SeeBeyond ICAN Suite™

eBAM Studio User's Guide

Release 5.0

S E E B E Y O N D®

Contents

List of Figures

List of Tables

Introduction

BAM (Business Activity Monitoring) involves the collection, aggregation, and presentation of business activity data according to specified Key Performance Indicators (KPIs). eBAM Studio (eBAM) provides the tools for generating custom cross-application digital dashboards for defining and monitoring KPIs that summarize the aggregated business data collected through the eInsight or eGate application layers. The eBAM Web interface allows business analysts to transform data that has been collected over time into meaningful, rich visual presentations.

KPIs provide a context for business processes by turning raw data into useful information, allowing the business analyst to focus on monitoring and analyzing measurable operations and processes across the enterprise. eBAM Studio provides the business analyst with different views of performance data, enabling the timely identification of business trends.

eBAM Studio renders real-time and historical data in familiar visual formats, such as pie and bar charts, for display in digital dashboards. These recognizable, easy-to-read contexts enable the business analyst to quickly translate information into action.

1.0.1 Document Purpose and Scope

The *eBAM Studio User's Guide* explains how to use eBAM to transform collected data into a visual context for accessibility over the Web. By leveraging the Web, key business information can be made accessible in a visual context across the enterprise to provide visibility, reporting, and analysis.

1.1 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning ICAN Suite system. This person must also understand any operating systems on which eGate will be installed (Windows or UNIX) and must be thoroughly familiar with Web browsers and Windows-style GUI operations.

1.2 Organization of information

The *eBAM Studio User's Guide* includes the following information:

- An overview of eBAM's application architecture.

- Prerequisites and installation instructions.

- An overview of eBAM's major user interfaces.

- A detailed description of eBAM's features, with step-by-step instructions to guide you through their setup and configuration.

- How to use eBAM, step-by-step, in a sample implementation that collects run-time data, analyzes it, and displays Key Performance Indicators as specified in the setup.

1.3 Writing Conventions

The following writing conventions are observed throughout this document.

Table 1 Writing Conventions

Text	Convention	Example
Button, file, icon, parameter, variable, method, menu, and object names.	**Bold** text	- Click **OK** to save and close. - From the **File** menu, select **Exit**. - Select the **logicalhost.exe** file. - Enter the **timeout** value. - Use the **getClassName()** method. - Configure the **Inbound** File eWay.
Command line arguments and code samples	`Fixed` font. Variables are shown in ***bold italic***.	`bootstrap -p ` ***`password`***
Hypertext links	**Blue** text	For more information, see **"Writing Conventions" on page 9**.

Additional Conventions

Windows Systems

For the purposes of this guide, references to "Windows" will apply to Microsoft Windows Server 2003, Windows XP, and Windows 2000.

Path Name Separator

This guide uses the backslash ("\") as the separator within path names. If you are working on a UNIX system, please make the appropriate substitutions.

1.4   Online Documentation

The documentation for the SeeBeyond ICAN Suite is distributed as a collection of online documents. These documents are viewable with the Acrobat Reader application from Adobe Systems. Acrobat Reader can be downloaded from:

http://www.adobe.com

When downloading Acrobat Reader, make sure to download the version that includes the option for searching **.pdf** files—Acrobat Reader with Search. This version is required to view the searchable master index.

1.5   The SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

About eBAM Studio

eBAM Studio enables the creation of business activity monitoring applications that intercept the flow of data through ICAN Suite components to produce visual presentations of data analysis based on Key Performance Indicators (KPIs) and alerts.

This chapter provides overviews of eBAM's architecture and overall approach to Business Activity Monitoring.

2.1 Architectural Overview

eBAM takes advantage of the overall J2EE architecture of the ICAN Suite to work hand-in-glove with other ICAN products. As Figure 1 suggests, eBAM can takes advantage of eGate and eInsight to leverage your investment in your other ICAN products, such as eVision and ePortal.

Figure 1 eBAM's Relationship to the ICAN Suite
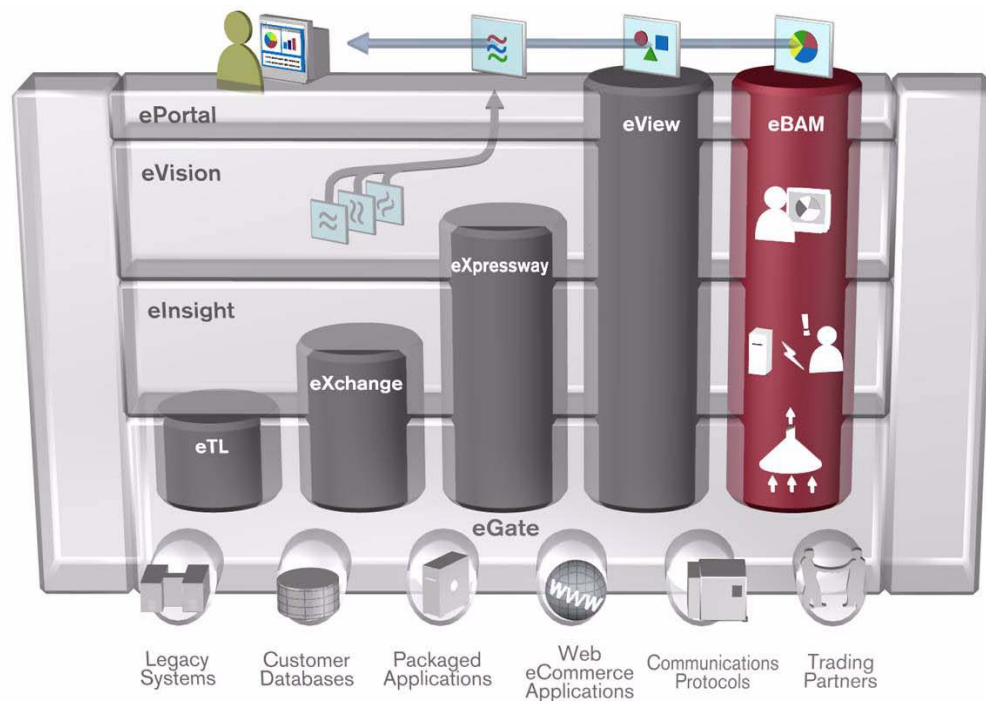
2.1.1 eBAM Architecture in Conjunction With ICAN

Here are some ways that eBAM leverages other ICAN Suite products and features:

- The Repository stores and controls check-out/check-in and ACL access to eBAM-specific objects such as data definitions, chart configurations, and alert settings.

- Integration Server security provides single-sign-on functionality at design time, and also provides runtime resource management.

- The Enterprise Designer GUI is leveraged for the design of the data collection, graphical display, and notification processes, and the Enterprise Manager GUI allows eBAM to plug in its additional reporting and monitoring functionality.

- The Business Process Modeler is used for alert handling and graphic generation.

- Common deployment experience enabled via ePortal as a natural place for viewing all eBAM applications.

2.1.2 eBAM Data Flow

To allow for task differentiation, scalability, tunability, and maintainability/robustness, the eBAM data flow consists of a stack of separable layers. See Figure 2.

- The *presentation layer* provides a front-end GUI that allows business analysts to see only as much or as little as they decide to see, using the indicators they find meaningful, presented in the fashion they find most congenial.

- The *tracking layer* collects, aggregates, and filters data, passing periodic updates of the information-of-interest upwards to the presentation layer.

- The *messaging layer* communicates with external systems, mediated by eGate's JMS and Collaborations and eInsight's Business Processes.

Figure 2   eBAM's Stacked Layers for Task Differentiation
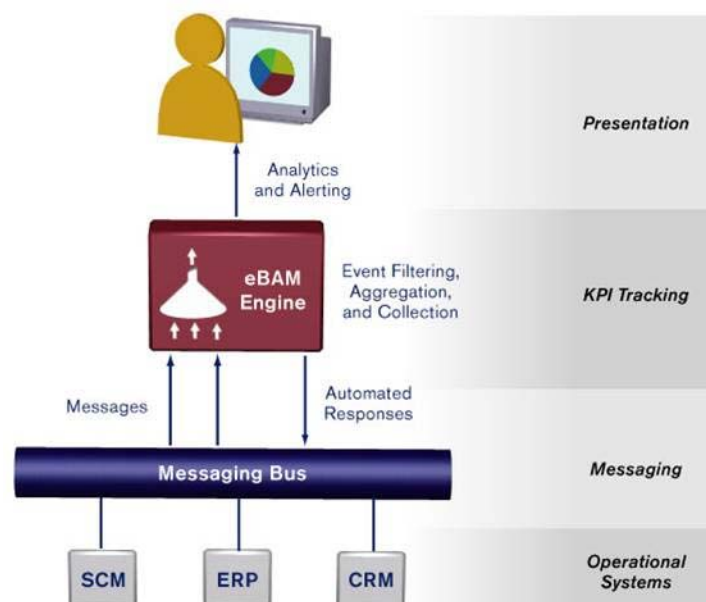
2.2 eBAM Means Business Activity Monitoring

eBAM takes a simple approach to business activity monitoring: Applications and KPIs.

- Each eBAM *Application* samples the ICAN data traffic and stores results in a way that allows standing queries to update and communicate their findings.

- Each eBAM *KPI* queries and filters the results and communicates them to the GUI. The GUI provides the data view that best suits the user or users who need to know what's happening in terms of the big picture, or trends, or anomalies.

2.2.1 What Is Inside an eBAM Application?

An eBAM application requires access to ICAN run-time resources, from which it gathers raw data and processes it (via user-configured aggregation and presentation) to render it into a meaningful format. An eBAM application requires the following:

- Access to run-time data from ICAN suite processes.

- User definitions for Key Performance Indicators.

- User definitions for thresholds and conditions—in other words, user-specified conditions that trigger an eBAM response.

- User-defined actions to take for Alerts and Notifications that are triggered when threshold boundaries are crossed.

- Graphical dashboard presentations—data presented in charts and graphs

- User-defined timeframes, such as frequency and starting and ending dates+times.

eBAM ties into raw data via JMS, eGate, and Web Services to perform real-time analysis on ICAN Suite composite applications.

2.2.2 What Is In a KPI?

An eBAM KPI requires that you specify the following attributes:

- Data points (the facts that form the basis for the calculation of a KPI value)

- The mathematical formula for calculating the KPI value.

- The dashboard (visual) representation of the KPI.

- The events that are triggered when the value of the KPI crosses a preset threshold.

eBAM provides an intuitive palette of graphical tools for constructing queries of the stored data.

Installing eBAM

This chapter provides the prerequisites and steps for installing eBAM Studio.

3.1 System Requirements

This section lists requirements for operating systems and prerequisite products. The Readme.txt file on the product media contains the most up-to-date information on system requirements for each supported platform.

3.1.1. Platform Support

eBAM supports the following operating systems:

- Microsoft Windows Server 2003, Windows XP SP1a, and Windows 2000 SP3 or SP4
- Sun Solaris 8 and 9 with required patches
- IBM AIX 5.1 and 5.2 with required maintenance level patches
- HP-UX 11.0 and 11i with required patches
- HP Tru64 V5.1A with required patches
- Red Hat Linux 8 (Intel) and Linux Advanced Server 2.1 (Intel version)

3.1.2. Prerequisite Products

The prerequisite products for installing eBAM 5.0 are: A **Repository** at release **5.0.1** or later, and *one or the other* of the following:

- **eGate Integrator** with **eInsight Integrator**, both at release **5.0.1** or later.
- **eInsight Enterprise System Bus** (ESB) at release **5.0.1** or later.

Because of the APIs and GUIs that eBAM requires, eGate.sar (or eInsightESB.sar) must be uploaded *before* uploading eBAM.sar; for details, see the following section.

If you want to run the sample project (provided in **eBAMDocs.sar**) or follow all of the implementation steps provided in **Chapter 5**, then the **File eWay** at **5.0.1** or later is also required.

Complete instructions on uploading and downloading **.sar** files and sample files are provided in the following section.

3.2 Installation Steps

The steps for installing eBAM are the same as for other products in the ICAN Suite. You can find general product installation instructions in the *ICAN Suite Installation Guide,* available on the product media and also accessible via Enterprise Manager (Documentation tab).

3.2.1. Uploading eBAM to the Repository

Before you begin

- A Repository server must be running on the machine where you will be uploading the eBAM product files, and a license file (license.sar) and prerequisite product files must have already been uploaded to this Repository.

To upload eBAM product files to the Repository

1 On a Windows machine, start a Web browser and point it at the machine and port where the Repository server is running:

```
http://<hostname>:<port>
```

where

- *<hostname>* is the name of the machine running the Repository server.
- *<port>* is the starting port number assigned when the Repository was installed.

For example, the URL you enter might look like either of the following:

```
http://localhost:12001
http://serv1234.company.com:19876
```

2 On the Enterprise Manager **SeeBeyond Customer Login** page, enter your username and password.

3 When Enterprise Manager responds, click the **ADMIN** tab. See Figure 3.

Figure 3 Enterprise Manager ADMIN Page
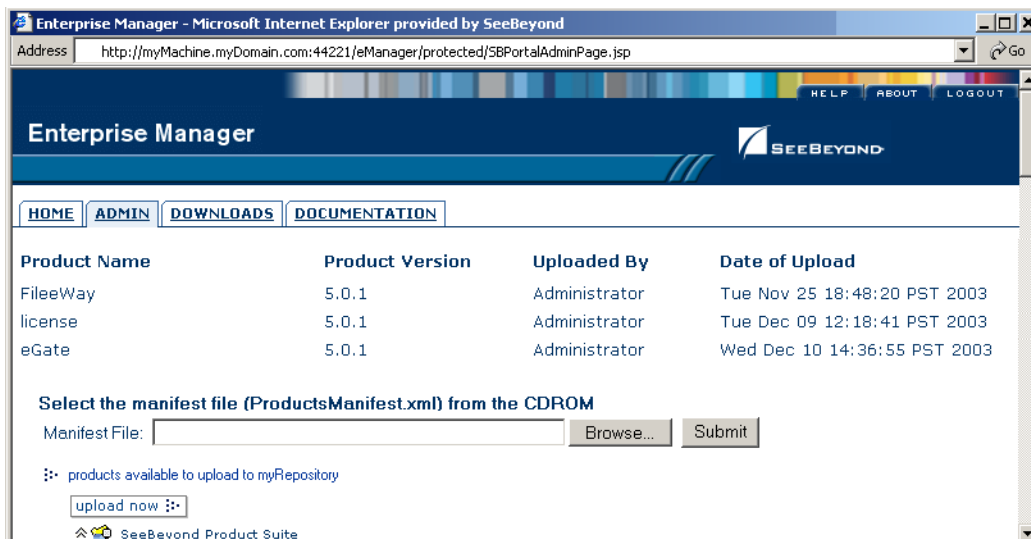
4   In the ADMIN page, click **Browse**.

5   In the **Choose file** dialog, click **ProductsManifest.xml**, and then click **Open**.

6   In the ADMIN page, click **Submit**.

After the manifest uploads, the lower half of the ADMIN page lists the product files you are licensed to upload to this Repository.

7   In the Products column, find **eBAM**, and then click the **Browse** button for it.

8   In the **Choose file** dialog, click **eBAM.sar**, and then click **Open**.

9   Repeat the previous two steps for **eBAMDocs.sar** and (if you haven't already uploaded it, and if you will be doing the sample implementation described in **Chapter 5**) **FileeWay.sar**.

