6.2, “Defining a Custom Query,” on page 68

6.6.1 Defining a Delta Query

1. Select an element in the *Definition Navigator* pane.
2. In the right pane, click the *Query* icon to display the *Editor* tab.
3. Use the *Editor* and *Source* tabs to define the generated or custom query.

   For more information, see Chapter 6, "Defining Queries and Properties," on page 57.
4. Click the *Property Details* tab.
5. To create properties for procedure results columns:

   5a *Select the Computed Property* check box.

   5b Select *Highest* from the drop-down list.

   5c To check for any delta results, specify the name of the result column in the *Column Name* field.

6.6.2 Defining a Custom Query

1. Select an element in the *Definition Navigator* pane.
2. In the right pane, click the *Query* icon.
3. Click the *Source* tab.
4. Include the Computed Property in the stored procedure invocation.

   For example, if the property using the *Computed Property* option is `HighestCustomerID`, then the execute statement might look like the following:

```
#if ( ${query.initial} )
  Execute sp_mysp 0 GETDATE()
#else
  Execute sp_mysp ${query.property.HighestCustomerID} GETDATE()#end
```

5. Click the *Editor* tab.
6. Click the *No* radio button next to *Remove Alarms Not in Query*.

   All alarms not in the subsequent query remain in Operations Center.

   If you click *Yes*, the alarms are removed and the adapter element shows only the delta alarms.

6.7 Using Macro Expression Query Preprocessing

Macro expression query preprocessing is available in Data Integrator for added flexibility in creating query text. Macro expressions allow embedding control statements and variable substitution in query statements. Operations Center software uses Velocity, which is a third party package provided by the Apache Organization that is bundled into Operations Center for macro expressions. Documentation for Velocity is available at http://jakarta.apache.org/velocity (http://jakarta.apache.org/velocity).

One example of a commonly used query is tracking the highest value entered for a computed property. The following example assumes `HighestCustomerID` is a computed property that tracks the highest customer ID queried. The first query selects all `CustomerID` values where the `CustomerID` is greater than 0.

```
select CustomerID from CustomerInfo where
#if ( ${query.initial} )
    CustomerID > 0
#else
    CustomerID > ${query.property.HighestCustomerID}
#end
```

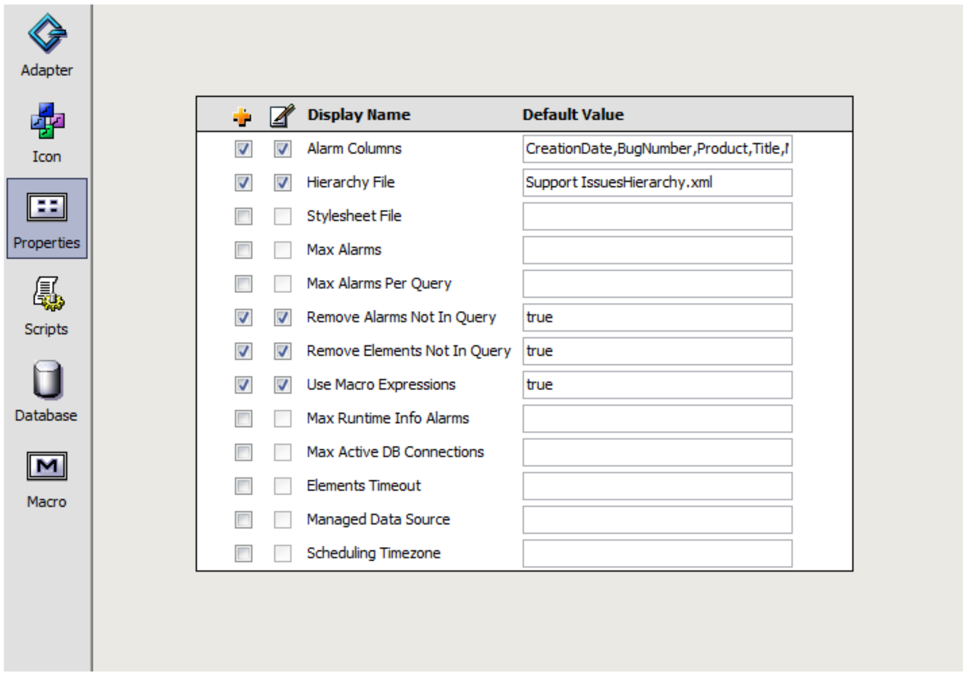
The second and subsequent queries select only `CustomerID` values greater than the `HighestCustomerID`. If new `CustomerID` values always increase, this query only selects new `CustomerID` values during subsequent queries, instead of requerying the entire table for each `CustomerID` during each scheduled query.

Macro expression query preprocessing is the preferred method for Data Integrator execution. NetIQ recommends not using the older delta query mechanism using `'{<COMPUTED PROPERTY NAME>::<INITIAL VALUE>}'` and might remove it from a future version of Data Integrator. However, during this transition period, to allow for backwards compatibility, the old Data Integrator delta query mechanism is the default for query preprocessing, unless Macro expression query preprocessing is enabled, as described below.

6.7.1 Enabling Macro Expression

To enable macro expression query preprocessing:

1. In the *Explorer* pane, expand *Administration* > *Adapters* > *Data Integrator*.
2. Right-click an adapter, then select *Edit Definition* to open the Data Integrator Editor.
3. In the *Definition Navigator* pane, click the top node in the adapter hierarchy.
4. In the right pane, click the *Properties* icon to display the adapter properties:



		Display Name	Default Value
☑	☑	Alarm Columns	CreationDate,BugNumber,Product,Title,I
☑	☑	Hierarchy File	Support IssuesHierarchy.xml
☐	☐	Stylesheet File	
☐	☐	Max Alarms	
☐	☐	Max Alarms Per Query	
☑	☑	Remove Alarms Not In Query	true
☑	☑	Remove Elements Not In Query	true
☑	☑	Use Macro Expressions	true
☐	☐	Max Runtime Info Alarms	
☐	☐	Max Active DB Connections	
☐	☐	Elements Timeout	
☐	☐	Managed Data Source	
☐	☐	Scheduling Timezone	

5. Click the *Activate* check box next to *Use Macro Expressions*.

   This enables macro expression query preprocessing. It is disabled by default.
6. In the *Default Value* field next to *Use Macro Expressions*, enter `true`.
7. Close the Data Integrator Editor.
8. When prompted to save changes, click *Yes*.

6.7.2 Understanding Macro Expression Variables

The following variable values are predefined and can be referenced from within a macro expression preprocessed query on the Operations Center server and from the Data Integrator Definition Editor:

- **${query.initial}:** True the first time the query is run; otherwise, False.
- **${query. macroExpression*n*}:** The value of the Macro Expression adapter property, where *n* is between 1 and 15.

  For more information, see Section 6.7.3, “Defining Default Macro Expression Adapter Values,” on page 71.
- **${query.integration.dbHost}:** The value of the database connection host.
- **${query.integration.dbPort}:** The value of the database connection port.

- **${query.integration.dbUser}:** The value of the database connection user.
- **${query.integration.dbType}:** The value of the database connection type.

The following predefined values can be referenced from within a macro expression preprocessed query on the Operations Center server only:

- **${query.property.*Computed_Property_Name*}:** The value of the Computed Property named in *Computed_Property_Name*. For example:

  ${query.property.HighestCustomerID}
- **${query.integration.*Integration_Property_Name*}:** The value of the integration property named in *Integration_Property_Name*. For example:

  ${query.integration.dbType}
- **${query.control.*Control_Property_Name*}:** The value of the integration property named in *Integration_Property_Name*. For example:

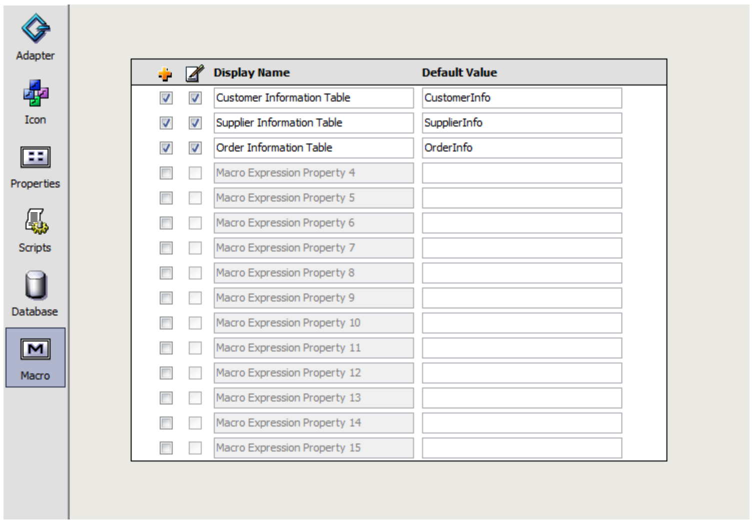
  ${query.control.statusMessage}
- **${query.integrationInstance}:** The Data Integrator integration object instance.
- **${query.controlInstance}:** The Data Integrator control object instance.
- **${query.controlInstance.fail(*message*)}:** Causes the Data Integrator integration to fail with the reason provided using the message argument. For example, on a SQL server, to stop your configuration from your SQL code:

  ```
  RAISERROR('${query.controlInstance.fail("Invalid Configuration:
  ${query.macroExpression4}")}',18,1);
  ```