Note: *eBAMDocs.sar contains documentation and sample files. The File eWay is used in the sample implementation to read data from the files and write output.*

10   In the ADMIN page, click the | upload now ⁝∙ | button.

11   After the files upload, in Enterprise Manager, click the DOCUMENTATION tab.

12   In the DOCUMENTATION page, under **Products** (on the lower left), click **eBAM Studio**; then, under eBAM Studio (on the lower right), click **Download Sample**. See Figure 4.

Figure 4   DOCUMENTATION Page: Downloading eBAM Sample Files
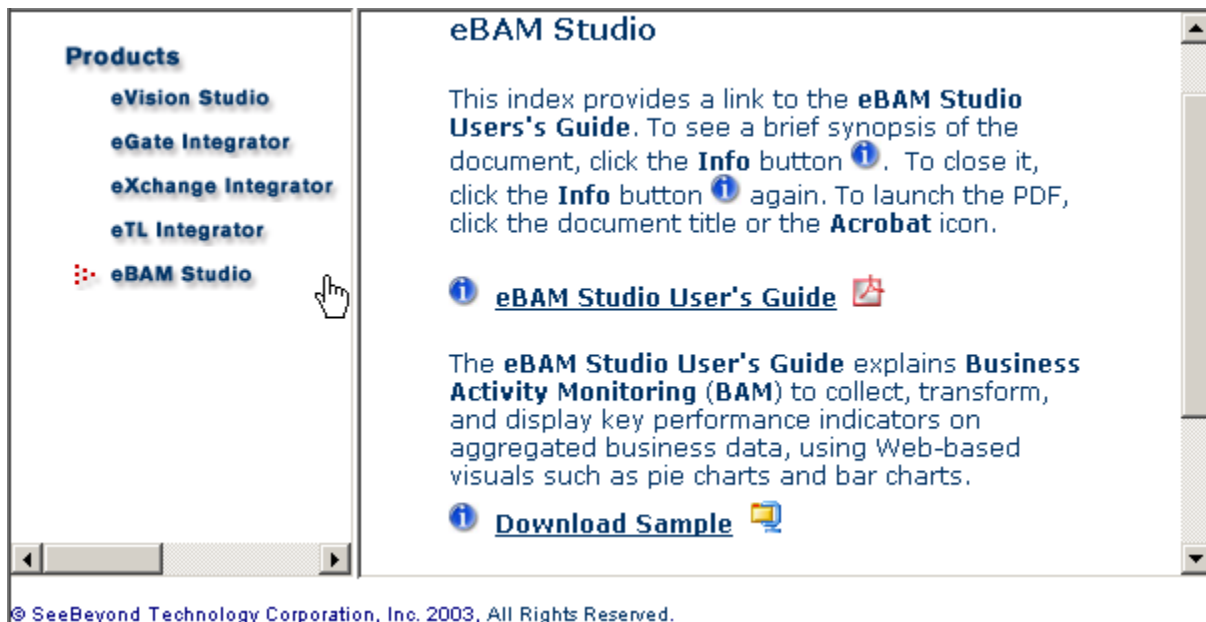
Result: Your repository can now serve the files in eBAM.sar to any Enterprise Designer that connects to it and uses the Update Center.

3.2.2. **Updating Enterprise Designer with eBAM**

Before you begin

- You must have already downloaded and installed Enterprise Designer.
- A Repository server must be running on the machine where you uploaded the eBAM product files.

To refresh an existing installation of Enterprise Designer

1   Start Enterprise Designer.

2   On the **Tools** menu, click **Update Center**.

The Update Center shows a list of components ready for updating. See Figure 5.

Figure 5   Update Center Wizard: Select Modules to Install
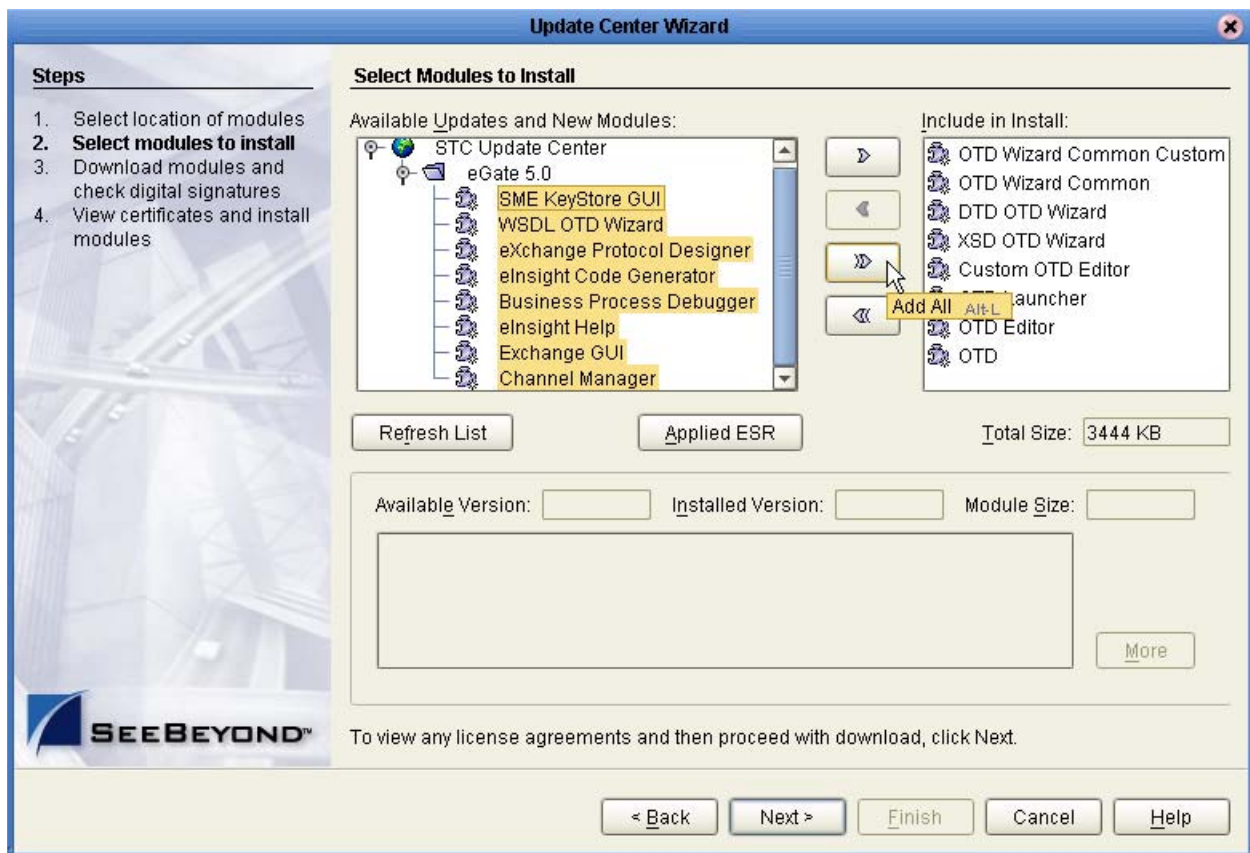
3   Click **Add All** (the button with a doubled chevron pointing to the right).

All modules move from the Available/New pane to the **Include in Install** pane.

4   Click **Next** and, in the next window, click **Accept** to accept the license agreement.

The wizard shows you the progress of the download. See Figure 6.

Figure 6   Update Center Wizard: Progress Bars
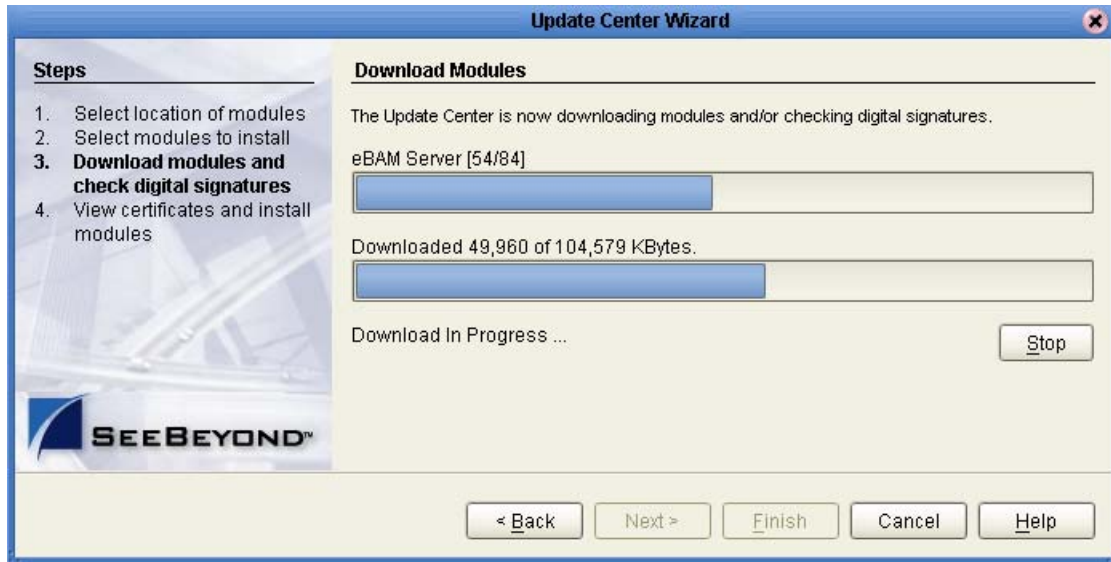
5   When the progress bars indicate the download has ended, click **Next**.

6   Review the certificates and installed modules, and then click **Finish**.

7   When prompted to restart Enterprise Designer, click **OK**. See Figure 7.

Figure 7   Update Center Wizard: Restart Enterprise Designer
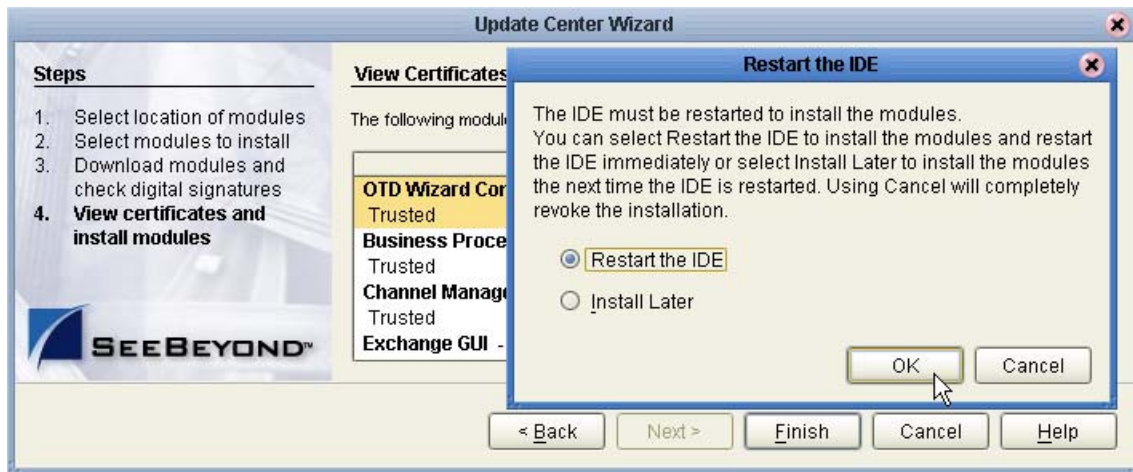
Result: When Enterprise Designer restarts, the installation of eBAM Studio is complete, and you can use all eBAM tools provided on the Enterprise Designer and Enterprise Manager frameworks.

Using eBAM Studio

This chapter provides information on the eBAM features and step-by-step instructions for using them.

The instructions in this chapter assume you are already familiar with the Repository and Logical Host, and focuses on the design-time GUI operations specific to eBAM.

4.1 eBAM Applications

Each eBAM Application consists of exactly one data definition, zero or more charts, and zero or more alert/notification conditions; the charts and alert conditions in an application are based on its data definition.

- The *data definition* is the infrastructure of the application. In it, you specify which data you are interested in, and how to label and organize the data. For further information and step-by-step procedures, see **"Setting Up the Data Definition for an eBAM Application" on page 19**.

- A*lert conditions* are triggered whenever a condition is met by one of the data items, such as exceeding a threshold. For further information and step-by-step procedures, see **"Setting Up the Application's Alert Conditions" on page 28**.

- *Charts* provide real-time feedback on the current data set according to conditions you set up. For further information and step-by-step procedures, see **"About Charts" on page 32**. and **"Setting Up the Application's Charts" on page 35**.

4.1.1 Setting Up the Data Definition for an eBAM Application

Before you begin: You must have a clear idea of the data you want to gather before you create the application. You should know the answers to the following questions:

- Altogether, how many data elements should be defined, and of what data type? (Each data element must be of type **char**, **float**, **integer**, **timestamp**, or **varchar**.)

- For each data element, what is the best name? (Each data element within an application must have a unique name; this is the name by which it is manipulated by the queries for constructing conditions and charts, but is not necessarily the label that appears on the chart itself.)

- For each data element, what should be the default value? (If not specified, the default default is a null value of the appropriate type.)

In other words, be prepared with information whose form is similar to that of Table 2:

Table 2 Metadata for Creating a Data Definition

	Name	Default Value	Data Type
1	OrderNum	999999999	integer
2	DateOfOrder	2099-12-31 23:59:59	timestamp
3	CustName1	!! Customer Name Must Be Supplied !!	varchar
(...)	(...)	(...)	(...)
n	TotalDollarsThisOrder		float

Tip: Data definitions, once created, cannot be modified. Therefore, when you reach the end of the wizard, be sure to review your definition carefully before clicking Finish.

To create a new eBAM Application and specify its data definition

1 In Enterprise Designer, in the project tree (left side), right-click the project and, on the popup context menu, point at New and click **eBAM Application**.

The eBAM Application wizard opens, prompting you to name the application.

2 In step 1 of the wizard, type in a name for application, and then click **Next**.

3 In step 2 of the wizard, type in a name for the data definition, and then click **Next**.

4 In step 3 of the wizard, click **Add** several times, and then supply names, default values, and data types for the data elements.

Tip: If the order of the elements is important to you, double-check frequently to ensure you have not omitted a row—clicking **Add** appends each new row to the end. (However, order is unimportant to the queries, and some of the GUIs alphabetize by field name.)