6.7.3 Defining Default Macro Expression Adapter Values

After enabling macro expressions, it is possible to define up to 15 adapter properties that macro expression directives in the query text can reference, as illustrated in Figure 6-3:

Figure 6-3 *Adapter Properties*



		Display Name	Default Value
✓	✓	Customer Information Table	CustomerInfo
✓	✓	Supplier Information Table	SupplierInfo
✓	✓	Order Information Table	OrderInfo
☐	☐	Macro Expression Property 4	
☐	☐	Macro Expression Property 5	
☐	☐	Macro Expression Property 6	
☐	☐	Macro Expression Property 7	
☐	☐	Macro Expression Property 8	
☐	☐	Macro Expression Property 9	
☐	☐	Macro Expression Property 10	
☐	☐	Macro Expression Property 11	
☐	☐	Macro Expression Property 12	
☐	☐	Macro Expression Property 13	
☐	☐	Macro Expression Property 14	
☐	☐	Macro Expression Property 15	

The previous figure defines five adapter properties. The first three properties have selected *Add to Property Page* check boxes, which means these properties display in the adapter property pages and can be modified. The last two properties are enabled using the default values, but do not display in adapter property pages and cannot be modified.

If macro expression processing is enabled, it is possible to reference these adapter properties from within queries, such as ${query. macroExpression*n*} where *n* is the macro expression property index from 1 to 15.

For example, the following statement selects the *CustomerID* column from the *CustomerInfo* table where the *CustomerID* value is between 50 and 100:

```
Select CustomerID from ${query.macroExpression1} where CustomerID >
${query.macroExpression4} AND CustomerID < ${query.macroExpression5}
```

If the administrator changes the Customer Information Table adapter property to a different value, then it selects from that table instead.

To define the adapter properties:

1. In the Data Integrator Editor *Definition Navigator* pane, select the adapter definition root element.
2. In the right pane, click the *Macro* icon to display the property names and values.
3. Click the (*Activate*) check box to enable the property.
4. Click the (*Add to Property Page*) check box to display the adapter property for the run-time adapter instance.

   This allows administrators to modify the property value.

   Leave the *Add to Property Page* check box deselected to activate the adapter property, but not display it for the adapter instance.
5. Specify a name in the *Display Name* field.

   The field cannot be edited if the *Add to Property Page* check box is deselected.
6. Specify a default property value in the *Default Value* field.
7. Close the Data Integrator Editor.
8. When prompted to save your changes, click *Yes*.

6.7.4 Filtering Data Based on Date

A common application of macro expressions is filtering data most recently added to the database, such as data entered in the last x hours or days.

In this example, upon the initial query, the adapter pulls in all new data entered into the database within the last hour. If no data was entered in the last hour, it uses the Last Time Stamp property to determine the last time stamp value read from the database. Each subsequent query of the database reads only the new data that was entered in the database with a time stamp greater than the time stamp recorded by the last query.

The following macro expression is in the WHERE clause of the Alarm query:

```
dbo.measurements.metric_id = dbo.metrics.metric_id and
dbo.measurements.metric_when >
#if ( ${query.initial} )
dateadd(hour, -1, getdate())
#else
'${query.getProperty().lastTimeStamp}'
#end
```

6.8 Clearing Queries

Clearing a query removes all previously defined query statements and property definitions and enables defining a new query.

To clear an entire query:

1. Click the *Query* icon for an element or alarm definition.
2. Click (*Clear Query Definition*).

   All previously defined query statements and property definitions are removed. However, the changes are not visible until the page is refreshed (select a different element, then reselect the page with the deleted query).

6.9 Testing Queries

It is easy to test a query to verify that it produces the desired results at run-time.

1. To test a query, after all parameters are specified for the query in the *Query Results* pane, click (*Execute Query*).

   The results display in the *Query Results* pane.
2. To stop a query execution (stopping the query in progress), click (*Stop Query Execution*).

6.10 Scheduling Queries

Each database element and alarm that has a query needs a basic schedule that allows the adapter to routinely check for additional information and updates.

Schedules are shared among all database and element definitions within the adapter definitions. Therefore, before changing a schedule that could be used by multiple definitions, make sure the changes are appropriate for all.

If you do not create and assign a schedule to the database element or alarm definition, the DEFAULT_SCHEDULE is automatically used. The default schedule, which can be modified, runs every 5 minutes over a 24 hour cycle.

Set a schedule to run at adapter start, on a routine basis, or at set times. Multiple options can be selected. When specifying a time, you can use UTC (coordinated universal time).

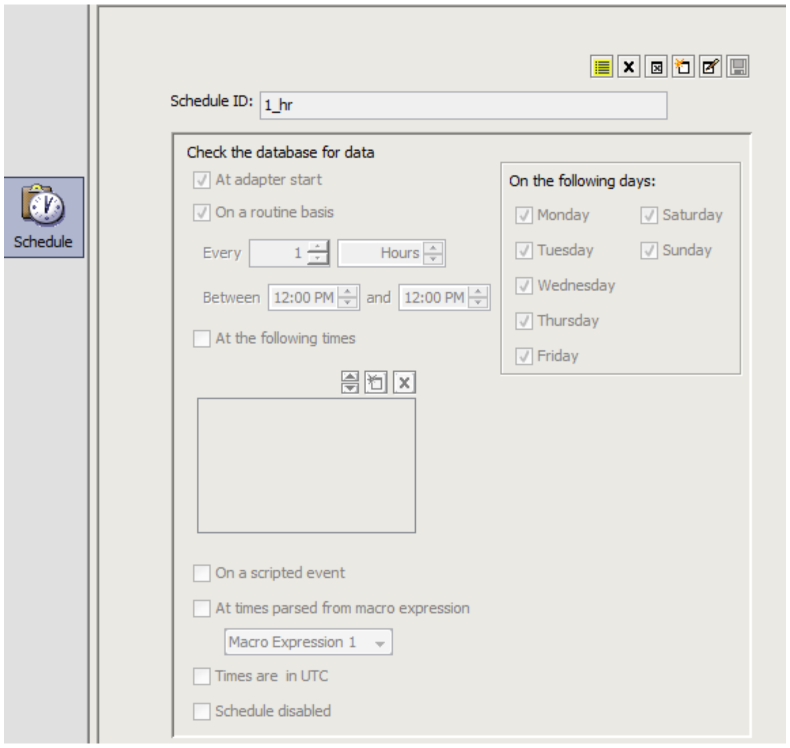
The adapter property Scheduling Timezone can determine which time zone is used to run scheduled queries. For more information, see Section 4.4, “Specifying Run-Time Adapter Properties,” on page 36.

6.10.1 Scheduling a Query

1 Select an element or alarm in the *Definition Navigator* pane.

2 In the right pane, click the *Schedule* icon to assign a schedule to the query.

If no schedules exist, skip ahead to the next step-by-step instructions on how to create a schedule:



Schedule ID: 1_hr

Check the database for data

- [✓] At adapter start
- [✓] On a routine basis

Every 1 Hours

Between 12:00 PM and 12:00 PM

- [ ] At the following times

On the following days:

- [✓] Monday
- [✓] Saturday
- [✓] Tuesday
- [✓] Sunday
- [✓] Wednesday
- [✓] Thursday
- [✓] Friday

- [ ] On a scripted event
- [ ] At times parsed from macro expression

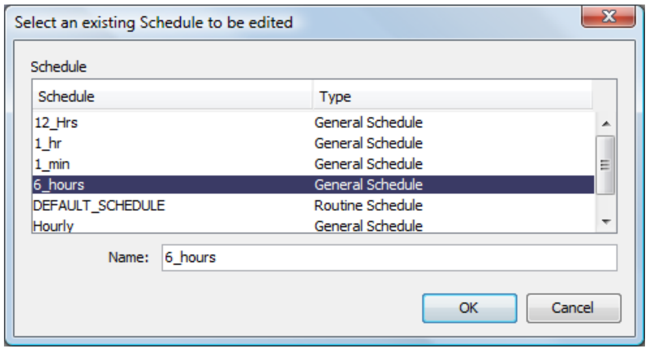
Macro Expression 1

- [ ] Times are in UTC
- [ ] Schedule disabled

3 Click (*Select an Existing Schedule*) to open the dialog box.

4 Select a defined schedule.

The schedule's name displays in the *Name* field:


Select an existing Schedule to be edited

Schedule

Schedule	Type
12_Hrs	General Schedule
1_hr	General Schedule
1_min	General Schedule
6_hours	General Schedule
DEFAULT_SCHEDULE	Routine Schedule
Hourly	General Schedule

Name: 6_hours

5 Click *OK* to assign the schedule to the definition.

6.10.2 Creating a Schedule

1 Select an element in the *Definition Navigator* pane.

2 In the right pane, click the *Schedule* icon.

3 Click (*Create a New Schedule*).

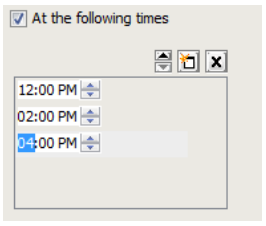
The *Query Results* pane is cleared so you can specify the schedule requirements.