5 As needed, click **Add** to append additional rows or click **Remove** to remove unneeded rows. When done, click **Next**. See Figure 8.
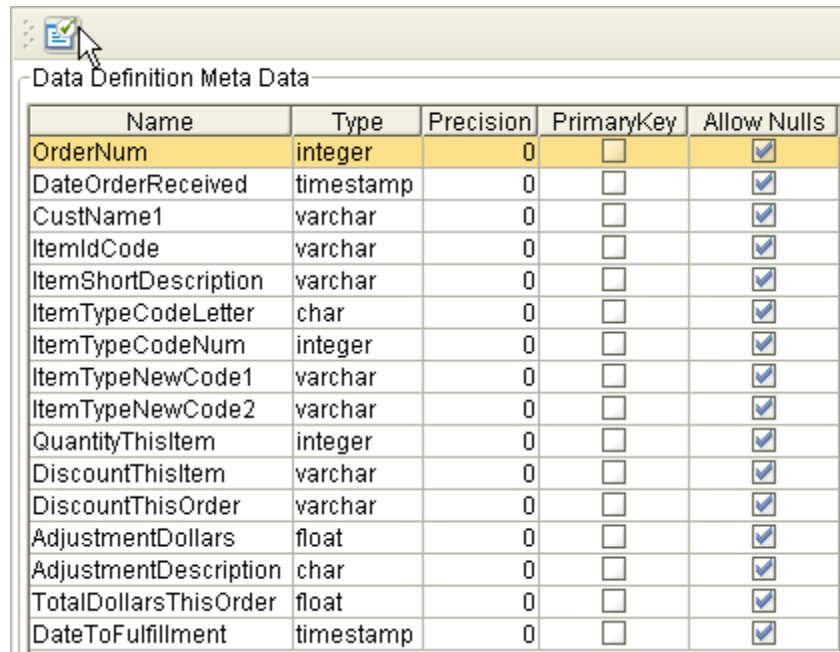
Figure 8   Setting Up a Data Definition
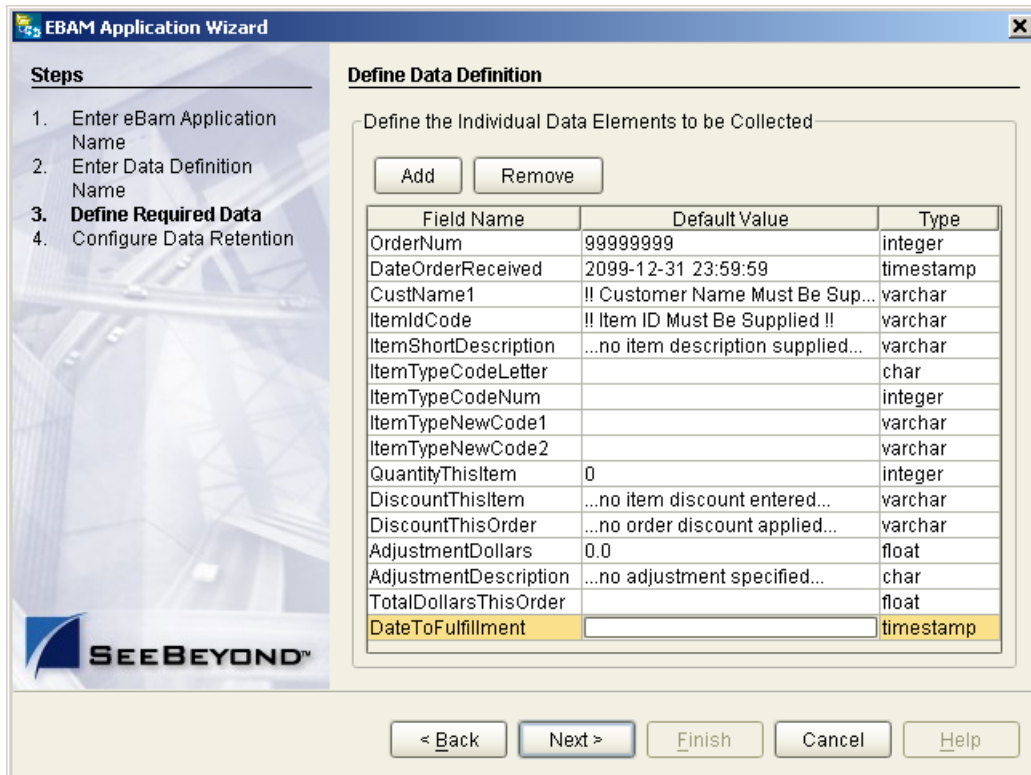
6   Review the completed data definition carefully before continuing. As needed, click Back and Next to return to a step and make changes.

7   Click **Finish** only after you are completely satisfied with the data definition.
After you click Finish, you will be unable to make further changes to this data definition.

Result: On the left side, the project tree displays the new eBAM application. On the right side (the canvas), the eBAM Editor opens to display the data definition; see Figure 9.

Figure 9   eBAM Editor Showing Completed Data Definition

Tip: Any time you complete a new data definition, it is good practice to use the
Show Sample Data tool to double-check the data definition against sample data.

To validate the metadata definition

1   In the eBAM Editor, on the tool palette, click: **Show Sample Data**

   The Sample Data pane appears below the main eBAM Editor.

2   On the Sample Data tool palette, click: **Import**

3   In the File Import wizard's **Select File** step, browse to the location of the sample
   data file and select it.

4   In the formatting step, make appropriate choices (see Table 3), and then click Next.

Table 3   Specifying Formatting Type and Encoding for Imported Sample Data

Item	Choices	Notes
Table name		Must start with letter, and contain only letters, numbers, and underscores
Encoding scheme	**ASCII (ISO646-US)** UTF-8 EBCDIC: USA-Canada (cp037)	**ASCII** — 7-bit encoding, roman characters. UTF-8 — 8-bit encoding of Unicode. EBCDIC — for certain mainframe systems
File format	Delimited Fixed-width	Delimited files specify escape characters to distinguish fields in a record and one record from the next; fixed-width files specify a preset length for each record.

5   In the parsing step, make appropriate choices for delimited data (see Table 4) or fixed-width data (see Table 5), and then click Next.

Table 4   Parsing Information for Imported Sample Delimited Data

Item	Value	Notes
Default SQL Type	Any of the following: bigint, bit, char, date, decimal, double, longvarchar, numeric, real, smallint, time, timestamp, tinyint, **varchar**	
Record Delimiter	**{newline (lf)}** {cr} {cr-lf}	**newline** — each new line starts a new record. cr — each carriage-return starts a new record. cr-lf — each carriage-return+linefeed starts a new record.
	You can type also type in a character or control character; for example, to specify TAB, type in: **\t**	
Field	**{comma}** {tab} {pipe}	**{comma}** — each instance of , starts a new field. {tab} — each tab character starts a new record. {pipe} — each instance of | starts a new field.
Text Qualifier	**none** " '	You can indicate whether text is distinguished by quotemarks; only doublequotes or singlequotes are supported.
First line contains field names?	**False** True	You can indicate whether the first record of the data consists only of labels for the fields, rather than actual data.

Table 5   Parsing Information for Imported Sample Fixed-Width Data

Item	Value	Notes
Default SQL Type	Any of the following: bigint, bit, char, date, decimal, double, longvarchar, numeric, real, smallint, time, timestamp, tinyint, **varchar**	
Record Length	**0**	To override the default, type in a nonzero number.
HeaderBytesOffset	**0**	To override the default, type in a nonzero number.
Field Count	**0**	To override the default, type in a nonzero number.

6   In the next step, verify the record layout and field properties in the Field information pane, enter the number of sample records to read and display in the preview pane, and then click **Preview**. See Figure 10, or **Figure 26 on page 48**.

Figure 10 Show Sample: Preview of Field Layout
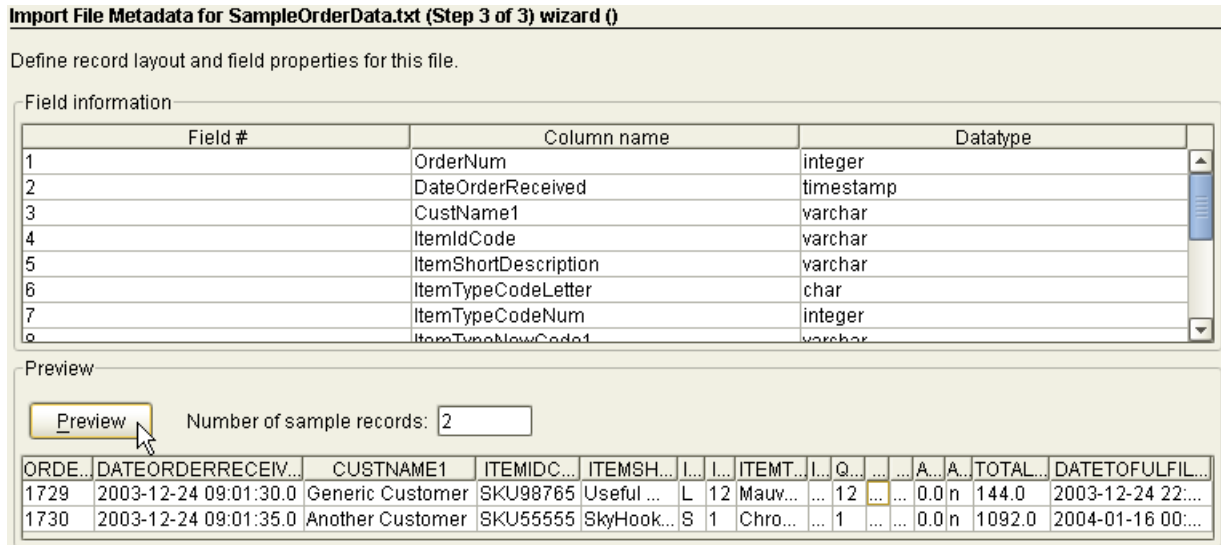
If the sample data is valid (that is, in accord with the layout, encoding, formatting, and properties you specified), the Preview pane displays a table whose rows are the first few records, with each column headed by the data element name.

7 As needed, you can click **Back** and **Next** to return to a step and make changes to layout or formatting, or specify a different input file. When satisfied, click **Finish**.

8 The entire data set is parsed and displayed in the Sample Data pane. For examples, see **Figure 27 on page 49** and Figure 11 below.

Figure 11 Sample Data Displayed in eBAM Editor
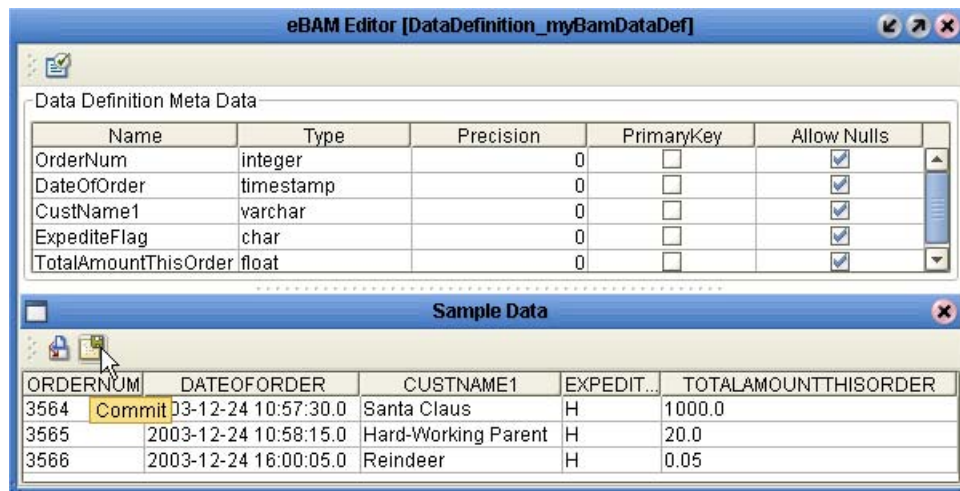
9 Optionally, if you want to commit this data set to being stored, click **Commit**.

After the data definition has been created, populated, and validated, it is available for use in one or more alert conditions and charts.

4.2 Key Performance Indicators: Alerts and Charts

eBAM "dashboard" of key performance indicators (KPIs) consists of zero or more alert conditions and zero or more charts. Each alert condition or chart requires:

- A name and an associated data definition.
  - The KPI's name and data definition are set when you run the wizard that creates the alert condition or chart.

- One or more conditions set on fields in the data definition.
  - For step-by-step instructions on setting up conditions, see **"Setting Data Definition Conditions for an Alert or Chart" on page 25** below, and **"Using the Condition Editor" on page 26**.

- One or more fields in the dataset view, with appropriate mappings to them from operators and data definition fields.
  - For instructions on setting up dataset views and mappings for alert conditions, see the **procedure on page 29**; for charts, see the **procedure on page 37**.

- Property settings for the KPI as a whole, such as resend frequency.
  - For instructions on configuring properties for an alert condition, see the **procedure on page 31**; for charts, see the **procedure on page 37**.

4.2.1 Setting Data Definition Conditions for an Alert or Chart

Common to both alerts and charts is the concept of conditions that are set on the data definition instance. For both charts and alerts, conditions are set in the properties sheet of the data definition instance; the **Condition** property is set using the Condition editor.

To configure a data definition used in an alert condition or chart

1 In the Alert Condition Editor or Chart Editor (discussed in detail later), right-click DatasetView and, on the menu, click **Properties**.

The **Properties** dialog opens, with the **Basic** tab displayed. See Figure 12.

Figure 12 Basic Properties of a Data Definition
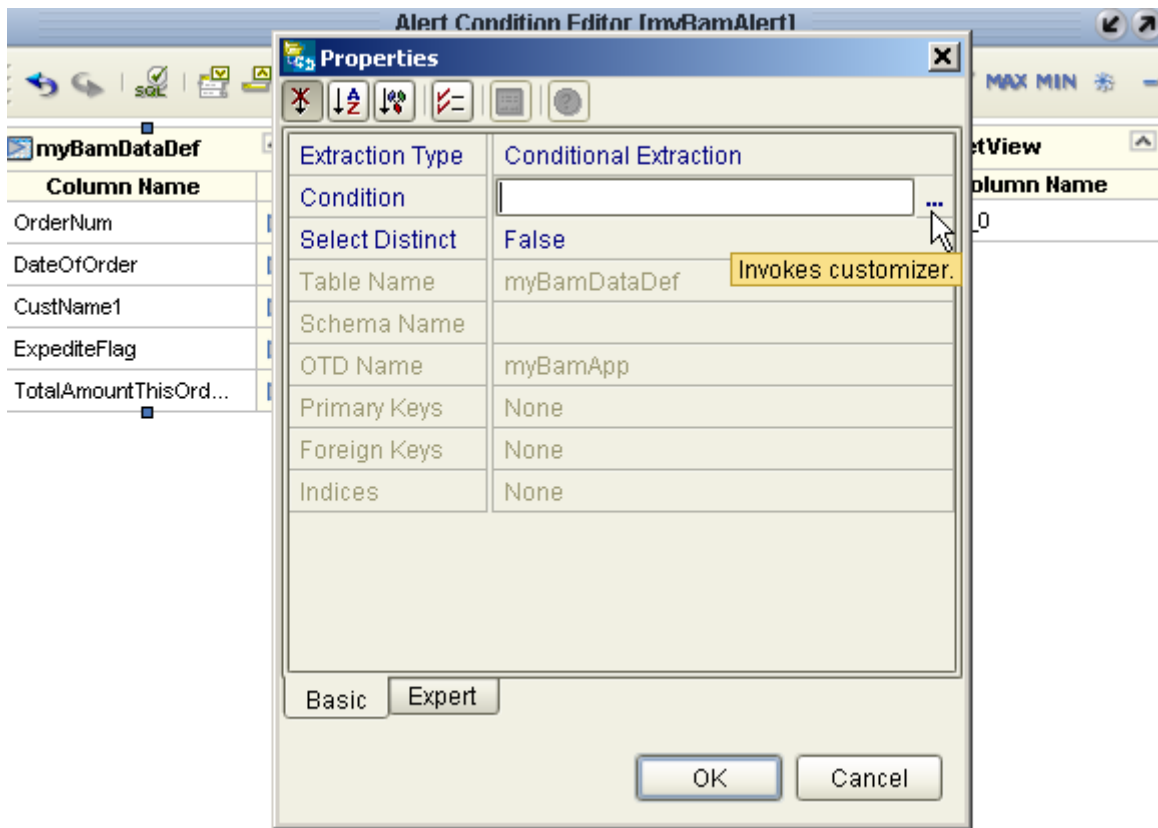
2   In the Properties dialog, click **Condition**, and then click the ellipsis [**...**] to open the **Condition** editor.