4 Specify the name in the *Schedule ID* field.

5 Specify when to run the query (you can select multiple check boxes):

- To run the query whenever the adapter is started, select the *At Adapter Start* check box.
- To run the query on a routine basis, select the *On a Routine Basis* check box.
- Use the spinners to specify how often the query runs between specific times of a day.

6 To run the query at specific times, select the *At the Following Times* check box, then create a time slot and specify the start time:



At the following times

12:00 PM
02:00 PM
04:00 PM

Multiple start times are allowed. Use the buttons to manage multiple time slots:

	Create a new time slot.
	Delete the current time slot.
	If multiple times exist, move the current one up or down in the list.

Keep in mind the adapter property Scheduling Timezone can determine which time zone is used to run scheduled queries.

For more information, see Section 4.4, “Specifying Run-Time Adapter Properties,” on page 36.

7 Select the *On the Following Days* check boxes to specify the days associated with the selected times.

8 To trigger the schedule through a scripted event, select the *On a Scripted Event* check box.

For more information, see Section 6.10.7, “Triggering Schedules through Scripted Events,” on page 77.

9 To use settings from a macro expression instead of those specified here, select the *At Times Parsed From Macro Expression* check box, then select an expression from the drop-down list.

10 To designate that times are in UTC (coordinated universal time), select the *Times Are in UTC* check box.

11 To save the new schedule, click (*Save Schedule Changes*).

The schedule is now available in the Schedule Library for use with other element and alarm definitions within this adapter definition.

6.10.3 Editing a Schedule

Schedules are shared between alarm and element definitions within an adapter definition. Verify that all changes made to a schedule are valid for other definitions that use it.

To edit a schedule:

1. Select an element in the *Definition Navigator* pane.
2. In the right pane, click the *Schedule* icon.
3. Click (*Edit an Existing Schedule*).
4. Select a schedule, then click *OK* to open the schedule.

   The schedule displays in the *Query Results* pane.
5. Make the necessary changes to the schedule.
6. Click (*Save Schedule Changes*).

6.10.4 Disabling a Schedule

Clicking the *Schedule Disabled* check box disables all queries associated with the schedule, including queries using macro expressions.

To disable a schedule so that all queries associated with the schedule do not fire and run:

1. Click the *Schedule* icon.
2. Click (*Edit an Existing Schedule*).
3. Select a schedule, then click *OK* to open the schedule.
4. Select the *Schedule Disabled* check box.
5. Click (*Save Schedule Changes*).

6.10.5 Clearing a Schedule

Clear a schedule so it does not apply to a specific definition. This is useful when you want to replace the current schedule with a new one. In effect, the *Clear* button removes a schedule from an adapter definition, but leaves it available for other queries.

To clear a schedule:

1. Select an element in the *Definition Navigator* pane.
2. In the right pane, click the *Schedule* icon.
3. Click (*Select an Existing Schedule*).
4. Select a schedule, then click *OK* to open the schedule.
5. Click (*Clear Schedule Selection*).

   All schedule fields are cleared and reset.

6.10.6 Deleting a Schedule

Deleting a schedule removes it from the system. If the schedule is assigned to other alarms and elements within the adapter definition, you must assign valid schedules to them.

To delete a schedule:

1. Select an element in the *Definition Navigator* pane.
2. In the right pane, click the *Schedule* icon.
3. Click (*Select an Existing Schedule*).
4. Select a schedule, then click *OK* to open the schedule.
5. Click (*Delete an Existing Schedule*) to permanently delete the schedule from the system.

6.10.7 Triggering Schedules through Scripted Events

Schedules can be triggered through a scripted event. The following are two example scripts:

```
// This script will fire a run of the Data Integrator schedule named ONCE for
// any Data Integrator adapters running that have the trigger by scripted
// event enabled.
load( 'util/bdievent' )
fireBDIEvent( 'ONCE' )

// This script will fire a run of the Data Integrator schedule named ONCE for a
// specific Data Integrator adapter running named BAM5 if it has the trigger by
// scripted event enabled.
load( 'util/bdievent' )
fireBDIEvent( 'ONCE' , 'BAM5' )
```

An alternative to the *Schedule* pane in the *Definition Navigator* pane is specifying schedule settings in text form as a macro expression (adapter property).

Enable the macro expression option and provide a macro expression that contains text (nonempty string). At run-time, the text is used to configure the schedule. All settings in the *Schedule* pane are ignored. If the macro expression is empty, then the settings in the *Schedule* pane are used. If the *Schedule* pane has the *Schedule Disabled* option selected, then the schedule does not fire, regardless of whether a macro expression is selected.

Macro expression schedule text can use either of the following formats:

- [setting1]; [setting2]; [setting3]; ...
- [setting1] & [setting2] & [setting3] & ...

Settings are case-insensitive. The following lists acceptable setting values:

- **‘disable' (aliases are 'disabled', 'off', 'none', 'false'):** If present, the schedule does not fire regardless of the other settings.
- **‘utc':** If present, then dates are interpreted as being in UTC format.
- **‘atstart' (aliases are 'at start', 'onstart', 'on start'):** If present, the schedule fires upon adapter startup.
- **‘atevent' (aliases are 'at event', 'onevent', 'on event'):** If present, then the schedule can be triggered by a scriptable event as described previously.

- **‘every *n* seconds|minutes|hours [between HH:MM and HH:MM] [on MON,TUE,WED,THU,FRI,SAT,SUN]':** Where *n* is a number representing the interval, which can be seconds, minutes or hours (abbreviate using the first letter).

  The between times and the *On Days* clauses are optional. HH hours are always in 24-hour notation as AM and PM are not allowed. Examples:

  - every 5m
  - every 1h between 20:00 and 23:00
  - every 1h on Mon,Wed,Fri
  - every 1 hour
  - every 5 hours between 10:00 and 16:00 on Sat,Sun
- **‘at HH:MM, HH:MM, HH:MM [on MON,TUE,WED,THU,FRI,SAT,SUN]':** Where the *On Days* clause is optional. HH hours are always in 24-hour notation as AM and PM are not allowed. Examples:
  - at 10:00
  - at 10:45, 22:45
  - at 13:30, 16:30 on Mon,Wed,Fri

6.10.8 Scheduling Queries at Adapter Startup and On Demand

Use a job script to run Data Integrator queries at fixed times during the day, such as during off-business hours or when the Data Integrator adapter starts or restarts.

The `executeQueriesForSchedule` method is available to trigger queries on demand for a run-once schedule. The following example script uses this method:

```
...
try
{
   var adapterElement = formula.Root.findElement( adapterName );
   if (adapterElement != null)
   {
       var adapter = adapterElement.getAdapter();
       if ((adapter != null) && (adapter.isStarted() == true))
      {
           var bdi = adapter.getControl();
           if (bdi != null)
          {
                bdi.executeQueriesForSchedule( scheduleName );
          }
       }
  }
catch(e)
{
     formula.log.info("Scheduling BDI from script failed, with following error
message: " + e);
}
...
```

7 Importing and Exporting Adapter Definitions

If various departments are interested in leveraging the same information in similar ways, use the export and import features to share adapter definitions.

- Section 7.1, "Exporting Adapter Definitions," on page 79
- Section 7.2, "Importing an Adapter Definition," on page 80

7.1 Exporting Adapter Definitions

The first step in sharing modifiable adapter definitions with others in your organization or with company partners is to export the adapter definition to an independent file that can be e-mailed or burned to CD-ROM.

- Section 7.1.1, "Exporting an Adapter Definition," on page 79
- Section 7.1.2, "Exporting an Adapter Definition for a Server without a Design-Time License," on page 80

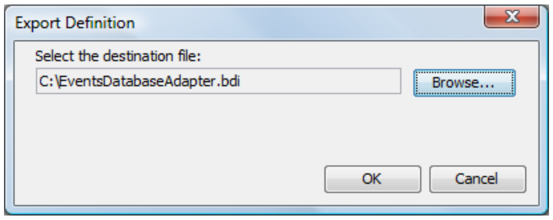
7.1.1 Exporting an Adapter Definition

To export an adapter definition as a `.bdi` file:

1 In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.

2 Right-click an adapter definition, then select *Export Definition* to open the Export Definition dialog box.

3 Click *Browse*, then select a destination directory and specify the name of the exported file.


Export Definition

Select the destination file:

C:\EventsDatabaseAdapter.bdi

4 Click *OK* to export the definition to the specified file.

An adapter definition export file uses the file extension `.bdi`. If you do not declare this extension when exporting the file, the `.bdi` extension is automatically added to the file name.

7.1.2 Exporting an Adapter Definition for a Server without a Design-Time License

Currently, a design-time license is required to access and deploy an adapter definition file. However, the following steps can be used to deploy an adapter definition for a system without a design-time license.

To export an adapter definition for a server without a design-time license:

1. Rename the exported definition to use the `.jar` extension (instead of `.bdi`).
2. Save the JAR file to the `/`*`OperationsCenter_install_path`*`/NOCintegrations` directory.
3. Restart the Operations Center server.

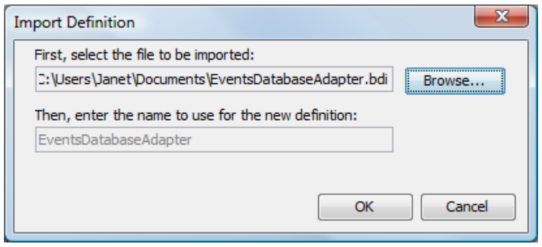
   The integration is available for creating adapter instances.

7.2 Importing an Adapter Definition

Importing an adapter definition file is a quick process that allows adapter definitions to be shared between individuals and organizational departments.

To import an adapter definition:

1. In the *Explorer* pane, expand the *Administration* root element > *Adapters*.
2. Right-click *Data Integrator*, then select *Import Definition* to open the Import Definition dialog box.
3. Click *Browse*, then select the file to import.



Import Definition

First, select the file to be imported:

C:\Users\Janet\Documents\EventsDatabaseAdapter.bdi

Then, enter the name to use for the new definition:

EventsDatabaseAdapter

4. Specify the name to use for the new definition.
5. Click *OK* to import the adapter definition.

   The imported definition displayed under the *Administration* > *Adapters* > *Data Integrator* tree.
6. Because database settings are stored with user preferences, redefine the database connection for the adapter definition.

   For instructions, see Section 4.6, “Setting a Run-Time Database Connection,” on page 39.

   Because development database settings for an adapter definition are stored with user accounts, the *Database Navigator* pane and the *Query Results* pane are unavailable in the Definition Editor when an imported adapter definition is first opened. Define a new database to use for the development environment.

8 Deploying and Revoking Adapters

After finalizing the adapter definition, deploy it in order for Operations Center users to create adapter instances.

8.1 Deploying an Adapter Definition

When an adapter definition is deployed, the adapter is available in the Operations Center console as a new adapter.

To deploy an adapter:

1. In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.
2. Right-click an adapter definition, then select *Deploy Definition as Adapter*.

   The Data Integrator converts the definition to a runtime adapter. It is now available as a Operations Center adapter.

   For more information regarding creating adapter instances, see "Creating Adapters" in the *Operations Center Adapter and Integration Guide*.

8.2 Undeploying Adapters

Situations might occur where an adapter must be revoked. For example, there could be a need to temporarily make an adapter unavailable to users, or the adapter definition has been changed.

Changes made to an adapter definition are not automatically applied to the runtime adapter. If you change adapter properties after deploying an adapter, it is necessary to stop and delete the adapter instance, undeploy the adapter, then redeploy the adapter definition and create adapter instances based on the redeployed definition.

To undeploy an adapter:

1. Stop all adapter instances created by the adapter definition that you want to undeploy.

   Before an adapter definition can be undeployed, it is necessary to stop all adapter instances that are based on the adapter definition.
2. In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.
3. Right-click the adapter definition, then select *Undeploy Definition as Adapter*.

8.3 Redeploying Adapters

To redeploy an adapter:

1. In the *Explorer* pane, expand the *Administration* root element > *Adapters* > *Data Integrator*.
2. Right-click the adapter definition, then select *Redeploy Definition as Adapter*.

9 Customizing Deployed Adapters

The Data Integrator can mine and visualize relationships that can be described in a database data source. This can be accomplished by customizing the generated Data Integrator hierarchy file for the adapter instance. This section explains how to customize deployed adapters to take advantage of Operations Center advanced rendering features that show relationships among elements. These relationships can be enhanced to show dependency relationships that promote state via the dependency tree.

WARNING: If you modify and redeploy the adapter definition, all manual changes are overwritten. Be sure to maintain a backup copy of all manual changes made to the hierarchy file. You should manually redo the changes after redeployment of an adapter.

The following sections contain information about customizing deployed adapters:

- Section 9.1, “Manually Configuring Alarm Column Names,” on page 83
- Section 9.2, “Using the Adapter’s Hierarchy File to Mine and Visualize Relationships,” on page 84
- Section 9.3, “Example: Mining and Visualizing Relationships,” on page 85

9.1 Manually Configuring Alarm Column Names

Although Data Integrator definitions cannot accept spaces in Alarm Query Property IDs, you can still control the display of alarm columns by customizing the Alarms Columns property for each adapter.

The Alarms Columns property contains a comma-separated list that determines which alarm columns display and the order in which the alarm items display (source of alarm, alarm class, etc.) in the Alarms view.

To customize adapter alarm columns:

1 Do one of the following:

- Create a new adapter and select the deployed Data Integration adapter as the type.
- Right-click the adapter under *Administration* > *Adapters*, and select *Properties*.

For information on creating adapters, see “Creating Adapters” in the *Operations Center Adapter and Integration Guide*.

2 Configure the *Alarm Columns* property as necessary to redefine column names using the syntax: `New_Display_Name=Alarm_Query_Property_ID`. For example,

```
Creation Date=CreationDate,Bug Number=BugNumber,Product,Title,Milestone,
Priority,Severity,Component,Status,Assigned To=Assigned
```

Where `Creation Date` is the name of the alarm column to display for alarms, and `CreationDate` is the Property ID as specified in property definition for the alarm query in Section 6.2.2, “Mapping Element or Alarm Properties,” on page 60.

3 If making changes on a running adapter, you must stop and restart the adapter for these changes to be applied.

9.2 Using the Adapter’s Hierarchy File to Mine and Visualize Relationships

Advanced rendering features can show relationships between elements. These relationships can be enhanced to show dependency relationships that promote state via the dependency tree.

Use Data Integrator to mine and visualize relationships that can be described in a database data source. This can be accomplished by customizing the generated Data Integrator hierarchy file for the adapter instance. Dependencies can be defined by creating an explicit hierarchy structure in the Data Integrator adapter.

Relationship data is supported in Data Integrator using one or more alarm definitions, which mine data properties and correlate them to element properties specified by the Data Integrator adapter.

Relationships can provide situational awareness. They can also direct the Service Configuration Manager (SCM) to dynamically create particular Service Views or Service Models. For more information about SCM, see “Using the Service Configuration Manager” in the *Operations Center Service Modeling Guide*.

To set up the hierarchy file to introduce relationship data into Operations Center via Data Integrator:

1. Identify elements in the Data Integrator adapter structure from which relationship data is associated.

   Make a note of element properties that can be used to uniquely identify an element and how they relate to the “to” element in the relationship.

2. Identify elements in the Data Integrator structure to which relationship data is associated.

   Make a note of element properties that can be used to uniquely identify an element and how they relate to the “from” element in the relationship.

   In some cases, the to and from can be the same hierarchy location.

3. Create an alarm definition that queries the database tables that contain relationship data.

   Include properties that can map the from element to the to elements.

4. Edit the adapter Hierarchy File to customize the entry for the alarm definition.

   Add XML syntax to the group element tag for the alarm definition, to indicate how the alarm-oriented data is used to create relationships using from and to elements.

   The following provides an overview of each tag (note that all values are case-sensitive):

   LinkDef.Count: The number of relationship link definitions found in this specific location in the hierarchy. Multiple relationship definitions are supported with the `LinkDef.number` syntax, which identifies the link definition instance that is referenced.

   LinkDef.number.Source.Path: The location of relationship’s from elements. Only applicable to group hierarchies. There is no need to differentiate generated structures.

   LinkDef.number.Source.Property: The from element property to be matched against the Alarm Source property as specified in LinkDef.number.Alarm.Source.Property. These two values must be the same in order to establish a relationship with the source element.

   LinkDef.number.Target.Path: The location of relationship’s to elements. This tag is treated exactly the same as LinkDef.number.Source.Path.

   LinkDef.number.Target.Property: The to element property to be matched against the Alarm Source property as specified in LinkDef.number.Alarm.Source.Property.

   LinkDef.number.Alarm.Source.Property: The alarm property used to link to the relationship’s from element property as specified in LinkDef.number.Source.Property.

LinkDef.number.Alarm.Target.Property: The alarm property used to link to the relationship's to element property as specified in LinkDef.number.Target.Property.

LinkDef.number.Name: The relationship name. This name displays as a reference in the Operations Center Relationship Browser.