Using the Condition Editor

The **Condition** editor is used to construct filters that constitute conditions on the data definition. Although you can type in (or use CTRL+V to paste in) native SQL in text form, the editor offers features that help SQL-adept users avoid making mistakes, while allowing other users with little or no SQL knowledge to construct statements by purely graphical means.

In the Condition editor, you can switch at will both between the two tree listings (Columns and Operators) and between the two user modes (SQL Code and Graphical).

▪ The **Columns** tree, which opens via double-click, lists all data elements in the data definition. In both SQL Code and Graphical mode, you drag elements from this tree onto the canvas, after which you will specify operations to be performed on them.

▪ The **Operators** tree lists all SQL operations, providing a quick language reference for SQL Code mode (although you can also drag-and-drop) and providing tools to be dragged onto the Graphical canvas. For any operator placed on the canvas, you can hover your cursor over the title or any field for help on the operator or field.

▪ The **SQL Code** canvas allows you to enter native SQL commands and/or to drag elements or operators from the trees on the left. Its tool palette provides two tools:

Table 6   Tools Provided in the SQL Code Canvas of the Condition Editor

	Synchronize SQL Code with Graph generates the SQL code listing from the current graphical representation. (When the two modes are out of synch, the background is of the last-changed mode is gray rather than white.)
	Validate checks the syntactical validity of the SQL code listing and reports success or any errors found.

▪ The **Graphical** canvas allows you to create conditions simply by dragging elements and operators from the trees on the left. Its tool palette provides several tools:

Table 7   Tools Provided in the Graphical Canvas of the Condition Editor

	Undo and **Redo** allow you travel backward and forward through the sequence of modifications you have made to the graphical canvas.
	Synchronize Graph with SQL Code generates the graphical representation corresponding to the current SQL code listing. (When the two modes are out of synch, the background is of the last-changed mode is gray rather than white.)
	Validate checks the syntactical validity of the graph and, in the Output pane, reports success or any errors found, highlighting all operators that are unsatisfied.
	Show SQL for Condition tries to validate the graph and, if successful, displays the corresponding SQL statement for the entire condition. In the Output pane, you can specify which "flavor" of SQL to display, of: Oracle8, Oracle9, SQL Server, internal, or ANSI92.
	Expand All Graph Icons provides more detail, by showing all input to (and output from) all fields in all operators on the canvas.
	Collapse All Graph Icons provides more screen space by minimizing all operators, showing only the connections to and from them.
	Autolayout All Graph Icons disentangles crossed connections and overlaps, and creates left-to-right flow of input to output.
	Print Graph allows you to print the graph, using any of various scaling options.

When the chart editor or alert editor displays a data definition that has a condition, a small "filter" icon— —appears to the right of the data element(s) being watched.

4.2.2 Setting Up the Application's Alert Conditions

eBAM allows you to set up tests whereby an alert (notification) is triggered whenever a condition is met by one of the data items, such as exceeding a threshold. As discussed earlier, you use the Condition editor to create queries that detect the condition. In addition to conditions on the data definition, you also set up a dataset view.

An eBAM application can contain many alert conditions, or only one, or none at all. Each alert condition requires the following:

- A name and an associated data definition; see **"To create a new alert condition"**.

- One or more conditions set on fields in the data definition; see **"To configure a condition on a data definition field" on page 29**.

- One or more fields in the dataset view, with appropriate mappings to them from operators and data definition fields; see **"To configure the dataset view, mapping operators and data definition fields to its fields" on page 29**.

- Settings for the alert condition as a whole, such as resend frequency; see **"To configure properties of the entire alert condition" on page 31**.

To create a new alert condition

1 In the project tree, right-click the eBAM application and, on the popup context menu, click: **New Alert Condition**

2 In step 1 of the wizard, type in a name for the alert condition, and then click **Next**.

3 In step 2, select the check box for the data definition, and then click **Finish**.

   The project tree displays the new object under Alert Conditions, and the Alert Condition Editor opens to display the data definition on the left side of the canvas and an empty dataset view on the right side.

4 To see the entire data definition instead of a scrollable view, right-click it and, on the menu, click **Fit To Size**. See Figure 13.

Figure 13   Newly Created Alert Condition
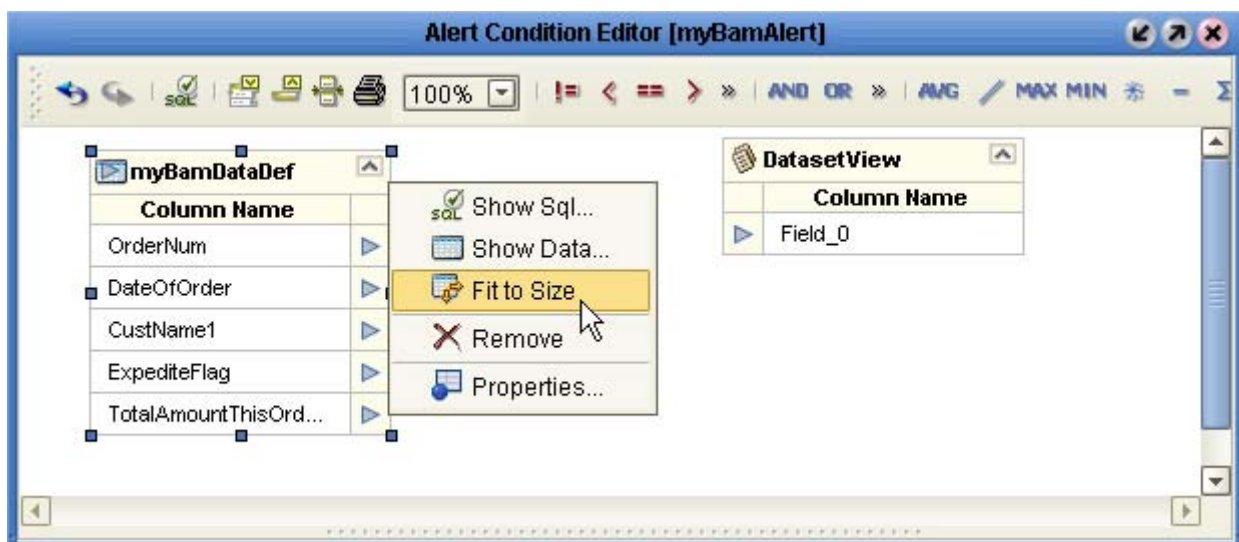
To configure a condition on a data definition field

1 In the Alert Condition Editor, right-click the data definition; on the popup menu, click **Properties**. In the Properties dialog, click **Condition**, and then click the ellipsis [**...**] to open the Condition editor.

2 In the **Columns** tab, double-click the data definition open it and display its fields.

3 See **"Setting Data Definition Conditions for an Alert or Chart" on page 25**.

To configure the dataset view, mapping operators and data definition fields to its fields

1 In the Alert Condition Editor, right-click **DatasetView** and, on the menu, click: **Configure**

2 In the **Configure Dataset View** dialog, click **Add** as many times as needed to accommodate all the report fields you will need.

3 Double-click each field, edit the name, and press ENTER. When finished, click OK.

4 In the Alert Condition Editor, drag operators as needed from the Operator Palette (see **Figure 14 on page 30**) into the canvas between the data definition and the dataset view, and then map zero or more fields from the data definition through zero or more operators, with all output ultimately going into the dataset view. For examples of this, see the background of **Figure 15 on page 31**, or see **Figure 31 on page 53**.

Note: *Data type conversions are done automatically where needed; for timestamps converted to varchar, permit truncation to occur.*

5 Optionally, from time to time, you may want to click **Show SQL** to see the SQL statements you are creating graphically, or use other the graphical design tools listed in Table 8.

Table 8   Tools Provided in the Tool Palette of the Alert Condition Editor

	Undo and **Redo** allow you travel backward and forward through the sequence of modifications you have made to the graphical canvas.
	Show SQL tries to validate the graph and, if successful, displays the corresponding SQL statements for the mapping. In the Output pane, you can specify which "flavor" of SQL to display, of: Oracle8, Oracle9, SQL Server, internal, or ANSI92.
	Expand All Graph Icons provides more detail, by showing all input to (and output from) all fields in all operators on the canvas.
	Collapse All Graph Icons provides more screen space by minimizing all operators, showing only the connections to and from them.
	Autolayout All Graph Icons disentangles crossed connections and overlaps, and creates left-to-right flow of input to output.
	Print Graph allows you to print the graph, using any of various scaling options.

Figure 14 Operator Palette for SQL Operations
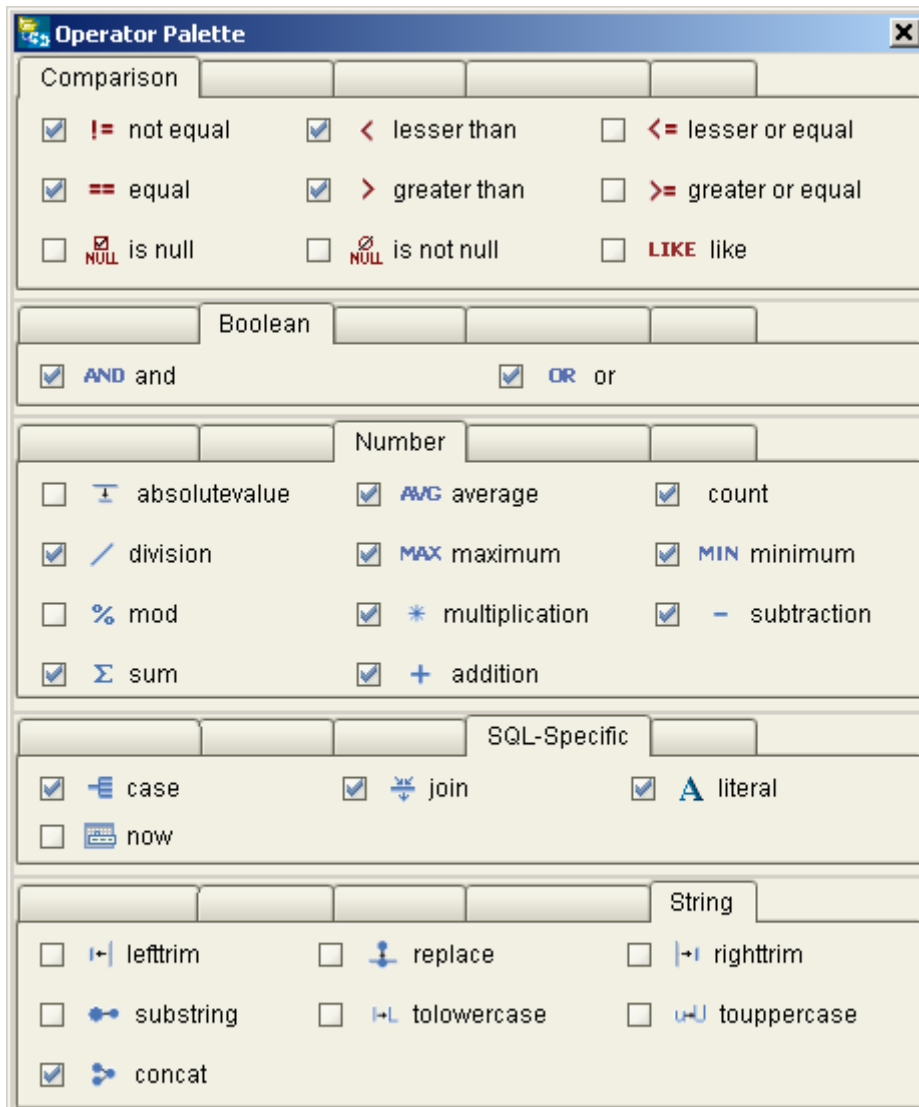
6 When you have set up all dataset view fields, save your work.

You are now ready to configure the properties of the alert condition as a whole.

To configure properties of the entire alert condition

1 In the Alert Condition Editor, right-click a blank portion of the canvas itself and, on the popup menu, click: **Properties** (see below)
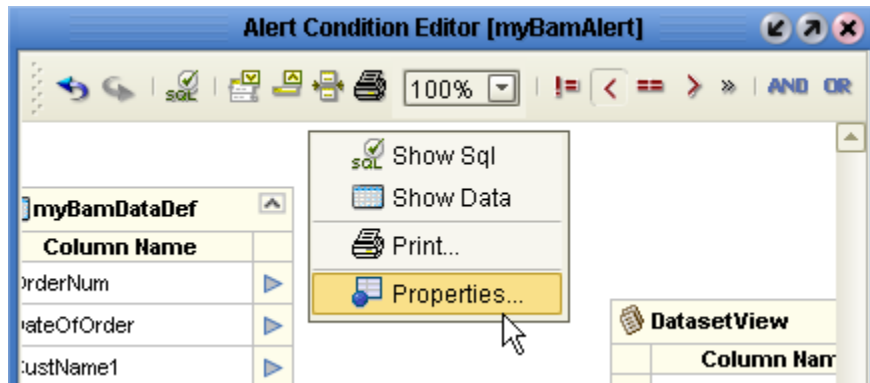
2 In the Alert Condition Properties dialog, change the Notification Interval and Resend Frequency to appropriate values and, from the dataset view fields listed under Keys, select one or more check boxes; then click Close. See Figure 15.