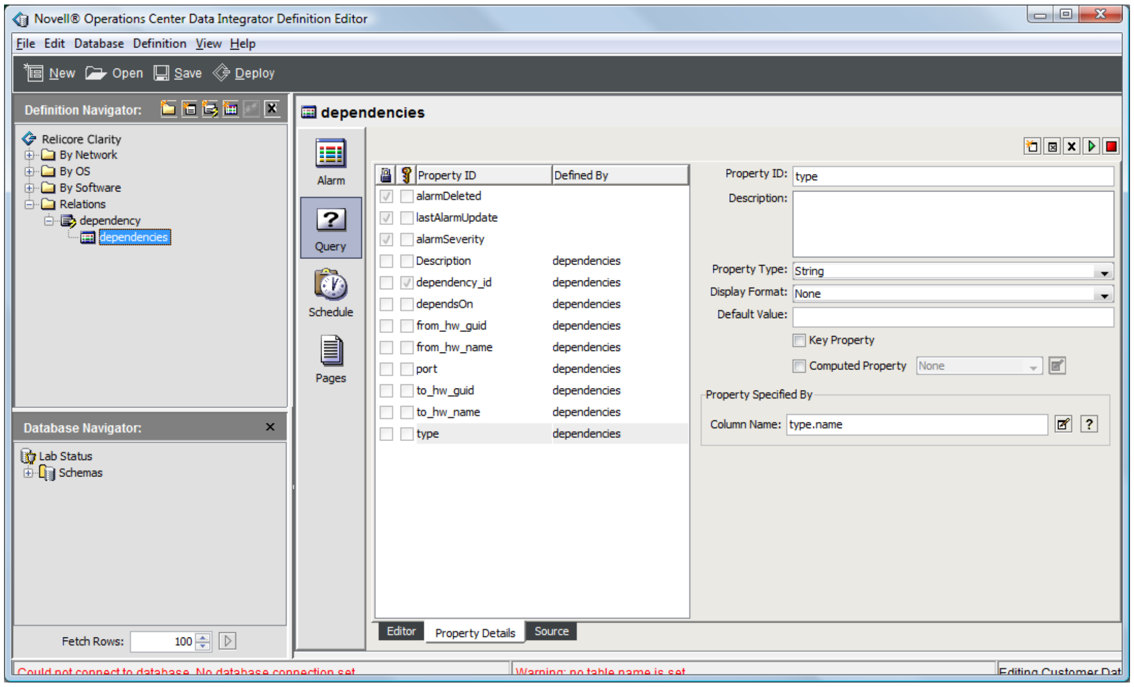
9.3 Example: Mining and Visualizing Relationships

This example shows how to match relationship objects with alarm data.

To match relationship objects with alarm data:

1 Identify the To and From structure locations for the relationships.

In this example, both are located in the same hierarchical structure (*By OS*), as shown in the following illustration:



Property ID	Defined By
alarmDeleted	
lastAlarmUpdate	
alarmSeverity	
Description	dependencies
dependency_id	dependencies
dependsOn	dependencies
from_hw_guid	dependencies
from_hw_name	dependencies
port	dependencies
to_hw_guid	dependencies
to_hw_name	dependencies
type	dependencies

Property ID: type
Description:
Property Type: String
Display Format: None
Default Value:
Key Property
Computed Property None
Property Specified By
Column Name: type.name

Could not connect to database. No database connection set
Warning: no table name is set
Editing Customer Dat

2 Look at the property definition for the hosts element, then identify `HARDWARE_GUID` as the property that can match relationship objects with alarm data.

3 Create an Alarm definition to populate relationship data as shown in the illustration in Step 1.

4 Create a Generator definition above the Alarm definition for debugging purposes and to visualize different classes or types of relationships that are found in the data source.

The Alarm definition includes the alarm properties *from_hw_guid* and *to_hw_guid*. These values are used to customize the hierarchy file.

The definition also populates *from_hw_name* and *to_hw_name* for debugging purposes to visually map related elements. It is important to include these properties in the Alarm Property Details to visualize the data. The relationship mapping works even if Alarm Property Details is not defined.

5 Edit the XML syntax in the hierarchy file to generate the relationship maps.

This instructs Data Integrator to put together all of the pieces.

This example uses the following XML code:

```
<param name="LinkDef.Count" value="1" />
<param name="LinkDef.1.Source.Path" value="By OS" />
<param name="LinkDef.1.Source.Property" value="HARDWARE_GUID" />
<param name="LinkDef.1.Target.Path" value="By OS" />
<param name="LinkDef.1.Target.Property" value="HARDWARE_GUID" />
<param name="LinkDef.1.Alarm.Source.Property" value="from_hw_guid" />
<param name="LinkDef.1.Alarm.Target.Property" value="to_hw_guid" />
<param name="LinkDef.1.Name" value="$alarm.port.replace(&quot;:&quot;,
&quot;|&quot;)" />
```

6 Enter the XML code in the *Hierarchy File* text area in the Group definition, as shown in the illustration in Step 1.

The Source Property as defined in the XML code is HARDWARE_GUID, which is a property defined in the Property Details for the Hosts Element definition.

A Data Integrator Demo Definition

The Data Integrator ships with a demonstration adapter definition that provides an example of how to leverage database information.

This sample adapter definition utilizes the Northwind database, a sample demonstration database that is shipped with Microsoft SQL Server. It sets up element and alarm definitions to pull out sample data in a hierarchy of customer and product inventory elements.

The illustration to the right shows the hierarchy tree of an adapter deployed from the demo adapter definition.

- Section A.1, "Configuring the Adapter Definition Demo," on page 87
- Section A.2, "Examining the Adapter Definition Demo," on page 89
- Section A.3, "Using the Deployed Adapter Definition Demo," on page 96

A.1 Configuring the Adapter Definition Demo

Before examining the actual demo definition setting, it is necessary to import the sample adapter definition and configure the database connection. The following sections explain this process:

- Section A.1.1, "Importing the Sample Adapter Definition," on page 87
- Section A.1.2, "Defining the Database Connection for the Sample Definition," on page 88

A.1.1 Importing the Sample Adapter Definition

To import the sample demo and demo adapter definition:

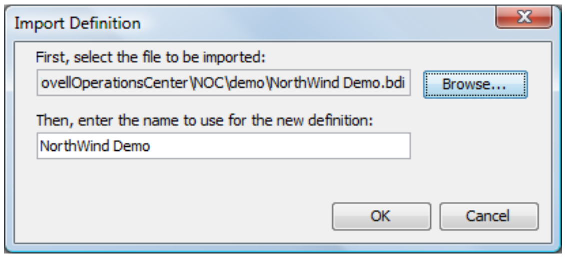
1. Copy the `/`*`OperationsCenter_install_path`*`/demo/NorthWind Demo.jar` file to the `/`*`OperationsCenter_install_path`*`/integrations` directory.
2. Copy the `/`*`OperationsCenter_install_path`*`/demo/NorthWind DemoHierarchy.xml` file to the `/`*`OperationsCenter_install_path`*`/integrations` directory.

3 In the *Explorer* pane, expand the *Administration* root element > *Adapters*.

4 Right-click *NOC - Data Integrator*, then select *Import Definition* to open the Import Definition dialog box.

5 Click *Browse*, then select the Data Integrator definition file from the `/`*`OperationsCenter_install_path`*`/demo`.

To find the sample adapter definition, browse to find the `/`*`OperationsCenter_install_path`*`/demo/northwind.bdi` file.


Import Definition
First, select the file to be imported:
Browse...
Then, enter the name to use for the new definition:
OK
Cancel

ovellOperationsCenter\NOC\demo\NorthWind Demo.bdi

NorthWind Demo

6 Specify the name to use for the new definition or use the northwind default.

7 Click *OK* to import the adapter definition demo.

The imported sample definition displays in the *Explorer* pane hierarchy tree under the *Administration* > *Adapters* > *Data Integrator* tree.

8 Because development database settings are stored with user preferences, redefine the database connection for the adapter definition.

For instructions, see Section A.1.2, "Defining the Database Connection for the Sample Definition," on page 88.

A.1.2 Defining the Database Connection for the Sample Definition

To define the database connection:

1 Right-click the sample adapter definition (Northwind, if you did not edit the name when importing), then select *Edit Definition* to open the Definition Editor.

2 In the Definition Editor, click *Database* > *Open Connection* to display the *Update Business Data Integration* wizard.

3 To define the database connection, select the *Define New Connection* radio button, then click *Forward*.

4 Specify the new name in the *Database Name* field, then click *Forward*.

5 Click the *SQL Server* tab.

6 Specify the database settings that point to the Northwind DB included with the SQL Server database.

7 Click *Finish* to save the settings and connect to the Northwind database.

8 View and explore settings in the Northwind sample definition.

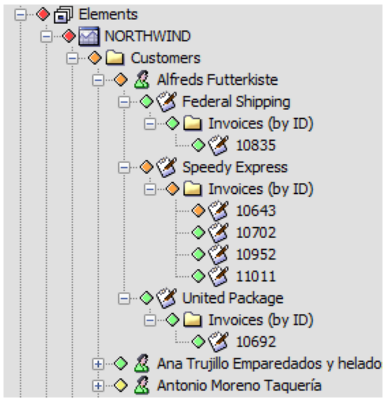
A.2 Examining the Adapter Definition Demo

The Northwind adapter definition comes preconfigured with all types of element and alarm definition. It utilizes the Northwind database to show how to expose database information.

In this example, when the Northwind adapter definition is deployed, it builds an adapter that has a hierarchy for both customers and product inventories.

Figure A-1 illustrates the customer hierarchy building out elements for invoices and groups by Shipment Type for each customer:

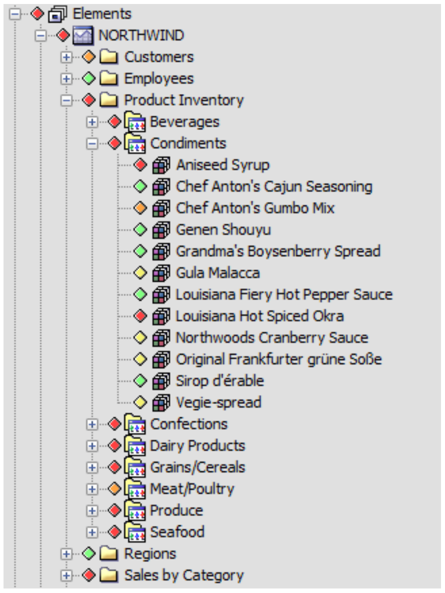
Figure A-1 *Explorer Pane by Shipment Type*



```
Elements
  NORTHWIND
    Customers
      Alfreds Futterkiste
        Federal Shipping
          Invoices (by ID)
            10835
        Speedy Express
          Invoices (by ID)
            10643
            10702
            10952
            11011
        United Package
          Invoices (by ID)
            10692
      Ana Trujillo Emparedados y helado
      Antonio Moreno Taquería
```

Figure A-2 illustrates the product inventory hierarchy building out elements for products and groups them by Product Type:

Figure A-2 *Explorer Pane by Product Type*



```
Elements
  NORTHWIND
    Customers
    Employees
    Product Inventory
      Beverages
      Condiments
        Aniseed Syrup
        Chef Anton's Cajun Seasoning
        Chef Anton's Gumbo Mix
        Genen Shouyu
        Grandma's Boysenberry Spread
        Gula Malacca
        Louisiana Fiery Hot Pepper Sauce
        Louisiana Hot Spiced Okra
        Northwoods Cranberry Sauce
        Original Frankfurter grüne Soße
        Sirop d'érable
        Vegie-spread
      Confections
      Dairy Products
      Grains/Cereals
      Meat/Poultry
      Produce
      Seafood
    Regions
    Sales by Category
```

The remainder of this section provides a high-level overview of specific examples from the demo adapter definition that created these various types of elements (in the deployed adapter) and their associated alarms.

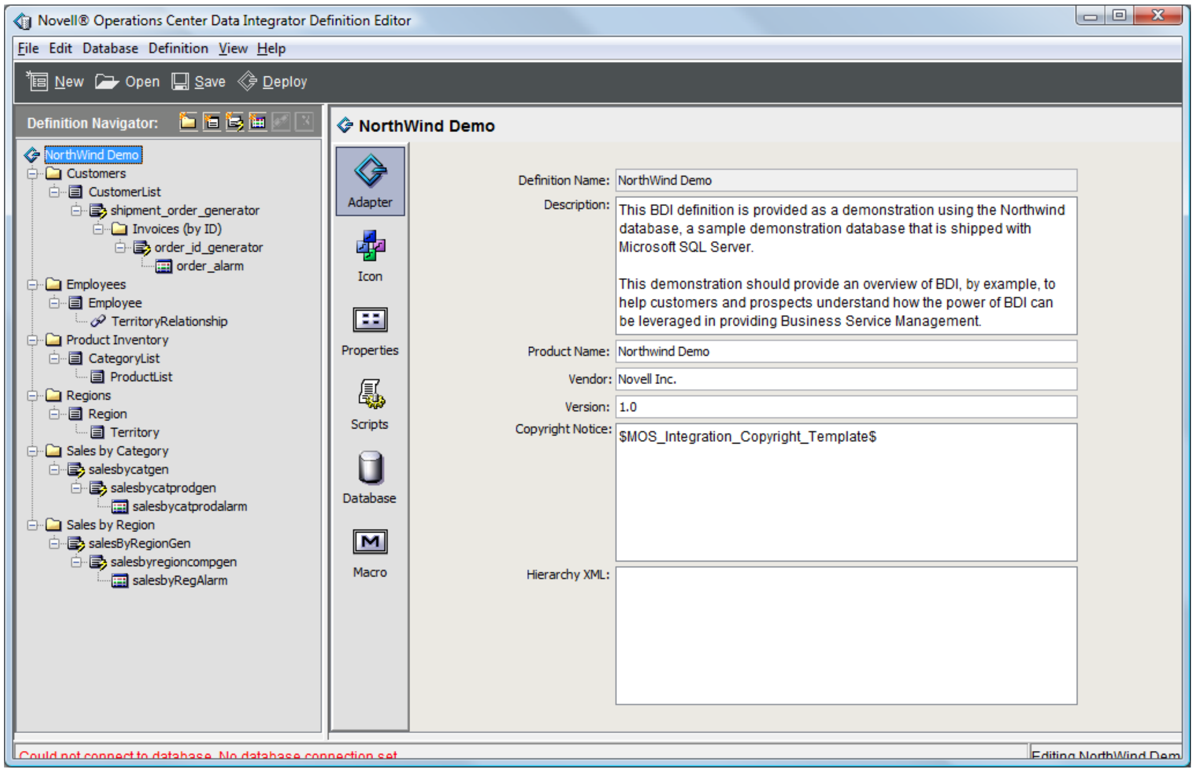
- Section A.2.1, “Looking at Basic Adapter Information,” on page 90
- Section A.2.2, “Looking at Element Groups,” on page 92
- Section A.2.3, “Looking at Database Elements,” on page 93
- Section A.2.4, “Looking at Generated Elements,” on page 94
- Section A.2.5, “Looking at Alarm Definitions,” on page 95

A.2.1 Looking at Basic Adapter Information

When an adapter definition is created, basic information and settings are defined for the adapter to use when deployed.

In Figure A-3, the run-time adapter leverages the Product Name, Vendor, Version, and Copyright Notice in the adapter properties:

***Figure A-3** Definition Editor*



Definition Name: NorthWind Demo

Description: This BDI definition is provided as a demonstration using the Northwind database, a sample demonstration database that is shipped with Microsoft SQL Server.

This demonstration should provide an overview of BDI, by example, to help customers and prospects understand how the power of BDI can be leveraged in providing Business Service Management.

Product Name: Northwind Demo

Vendor: Novell Inc.

Version: 1.0

Copyright Notice: $MOS_Integration_Copyright_Template$

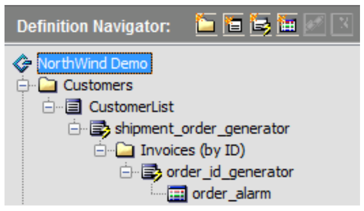
Hierarchy XML:

Could not connect to database. No database connection set

Editing NorthWind Dem

To examine the adapter settings example:

1. In the Definition Editor, select the root element in the *Definition Navigator* pane (in this instance, it is the *northwind* element).



2. Click the following icons to see how the adapter icon, default properties, scripts, and databases are set:

   Adapter: Defines basic descriptive information for the adapter, including copyright information.

   Icon: Allows assigning an icon for the root adapter element in the run-time adapter.

   Properties: Defines defaults and permissions for alarm columns, hierarchy file, stylesheet file, and maximum number of alarms for the adapter.

   Scripts: Defines defaults and permissions for any scripts to be run with adapter activity.