Figure 15 Setting Properties for the Entire Alert Condition
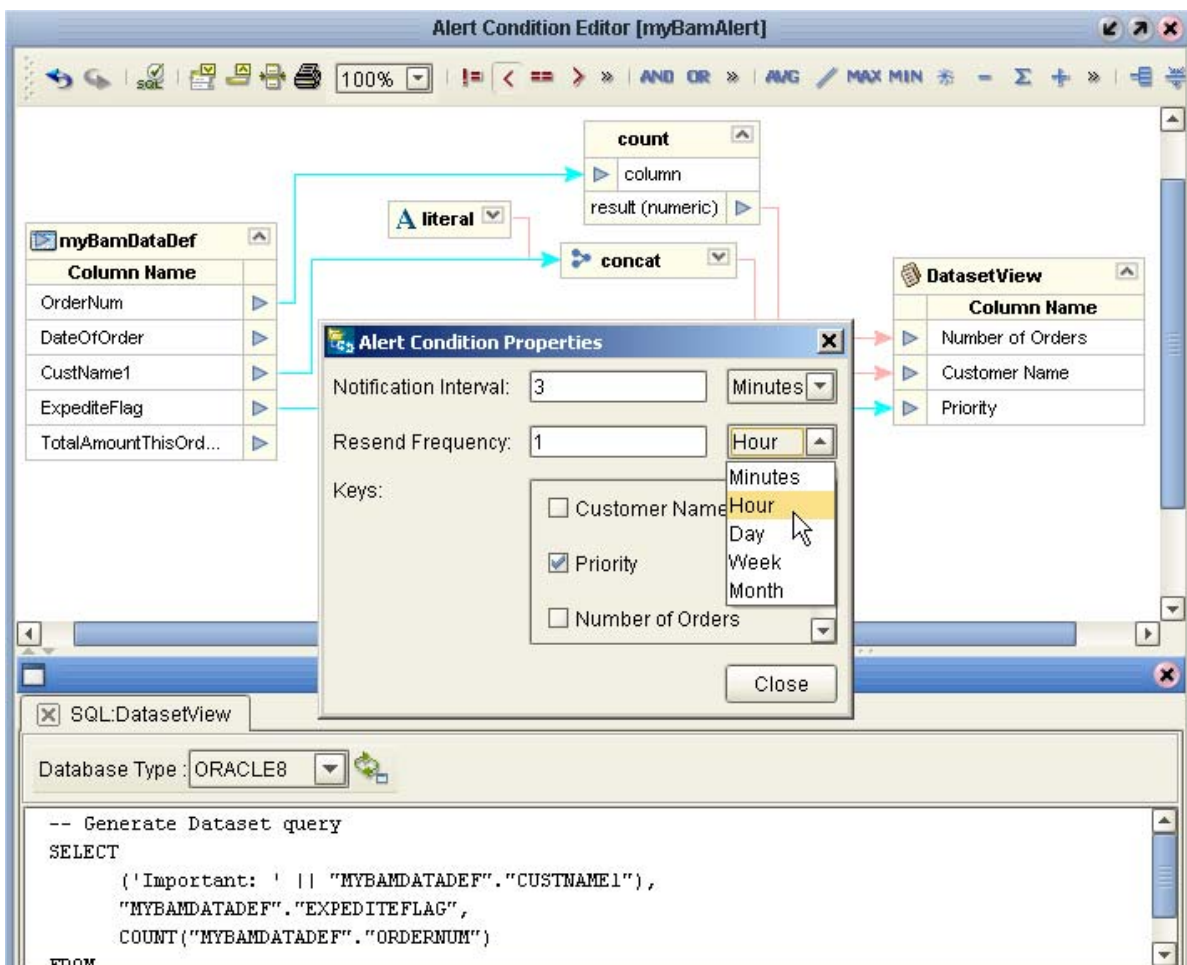
4.2.3 **About Charts**

eBAM allows you to create three types of charts:

- *Pie charts* display elements in the dataset view as wedge-shaped segments (slices) comprising an entire disk (pie). Each segment's relation to the pie and to other segments is cued visually by its apparent area, based on its subtended angle. Specific view fields can be "exploded" to highlight them.

- *Bar charts* display elements in the dataset view as rectangles (bars) in a rectilinear setup where each bar's relation to the total and to other bars is cued visually by its apparent area, based on its length. Bar charts can have labels on either or both axes to facilitate quantitative readings.

- *Meters*, also called *meter charts*, display conditions as a needle on a dial, similar to a tachometer or clock face. At designated thresholds, differing colors can be used to signify when the measured condition is in normal range, warning range, or critical range. For an example, see **Figure 21 on page 39**.

Figure 16 provides an example of how chart types can be combined in ePortal to create a dashboard-like set of key performance indicators (KPIs).

Figure 16 Chart Types Combined in ePortal to Form a "Dashboard" of KPIs
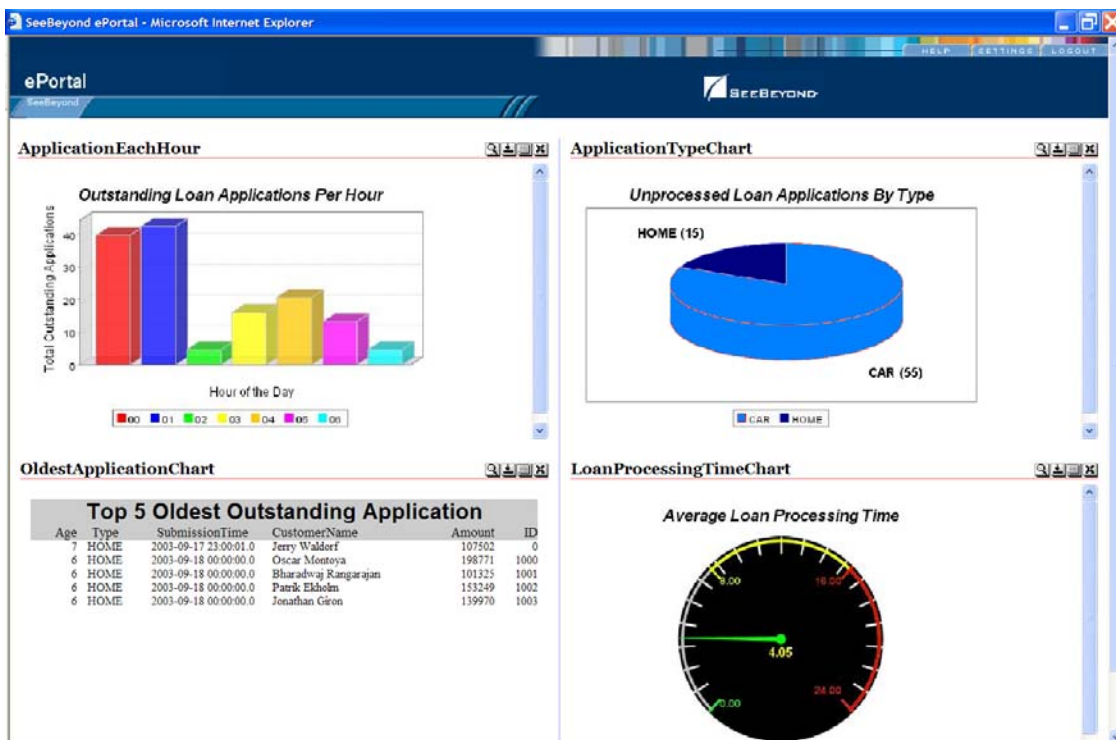
All charts have common properties, and each has properties specific to its own type:

- **Table 9 on page 33** lists properties common to all three chart types.
- **Table 10 on page 33** lists properties common to pie and bar charts only.
- **Table 11 on page 34** lists properties specific to pie charts.
- **Table 12 on page 34** lists properties specific to bar charts.
- **Table 13 on page 34** lists properties specific to meters.

Table 9 Properties Common to All Three Chart Types

Property Name	Value	Notes
Display Title	True	Whether or not the chart title is displayed.
Title		Any string; initially set to match the name provided when the chart was created
Title Font		You can use any of 52 fonts, ranging in size from 9-point to 72-point, with or without bold and/or italic attributes.
Title Alignment	CENTER	You can change the default (CENTER) to either LEFT or RIGHT.
Title Color		You can specify any color using any of these methods: ▪ Pick a swatch from the display.
Title Background		▪ Specify percentages for HSB: Hue, Saturation, Brightness ▪ Specify values (0-255) for RGB: Red, Green, Blue.
Draw Border	False	Whether or not to draw a border around the chart.
Border Color		You can specify any color either by picking a swatch or by specifying HSB or RGB.
Background Color		
Image Width	680	
Image Height	420	
Chart Width	680	
Chart Height	420	
Frequency in seconds	60	How often to update the chart with a new data view.

Table 10 Properties Common to Pie and Bar Charts (Continued)

Property Name	Value	Notes
3D	True	Whether or not a three-dimensional effect is displayed.
Depth Factor	0.3	Amount by which the chart seems three-dimensional.
Include Legend	False	Whether or not to display a legend with the chart.
Legend Anchor	SOUTH	Where to position the starting point for the chart legend.
Circular	False	
Null Real	0.0	Value to use for real numbers (float) without data.
Null Integer	0	Value to use for integers without data.
Null String		Value to use for strings (varchar) without data.
Integer Number Format	0	How many digits to display, and whether to display digits in comma-separated groups of three.
Real Number Format	0.000	Whether to display digits in comma-separated groups of three, and whether to display three digits after the decimal point (default) or two, with a dollar sign preceding the value.

Table 11 Properties Specific to Pie Charts

Property Name	Value	Notes
Radius	0.7	
Section Label	Name,Value	Choices consist of: None; Name; Value; Percent; Name,Value; Name,Percent, and Value,Percent.
Section Label Font		Any of 52 fonts, 9-pt to 72-pt, bold and/or italic or not.
Section Label Color		You can specify any color either by picking a swatch or by specifying HSB or RGB.
Section Label Gap	0.3	
Show Series Labels	False	Whether to display a label for the series.
Series Label Font		Any of 52 fonts, 9-pt to 72-pt, bold and/or italic or not.
Series Label Color		You can specify any color by swatch, HSB, or RGB.
Direction	Clockwise	Each successively larger value is displayed either right (clockwise) or left (anticlockwise) of the next smaller.
Interior Gap		

Table 12 Properties Specific to Bar Charts

Property Name	Value	Notes
Orientation	VERTICAL	Which direction (vertical or horizontal) the bars run.
Show X-Axis	True	Whether or not to display the X axis.
Show X-Axis Label	True	Whether or not to display the label for the X axis.
X-Axis Label	domain	The text to display as a label for the X axis.
Show Y-Axis	True	Whether or not to display the Y axis.
Show Y-Axis Label	True	Whether or not to display the label for the Y axis.
Y-Axis Label	range	The text to display as a label for the Y axis.

Table 13 Properties Specific to Meters

Property Name	Value	Notes
Minimum Value	0.0	
Maximum Value	0.0	
Value Font		Any of 52 fonts, 9-pt to 72-pt, bold and/or italic or not.
valuePaint		
Minimum Normal Value	0.0	Values below this threshold are too low to be normal.
Maximum Normal Value	0.0	Values above this threshold are too high to be normal.
Normal Range Color		You can specify any color either by picking a swatch or by specifying HSB or RGB.
Minimum Warning Value	0.0	Values below this threshold are too low for warnings.
Maximum Warning Value	0.0	Values above this threshold are too high for warnings.

Table 13 Properties Specific to Meters (Continued)

Property Name	Value	Notes
Warning Range Color		You can specify any color by swatch, HSB, or RGB.
Minimum Critical Value	0.0	Values below this threshold are too low to be critical.
Maximum Critical Value	0.0	Values above this threshold are beyond being critical.
Critical Range Color		You can specify any color by swatch, HSB, or RGB.
Units	units	
Draw Chart Border	False	Whether or not to display a border around the chart.
Border Type	Normal Range	You can change this Warning, Critical, or Full Range.
Dial Type	Circle	You can change this to either Pie or Chord.
Dial Border Color		You can specify any color by swatch, HSB, or RGB.
Dial Background Color		You can specify any color by swatch, HSB, or RGB.
Tick Label Type	Value Label	Whether to show marks with values, or just marks.
Tick Label Font		Any of 52 fonts, 9-pt to 72-pt, bold and/or italic or not.
Needle Color		You can specify any color, by swatch, HSB, or RGB.
Meter Angle	270	Angle subtended by the entire range of the meter.

4.2.4 Setting Up the Application's Charts

In eBAM, charts provide real-time feedback on the current data set according to conditions you set on one or more of the elements in the data definition. As discussed earlier (see **"Using the Condition Editor" on page 26**), you use the Condition editor to create queries that detect the condition. In addition to conditions on the data definition, you also set up a dataset view.

An eBAM application can contain many charts, or only one, or none at all. Each chart requires the following:

- A name and an associated data definition; see **"To create a new chart"**.

- One or more conditions set on fields in the data definition; see **"To configure a condition on a data definition field" on page 36**.

- One or more fields in the dataset view, with appropriate mappings to them from operators and data definition fields; see **"To configure the dataset view, mapping operators and data definition fields to its fields" on page 37**.

- Settings for the chart's properties as a whole. In addition to generic chart properties discussed earlier (see **"About Charts" on page 32**), you can also set chart properties that depend on the dataset view fields of your particular application, such as group-by (aggregation) fields and order-by settings; see **"To configure properties of the entire chart" on page 37**.

To create a new chart

1 In the project tree, right-click myBamLoan and, on the popup context menu, click:
New Chart

2 In step 1 of the Chart wizard, specify a chart type (Pie Chart, Bar Chart, or Meter) and a name.

3 In step 2 of the wizard, select the check box for the data definition.

4 In step 3 of the wizard, adjust the common and chart-specific parameters in all three tabs to suit your taste. (Alternatively, you can defer your chart parameter decisions to a later time, and simply accept all defaults for now).

5 Click **Finish**.

Result: The project tree displays the new object under Charts, and the Chart Editor opens to display the data definition on the left side of the canvas and an empty dataset view on the right side. See Figure 17.

Figure 17 Chart Editor
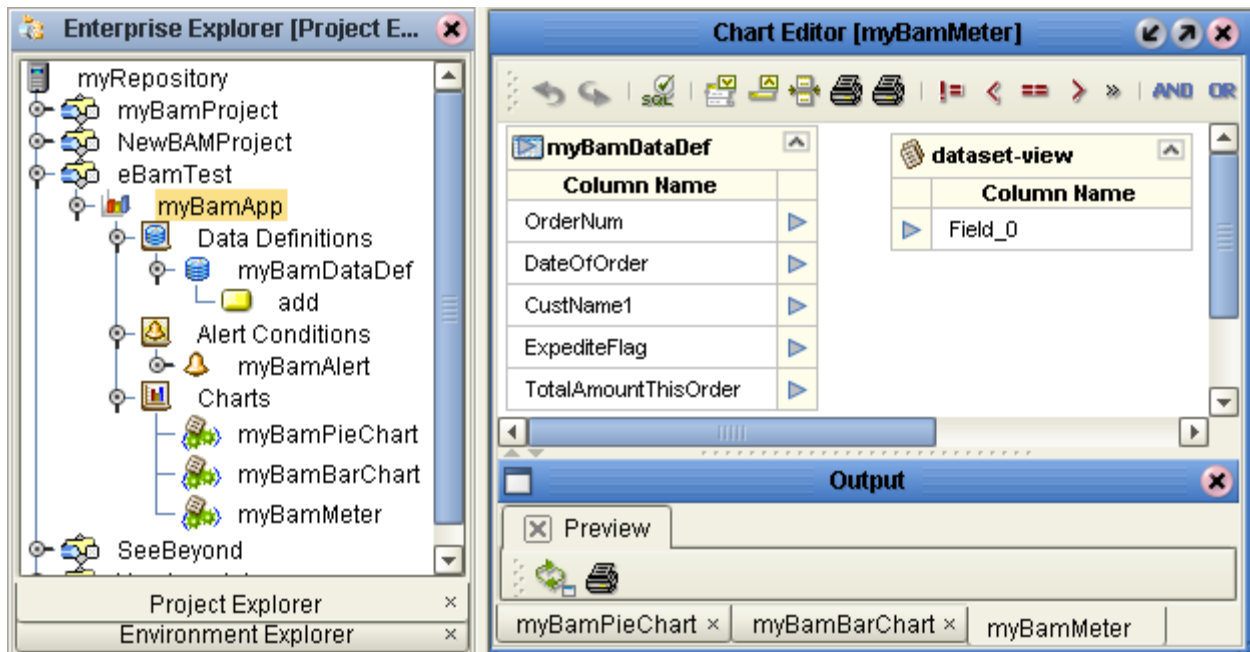
Tip: The tool palettes for the Chart Editor and its output window are the same as for the Alert Conditions Editor. **Table 8 on page 29** explains the graphical controls, and **Figure 14 on page 30** lists the SQL operators.

To configure a condition on a data definition field

1 In the Chart Editor, right-click the data definition; on the popup menu, click **Properties**. In the Properties dialog, click **Condition**, and then click the ellipsis [...] to open the Condition editor.

2 In the **Columns** tab, double-click the data definition open it and display its fields.

3 See **"Setting Data Definition Conditions for an Alert or Chart" on page 25**.

To configure the dataset view, mapping operators and data definition fields to its fields

1 In the Chart Editor, right-click **dataset-view** and, on the menu, click: **Configure**

2 In the **Configure Dataset View** dialog, click **Add** as many times as needed to accommodate all the report fields you will need.

3 Double-click each field, edit the name, and press ENTER. When finished, click OK.

4 In the Alert Condition Editor, drag operators as needed from the Operator Palette (see **Figure 14 on page 30**) into the canvas between the data definition and the dataset view, and then map zero or more fields from the data definition through zero or more operators, with all output ultimately going into the dataset view. For examples of this, see **Figure 31 on page 53** and Figure 18 below.

Figure 18 Configuring a Dataset View Using Operators and Data Definition Fields
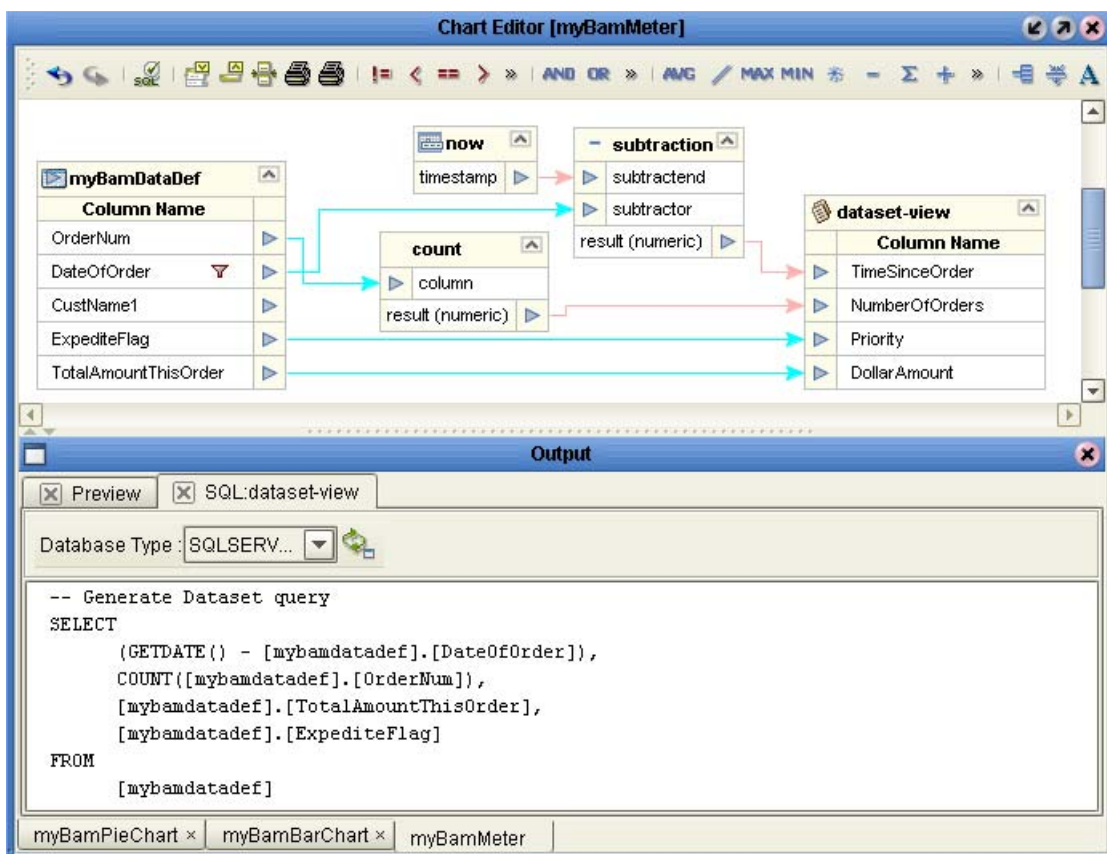
To configure properties of the entire chart

1 In the Chart Editor, right-click the blank canvas and, on the menu, click: **Properties**

2 In the tree on the left side, select the **GroupBy** folder. Under Available Group-By Columns, select the dataset view element you want to group by, and then click **Add** to move the element to the Selected Group-By Columns pane. See Figure 19.

Figure 19   Chart Properties: "Available Group-By" and "Selected Group-By" Columns
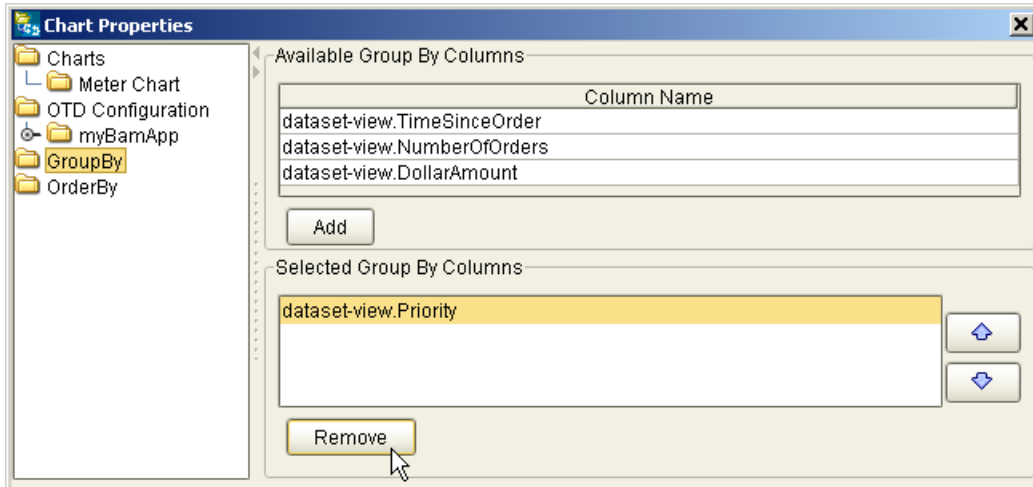
3   Optionally, select the **OrderBy** folder, use the Sort Order controls to set the order (ascending or descending) for each view element, select the view element you want to order by, and then click **Add** to move the element to the Selected Order-By Columns pane. See Figure Figure 20.

Figure 20   Chart Properties: "Available Order-By" and "Selected Order-By" Columns
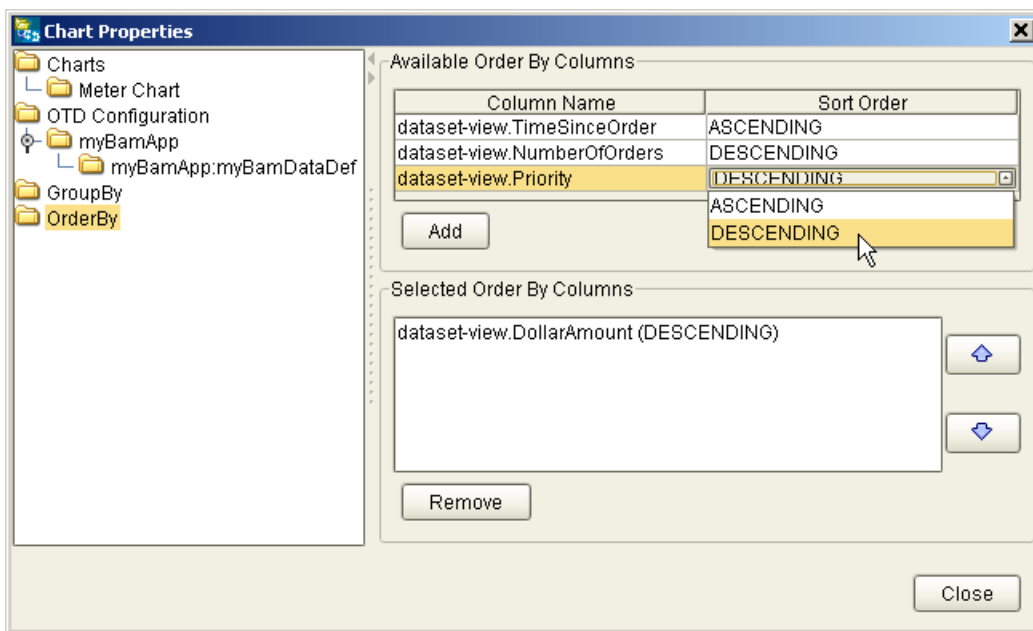
4   Optionally select the **Charts** folder and modify any of the common properties or chart-specific properties. (This is the alternative mentioned in step 4 of the **procedure on page 36**.)

5   When you have configured all chart properties, click **Close**.

6   To preview the chart, right-click the blank canvas and, on the menu, click: **Preview**

The Output window shows a preview of how the chart will appear. For an example, see Figure 21.

Figure 21 Previewing a Chart
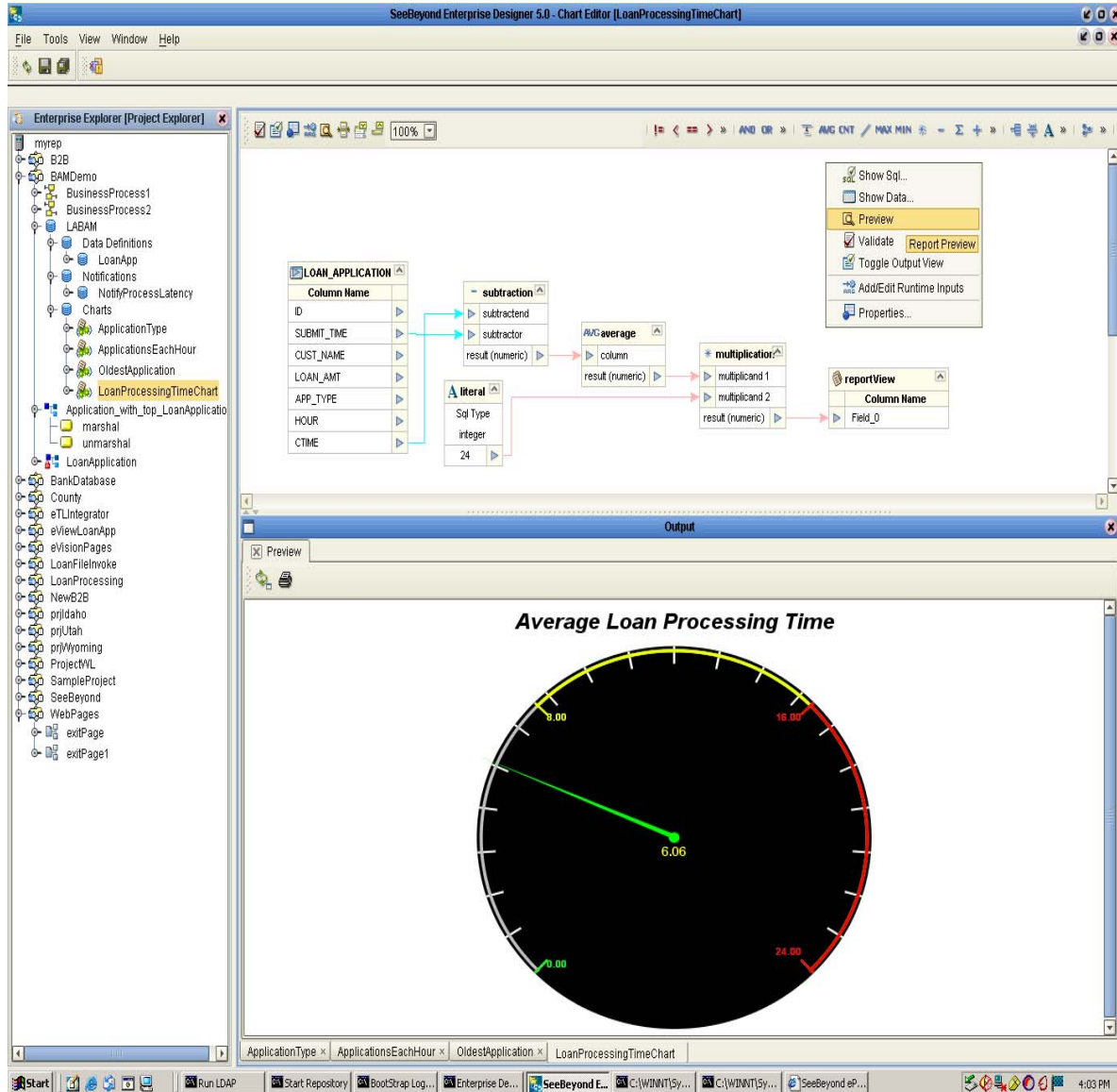
7   In the **Output** window, with the Preview tab selected, you can:

 • Use the ⟳ **Refresh** button to update the preview any time you make a change to the chart properties.

 • Use the 🖶 **Print** button to print a hardcopy of the previewed chart.

Implementing eBAM

This chapter provides step-by-step instructions for using the sample project and files supplied with the eBAM product file.

The instructions in this chapter assume you have installed the prerequisite products (either eInsightESB, or else both eGate and eInsight), and that you have also installed the File eWay, which the sample implementation uses for input/output of sample data.

5.1 Last Things First

When you finish this chapter, you will have designed, deployed, run, and monitored an eBAM application that matches the sample supplied in eBAMDocs.sar. When this application is fed the "typical" data file, the result will be a pie chart that looks like Figure 22, with the display updating every 30 seconds.