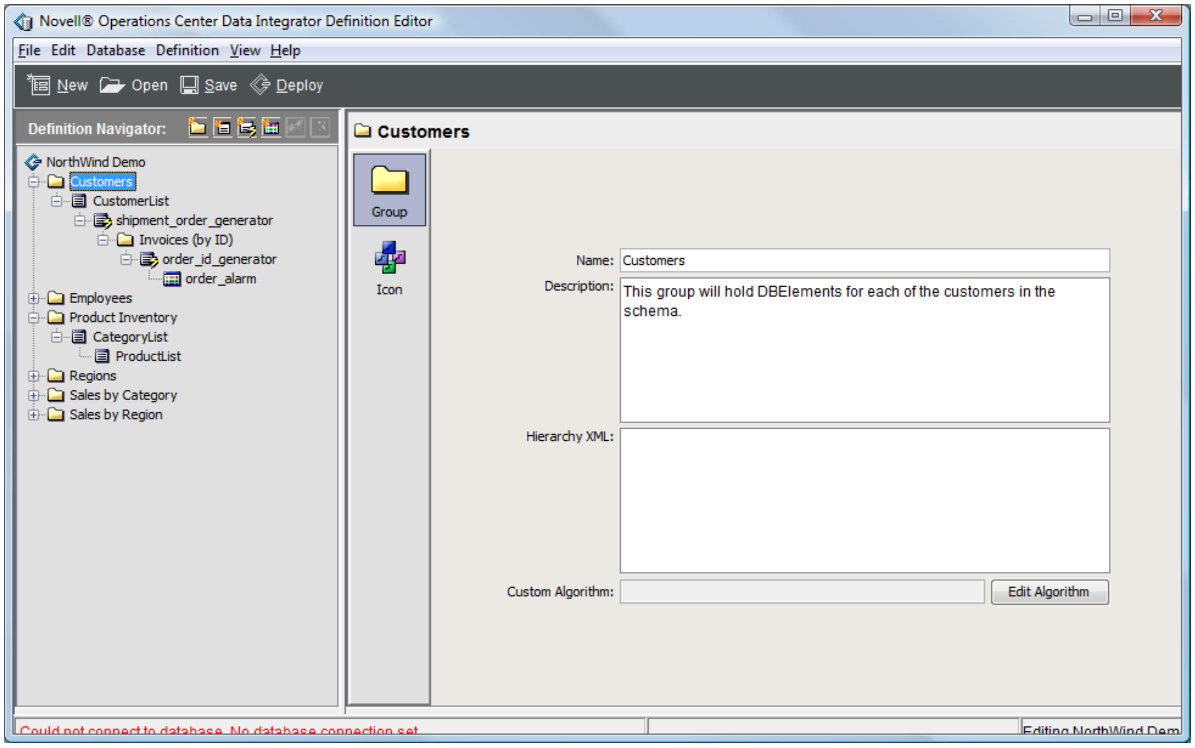
   Database: Defines database information and permissions for the database connection used by the run-time adapter.

A.2.2 Looking at Element Groups

Element Group is a hard-coded folder (not dynamically created) that might hold children and alarms in the deployed adapter.

In Figure A-4, the run-time adapter uses the Group definition to create a single Operations Center element (or folder) named *Customers*:

***Figure A-4** Definition Editor Group Definition Tab*



Name: Customers

Description: This group will hold DBElements for each of the customers in the schema.

Hierarchy XML:

Custom Algorithm:

Could not connect to database. No database connection set.

Editing NorthWind Dem

In the Northwind demo, there are several examples of Element Groups, which include Customers, Invoices (by ID), and Product Inventory.

To examine an element group example:

1 In the Definition Editor, click to open and browse the hierarchy in the *Definition Navigator* pane.

2 Select the *Customers* folder.

This is an element group definition.

3 Click the *Group* and *Icon* icons to see how group elements (folders) are created and assigned icons in the demonstration definition.

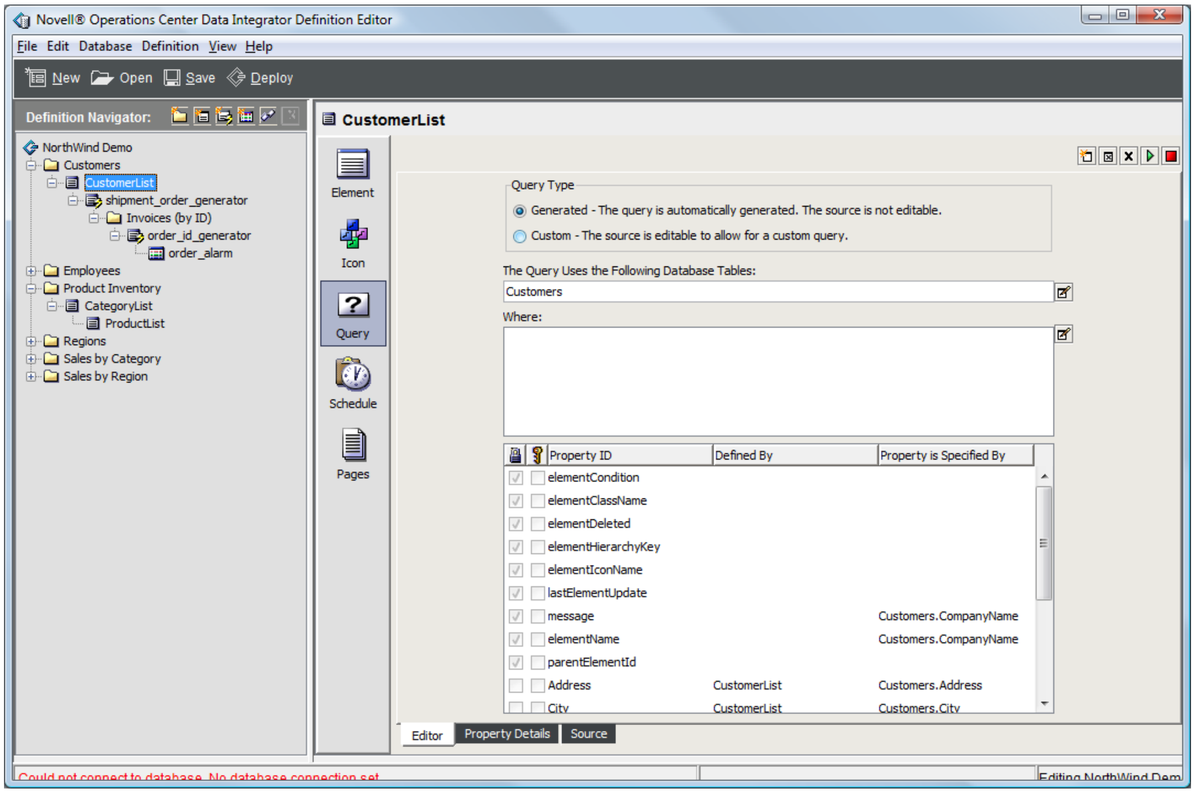
The *Group* pane defines basic information about the *Element Group* definition. In this case, the name specified for this element is used for this element in the deployed adapter.

The *Icon* pane allows assigning an icon that is used for this parent element (folder) in the deployed adapter.

A.2.3 Looking at Database Elements

In Figure A-5, the run-time adapter uses the Database Element definition to create Operations Center elements for all unique customer names found in the Customers table:

Figure A-5 Definition Editor: Database elements are displayed in the Definition Navigator pane.


Novell® Operations Center Data Integrator Definition Editor
File Edit Database Definition View Help
New Open Save Deploy
Definition Navigator:
Element
Icon
Query
Schedule
Pages
Editor
Property Details
Source

- NorthWind Demo
  - Customers
    - CustomerList
      - shipment_order_generator
        - Invoices (by ID)
          - order_id_generator
            - order_alarm
  - Employees
  - Product Inventory
    - CategoryList
      - ProductList
  - Regions
  - Sales by Category
  - Sales by Region

CustomerList

Query Type
[✓ Generated - The query is automatically generated. The source is not editable.]
Custom - The source is editable to allow for a custom query.

The Query Uses the Following Database Tables: Customers

Where:

		Property ID	Defined By	Property is Specified By
✓		elementCondition		
✓		elementClassName		
✓		elementDeleted		
✓		elementHierarchyKey		
✓		elementIconName		
✓		lastElementUpdate		
✓		message		Customers.CompanyName
✓		elementName		Customers.CompanyName
✓		parentElementId		
		Address	CustomerList	Customers.Address
		City	CustomerList	Customers.City

Could not connect to database. No database connection set.

Editing NorthWind Dem

The Northwind demo has several examples of database elements, including *CustomerList*, *CategoryList*, and *ProductList*.

To examine a database elements example:

1 In the Definition Editor, click to open and browse the hierarchy tree in the *Definition Navigator* pane.

2 Select the *CustomerList* element.

This is a database element definition. With the deployed adapter, it instructs the creation of an element under the *Customers* folder for each unique value found in the *CustomerName* column of the Customers table.

3 Click the following icons to see how database elements are defined in the demonstration definition:

Element: Defines basic information about the database element definition. In this case, the name and description are only used within the adapter definition and the Definition Editor.

Icon: Assigns an icon for these elements in the deployed adapter.

Query: Defines the query that pulls database information to create these elements. It is also used to define various properties for the elements.

Schedule: Sets up a schedule for polling the database for new information with the query specified in the *Query* tab.

Pages: Defines one or more property pages for the element using the properties available in the *Query* tab.