Figure 22   Pie Chart for Sample
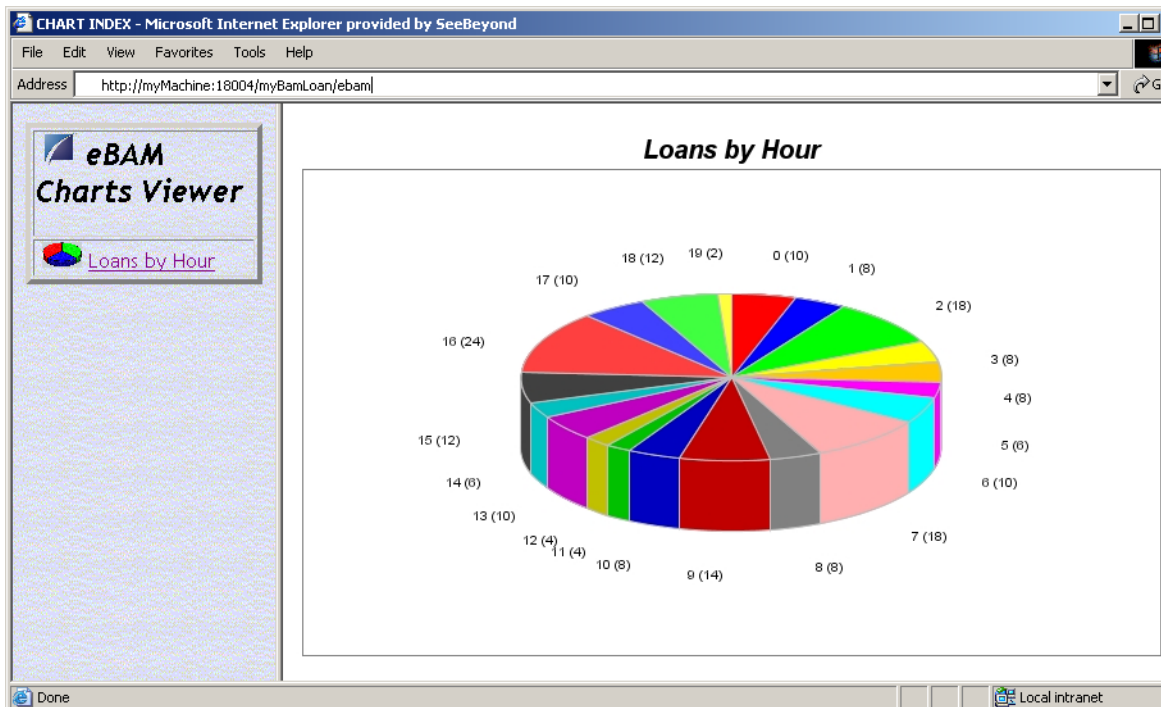
- To replicate the sample on your own system (recommended), continue with **"Overview of Steps for Setting Up the Sample Implementation" on page 41**.

- To start from scratch and create your own implementation, skip ahead to **"Your eBAM Implementation" on page 43**.

5.1.1 Overview of Steps for Setting Up the Sample Implementation

The eBAM Studio product includes a complete sample implementation, included in the eBAMDocs.sar file, that allows you to see the end results without having to go through design-time steps. This section provides steps on installing and setting up the sample:

- **Installing the Sample Files** on page 41
- **Creating the Sample Environment** on page 42
- **Starting the Logical Host for the Sample** on page 42
- **Importing the Sample Project** on page 43

The steps for creating a deployment profile, activating it, and monitoring the sample results are the same as for the homegrown implementation:

- **Creating and Configuring the Connectivity Map** on page 58
- **Activating and Running the Project** on page 61
- **Monitoring the Key Performance Indicators with Live Data** on page 62

5.1.2 Installing the Sample Files

These steps assume the existence of a temporary eBAM directory for sample files, such as **C:\temp\eBAM**. You will install the sample files to this directory.

To install the sample files

Before you begin: Your repository must already be running, and you must be logged in to Enterprise Manager. If you have already uploaded the documentation for eBAM, you can skip steps 1 and 2 and start with step 3.

1 In the ADMIN tab, if you have not already done so, browse to the [...]\Documentation**ProductsManifest.xml** file and submit it.

2 In the ADMIN tab, if you have not previously done so, browse to the **eBAMDocs.sar** file, select it, and click the **upload now** button.

3 In the DOCUMENTATION tab, in the Products window, click **eBAM Studio**.

4 In the window that appears on the right side, click **Download Sample**.

5 Preserving file paths, extract the files to your temporary eBAM samples directory.

Result: The following directories and files are created:

```
C:\temp\eBAM\Sample\Projects\SampleBamProject.zip
C:\temp\eBAM\Sample\Data\In\LoanDataAllSmall.txt
C:\temp\eBAM\Sample\Data\In\LoanDataOneBig.txt
C:\temp\eBAM\Sample\Data\Out\
```

5.1.3 Creating the Sample Environment

These steps assume you will use a default application server running on default ports. If you use anything other than a SeeBeyond Integration Server on ports 18000–18009, make adjustments in step 6 below or in the URL in step 2 in the **procedure on page 62**.

To create the sample environment

1 In Enterprise Designer, near the lower left of the window, click the **Environment Explorer** tab.

2 In the environment tree, right-click the repository and, on the popup context menu, click **New Environment**

   ◆ Rename the newly created environment to **SampleBamEnv**

3 Right-click SampleBamEnv and, on the menu, click: **New File External System**

   ◆ Name the new external **SampleExternalFileIn**, designate it an Inbound File eWay, and click OK.

4 Right-click SampleBamEnv > **New File External System**

   ◆ Name it **SampleExternalFileOut**, designate it Outbound, and click OK.

5 Right-click SampleBamEnv > **New Logical Host**

   ◆ Retain the default name: LogicalHost1

6 Right-click LogicalHost1 and click: **New SeeBeyond Integration Server**

   ◆ Retain the default name: IntegrationSvr1

Result: The sample environment now contains the three servers it needs.

5.1.4 Starting the Logical Host for the Sample

These steps assume you have already installed a logical host named LogicalHost1, and that the environment you have created or imported is named SampleBamEnv.

To bootstrap the logical host

1 Open a command prompt and change directories to the location of your logical host's bootstrap executables. For example:

   ```
   cd \ican5\logicalhost\bootstrap\bin
   ```

2 Start the bootstrap script using appropriate parameters. For example:

   ```
   bootstrap -r http://myBox:12345/myRep -i myId -p myPassword
   -e SampleBamEnv -l LogicalHost1
   ```

   ▪ For the **-r** (repository) parameter), supply the correct URL with repository name.
   ▪ For the **-i** and **-p** (ID and password) parameters, supply the appropriate values.
   ▪ For **-e** (environment) parameter, use: **SampleBamEnv**
   ▪ For **-l** (logical host name) parameters, use: **LogicalHost1**

Result: The logical host is now running, and ready to have a project deployed to it.

5.1.5 # Importing the Sample Project

To install the sample Project

Before you begin: Your repository must already be running, and you must be logged in to Enterprise Designer.

1   In Enterprise Designer, in Project Explorer, right-click the repository and, on the popup context menu, click: **Import**

2   In the **Import Manager** dialog, open the folder where you installed the sample files (such as `C:\temp\eBAM\Sample\Projects`), select `SampleBamProject.zip`, and click **Open**.

3   In the **File Destination** dialog, select **Import to a new project**, enter the name **SampleBamProject**, and click OK.

4   On the main toolbar, click: ⟳ **Refresh All from Repository**

5   For the remaining steps, skip ahead to **"To configure the eWays" on page 60**. Upon finishing it, continue with **"Activating and Running the Project" on page 61** and then **"Monitoring the Key Performance Indicators with Live Data" on page 62**.

5.2 # Your eBAM Implementation

This section provides detailed step-by-step instructions to replicate the design of the sample project. After you complete the design work, you will need to go through the usual steps for creating an environment, activating the project, and running it on a bootstrapped logical host. When the project is running, further steps show you how to monitor the results of feeding different kinds of data.

Preliminary steps

▪ **Setting Up the Project and Environment** on page 44

▪ **Starting the Logical Host** on page 45

Design-time steps

▪ **Creating and Validating the OTD** on page 45

▪ **Creating and Validating the Data Definition** on page 47

▪ **Creating and Configuring the Alert Condition** on page 49

▪ **Creating and Configuring Charts** on page 52

▪ **Creating and Configuring Business Processes** on page 54

▪ **Creating and Configuring the Connectivity Map** on page 58

Activation, runtime, and troubleshooting steps

▪ **Activating and Running the Project** on page 61

▪ **Monitoring the Key Performance Indicators with Live Data** on page 62

5.2.1 Setting Up the Project and Environment

To create the project

1 In Enterprise Designer, in Project Explorer, right-click the repository and, on the popup context menu, click: **New Project**

2 Change the name of the new created project to: **myBamProject**

To create the environment

These steps assume you will use a default application server running on default ports. If you use anything other than a SeeBeyond Integration Server, you will need to make the appropriate adjustment to step 6.

1 In Enterprise Designer, near the lower left of the window, click the **Environment Explorer** tab.

2 In the environment tree, right-click the repository and, on the popup context menu, click **New Environment**. Rename the newly created environment to: **myBamEnv**

3 Right-click SampleBamEnv and, on the menu, click: **New File External System**. Name the new external **myExternalFileIn**, designate it an Inbound File eWay, and click OK.

4 Right-click SampleBamEnv > **New File External System**; name the new external **myExternalFileOut**, designate it an Outbound File eWay, and click OK.

5 Right-click SampleBamEnv > **New Logical Host** and retain the default name: LogicalHost1

6 Right-click LogicalHost1 > **New SeeBeyond Integration Server** and retain the default name: IntegrationSvr1

Result: The environment has all servers needed for the sample. See Figure 23.

Figure 23   Environment for Sample Implementation: myBamEnv
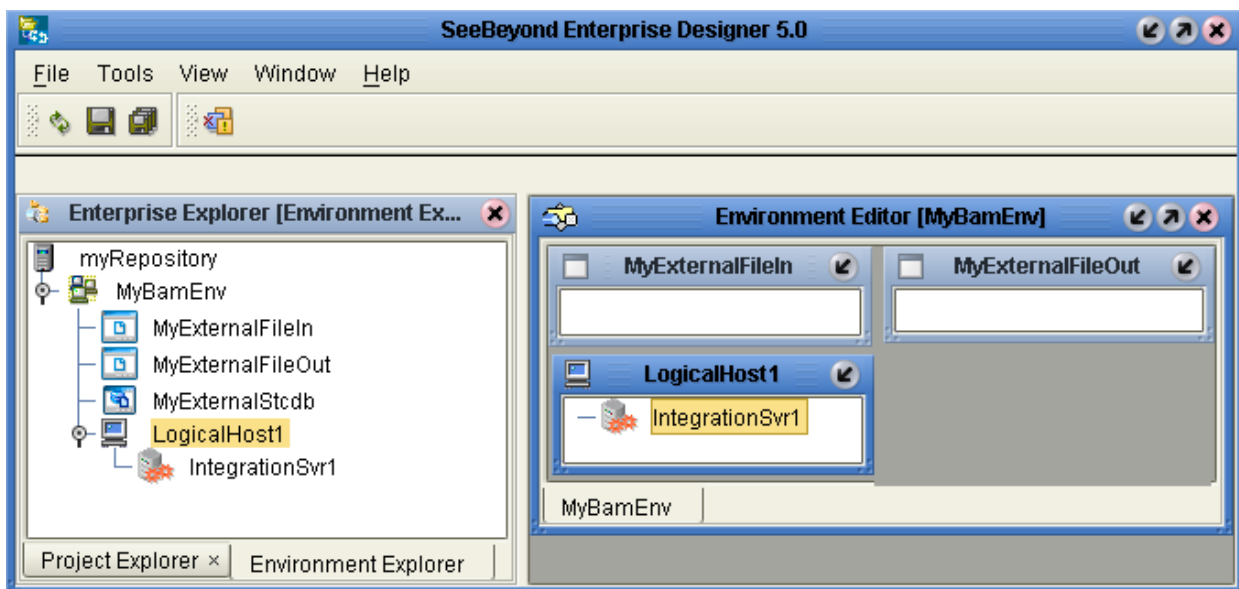
5.2.2 **Starting the Logical Host**

These steps assume you have already installed a logical host.

To bootstrap the logical host

1 Open a command prompt and change directories to the location of your logical host's bootstrap executables. For example:

```
cd \ican5\logicalhost\bootstrap\bin
```

2 Start the bootstrap script using appropriate parameters. For example:

```
bootstrap -r http://myBox:12345/myRepository -i myId -p myPassword
-e myBamEnv -l LogicalHost1
```

  ▪ For the **-r** (repository) parameter), supply the correct URL with repository name.

  ▪ For the **-i** and **-p** (ID and password) parameters, supply the appropriate values.

  ▪ For **-e** (environment) parameter, use: **myBamEnv**

  ▪ For **-l** (logical host name) parameters, use: **LogicalHost1**

Result: The logical host is now running, and ready to have a project deployed to it.

5.2.3 **Creating and Validating the OTD**

This particular implementation requires an User-Defined OTD to unmarshal data from a file that uses commas to delimit its records' six fields. Nothing in the following steps is specific to eBAM; however, the OTD is tightly coupled to the sample data provided, so take care that in the resulting OTD, the six fields appear in the exact order specified (CustName, Hour, ID, LoanAmt, SubmitTime, TypeOfApp).

To create myBamOtd

1 Right-click myBamProject and, on the popup context menu, point at **New** and click: **Object Type Definition**

2 In the wizard, select **User-Defined OTD** and then click Next.

3 Name the OTD **myBamOtd** and then click Finish.

Result: The OTD is created, and is displayed in the OTD Editor.

To configure myBamOTD as comma-delimited

1 In the OTD Editor, in the Properties pane, double-click the properties column for **delim**, and then click the ellipsis [**...**] to open the Delimiter List Editor.

2 Right-click within the empty delimiter list and, on the popup context menu, click: **Add Level To End**

3 Right-click the newly created **Level 1** and, on the menu, click: **Add Delimiter**

4 In the newly created **Delimiter** row, double-click the **Delimiter Bytes** column.

5 Type , (in other words, press the COMMA key), press ENTER, and then click OK.

To add the six fields to myBamOTD

- In the OTD Editor, in the Object Type Definition pane, repeat the following two steps until you have create six fields, thus:

  A Right-click myBamOTD; in the popup context menu, point at Add Field, and then click: **Add Field As Child Node To End**

  B In the Input dialog, name the new fields, in order, from first to last: **CustName**; **Hour**; **ID**; **LoanAmt**; **SubmitTime**; **TypeOfApp**

To validate myBamOTD with sample data

1 In the OTD Editor, on the toolbar palette, click: ☑ **Run Test**

   The OTD Tester pane appears below the main OTD Editor.

2 In the OTD Tester, with the **Input** tab selected, type or paste the following string:

   `GIRON LLC,6,10032,32923.66,2003-09-01 12:59,CAR`

   Since the field order is {CustNum, Hour, ID, LoanAmt, SubmitTime, TypeOfApp}, you can see that this comma-delimited record indicates that loan application #10032, submitted September 1st at 12:59, during the sixth hour of the workday, requested a car loan of $32,923.66 for a customer named GIRON LLC.

3 On the OTD Tester toolbar palette, click: 🔄 **Refresh the Data**

4 In the left pane of the tester, expand myBamOtd and check results. See Figure 24.

Figure 24   myBamOtd, Fully Configured and Tested
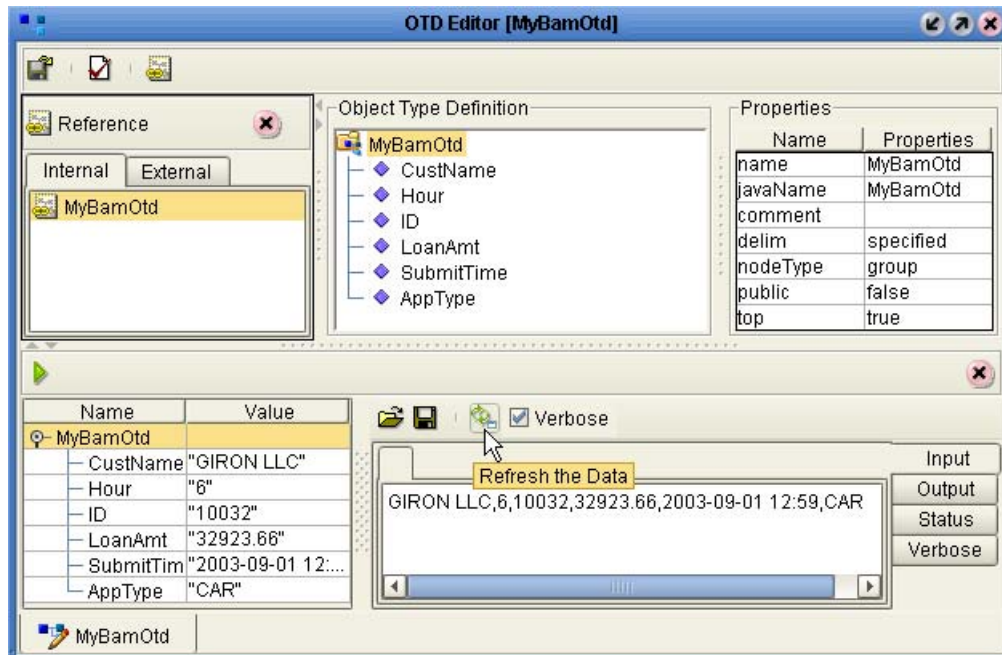
5 Before continuing, save your work and close all canvases.