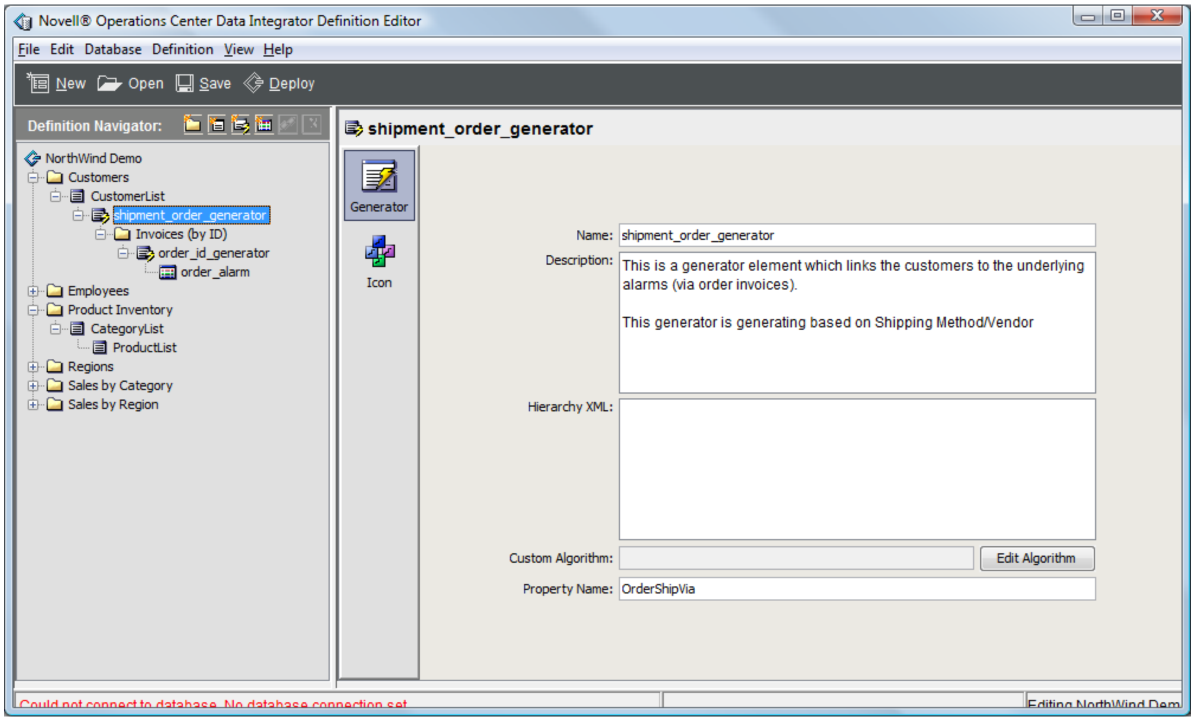
A.2.4 Looking at Generated Elements

A generated element is created in the adapter definition when you want to create parent elements directly from a table column in the database.

It is important, however, that the associated Property Name value exactly matches a property created from a query of a child DB Element or Alarm Definition.

In Figure A-6, the run-time adapter uses the generator definition to create a Operations Center element for all values found in the table column that identifies the shipping method used to send the order:

Figure A-6 *Definition Editor: Operations Center elements can be generated from database table columns.*



Name: shipment_order_generator

Description: This is a generator element which links the customers to the underlying alarms (via order invoices).

This generator is generating based on Shipping Method/Vendor

Hierarchy XML:

Custom Algorithm:

Property Name: OrderShipVia

Could not connect to database. No database connection set.

Editing NorthWind Dem

In the Northwind demo, two examples of generated elements are *shipment_order_generator*, and *order_id_generator*.

To examine an element group example:

1. In the Definition Editor, click to open and browse the hierarchy in the *Definition Navigator* pane.
2. Select the *shipment_order_generator* folder.

   This is a generator definition.

3 Click the *Generator* and *Icon* icons to see how generators are created and icons are assigned in the demonstration definition.

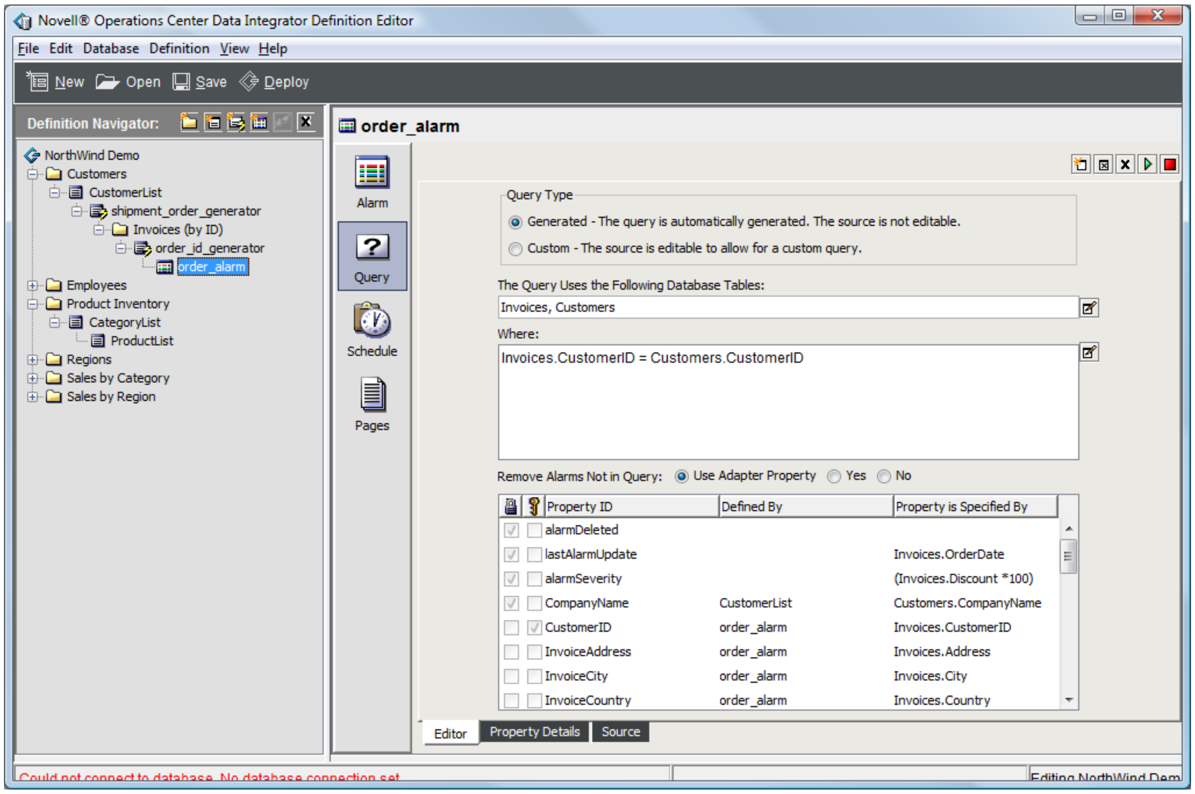
The *Generator* pane defines basic information about the generator definition. The *Name* and *Description* values are used only by the adapter definition and the Definition Editor. The *Property Name* value must directly match a property created from a query of a child *Database Element* or *Alarm* definition.

The *Icon* pane is used to assign an icon to the elements created by the *Generator Definition* by the deployed adapter at run-time.

A.2.5 Looking at Alarm Definitions

Figure A-7 illustrates that alarm definitions can be created in the adapter definition when you want to have alarms generated for an element from database information:

***Figure A-7** Definition Editor with Alarm Definitions*



Query Type

[✓ Generated - The query is automatically generated. The source is not editable.]
Custom - The source is editable to allow for a custom query.

The Query Uses the Following Database Tables: Invoices, Customers

Where:

```
Invoices.CustomerID = Customers.CustomerID
```

Remove Alarms Not in Query: [✓ Use Adapter Property] Yes No

🔒	🔑	Property ID	Defined By	Property is Specified By
✓		alarmDeleted		
✓		lastAlarmUpdate		Invoices.OrderDate
✓		alarmSeverity		(Invoices.Discount *100)
✓		CompanyName	CustomerList	Customers.CompanyName
	✓	CustomerID	order_alarm	Invoices.CustomerID
		InvoiceAddress	order_alarm	Invoices.Address
		InvoiceCity	order_alarm	Invoices.City
		InvoiceCountry	order_alarm	Invoices.Country

Could not connect to database. No database connection set.

Editing NorthWind Dem

In the Northwind demo, one example of an alarm definition is *order_alarm*.

To examine a database elements example:

1 In the Definition Editor, click to open and browse the hierarchy in the *Definition Navigator* pane.

2 Select the *order_alarm definition* folder.

This is an alarm definition. With the deployed adapter at run-time, it instructs the creation of alarms for the elements created by *order_id_generator*.

3 Click the following icons to see how database elements are defined in the demonstration definition:

   Alarm: Defines basic information about the Alarm definition. In this case, the name and description are only used within the adapter definition and Definition Editor.

   Icon: Allows assigning an icon for these alarms in the deployed adapter.

   Query: Defines the query that pulls database information to create alarms for the corresponding element. It is also used to define various properties for the alarms.

   Schedule: Sets up a schedule that is used to poll the database for new information using the query specified in the *Query* tab.

   Pages: Defines one or more property pages for the alarm using those properties available from the *Query* tab.

A.3 Using the Deployed Adapter Definition Demo

Operations Center is preconfigured with an adapter type built directly from the Northwind definition demo.

To view how the demo definition builds a deployed adapter, do one of the following:

- In the Operations Center console, under *Administration* > *Adapters*, create an adapter, then select *Adapter: NorthWind Demo*.

  Be sure to specify correct values for the Northwind DB found in the running version of SQL Server.
- Deploy the imported adapter definition and create a new adapter from it.

  This might be the best way to practice building element and alarm definitions, then deploy them.