Result: The OTD for unmarshaling and parsing the sample data is now ready for use.

5.2.4 Creating and Validating the Data Definition

To create an eBAM Application

1 Right-click myBamProject; on the popup menu, click New > **eBAM Application**

2 In the wizard, for eBAM Application Name, type **myBamLoan** and then click Next.

3 For Data Definition Name, type **myLoanDataDef** and then click Next.

4 In the Define Data step, click **Add** five times. Then, for each of the six new records, double-click the field name to edit it to resemble the names below, pressing ENTER after each key-in, and also change the Type values to those specified below.

Field Name	Default Value	Type
CustName		varchar
Hour		integer
ID		integer
LoanAmt		float
SubmitTime		timestamp
TypeOfApp		varchar

5 After creating and configuring the six rows, click Next, and then click Finish.

Result: The project tree displays myBamLoan with daughter folders (Data Definitions, Alert Conditions, and Charts); on the canvas, the eBAM Editor displays your metadata.

To validate the metadata definition

1 In the eBAM Editor, on the tool palette, click: 📝 **Show Sample Data**

The Sample Data pane appears below the main eBAM Editor.

2 On the Sample Data tool palette, click: 📥 **Import**

3 In the File Import wizard's **Select File** step, browse to the location of the sample data files (such as C:\temp\eBAM\Sample\Data\In) and select: **LoanDataAllSmall.txt**

4 In the formatting step, for File format, select **Delimited** and then click **Next**.

Table name:	LOANDATAALLSMALL_TXT
Encoding scheme:	ASCII (ISO646-US)
File format:	Delimited

5 In the parsing step, keep all defaults and then click **Next**.

Default SQL Type	varchar
Record Delimiter	{newline (lf)}
Field Delimiter	{comma}
Text Qualifier	none
First line contains field names?	False

6   In the record layout and field properties step, click **Preview**. See Figure 25.

Figure 25   Show Sample: Preview of Field Layout
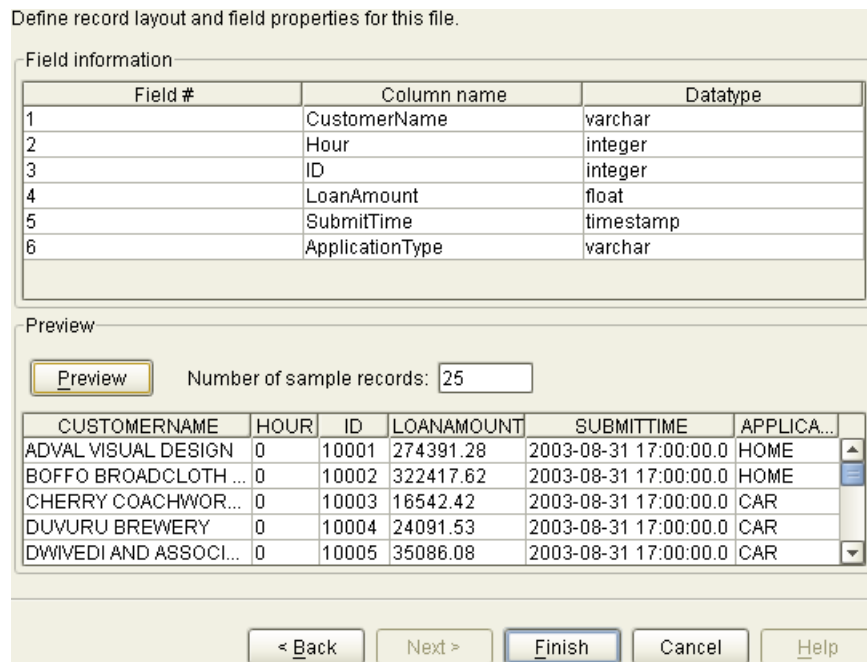
7   Click Finish. The entire data set is parsed and displayed in the Sample Data pane. See Figure 26.

Figure 26   Sample Data Displayed in eBAM Editor
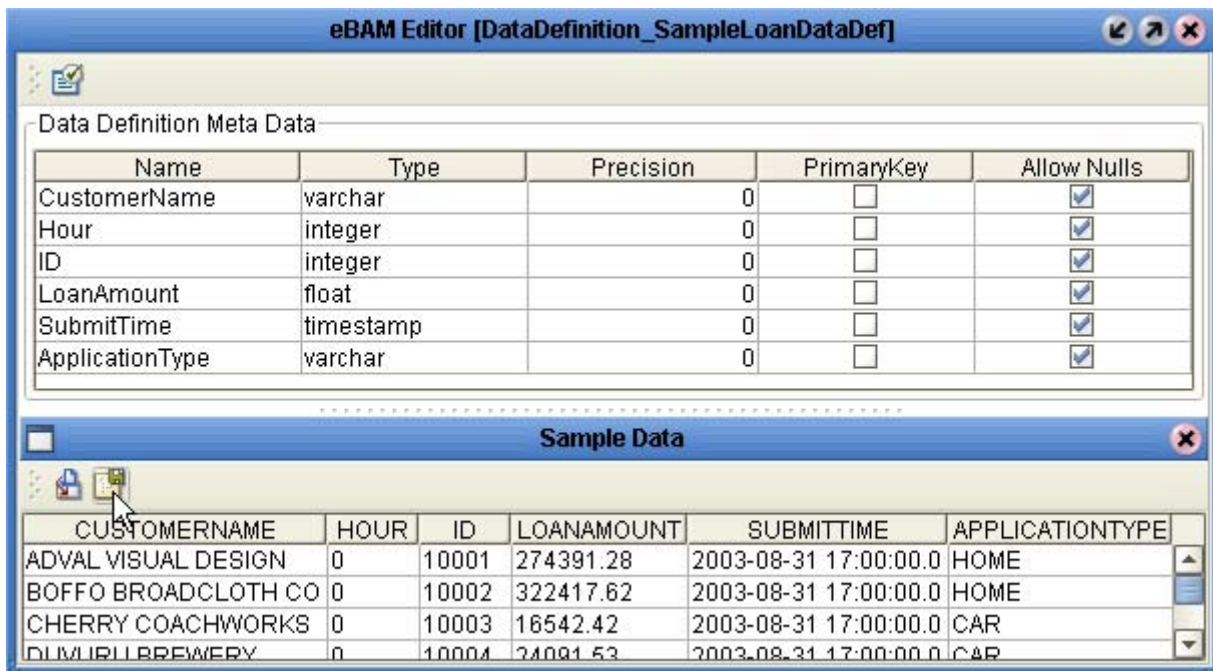
8   On the tool palette, click ▣ **Commit**. Then save your work and close all canvases.

5.2.5 Creating and Configuring the Alert Condition

If the loan application is for an amount in excess of $1,000,000.00, a notification should be issued. In the following steps, you will create an alert condition, use the graphical editor to create a query detecting the condition, and then set the notification interval to one minute, with resends every three minutes.

To create a new alert condition

1 In the project tree, right-click myBamLoan and, on the popup context menu, click: **New Alert Condition**

2 In step 1 of the wizard, specify a name: **VeryBigLoan**

3 In step 2 of the wizard, select the check box for **myBamLoan:myLoanDataDef**

Result: The project tree displays VeryBigLoan under Alert Conditions; on the canvas, the Alert Condition Editor displays myLoanDataDef and an empty dataset view.

To configure a condition on a data definition field

1 In the Alert Condition Editor canvas, right-click **myLoanDataDef** and, on the menu, click: **Properties**

2 In the Properties dialog, click **Condition**, and then click the ellipsis [...] to open the Condition editor.

3 On the right side of the Condition editor, click the **Graphical** tab. On the left side, under Columns, double-click the data definition to expand it. See Figure 27.

Figure 27   Preparing to Set a Condition on the Data Definition
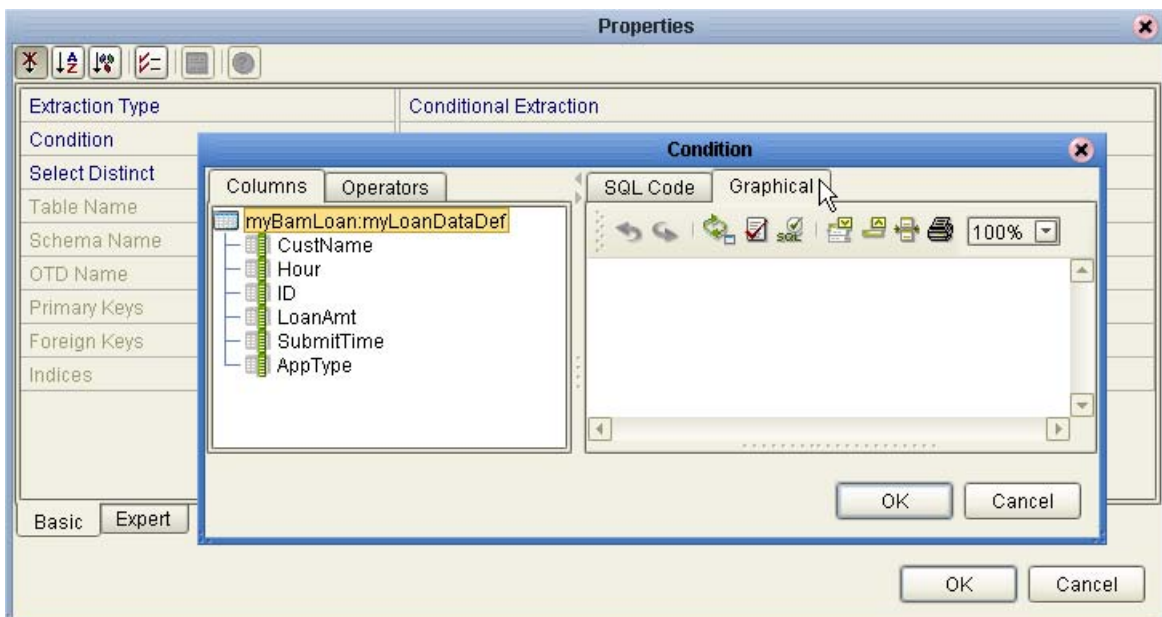
4 From the Columns tree, drag **LoanAmt** onto the upper left of the Graphical canvas.

5 Click the Operators tab and expand the folders for **Comparison** and **SQL-Specific**.

6 From Comparison, drag **greater than** onto the upper right of the Graphical canvas.

7 From SQL-Specific, drag **literal** onto the lower left of the Graphical canvas.

8 In the New Literal Object dialog, specify Type as **float** and Value as: **1000000.00** (Be sure that the number you enter includes a decimal point, to make it a **float**).
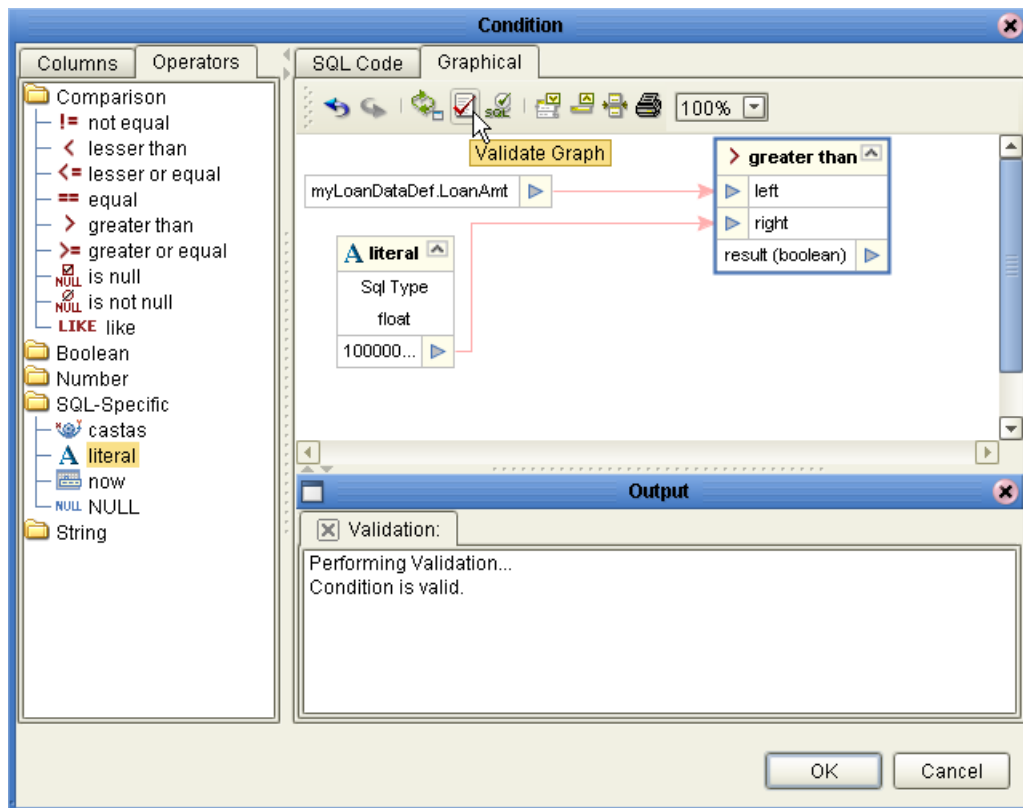
9 In the Graphical canvas, drag ....**LoanAmt** onto **left** and drag **1000000** onto **right**.

10 On the Graphical toolbar palette, click ☑ **Validate Graph**. See Figure 28.

Figure 28 Validating the Condition on the Data Definition

11 Click OK to redisplay the Properties dialog, with the value for **Condition** showing the SQL text of the query you created graphically. Click OK.

Result: The Alert Condition Editor now displays the data definition with its a small with a small "filter" icon (▽) on the **LoanAmt** field, indicating that a condition has been set on this field.

To create report fields and map data definition fields to them

1 In the Alert Condition Editor, right-click **DatasetView** and, on the menu, click: **Configure**

2 In the **Configure Dataset View** dialog, click **Add** five times, yielding six fields.

3 Double-click each of the six fields, edit the field name appropriately (to: Customer, Hour, ID, LoanAmount, SubmitTime, and TypeOfApplication), and click OK.

4 In the Alert Condition Editor, drag each field from myLoanDataDef to the corresponding field in DatasetView. Data type conversions are done automatically where needed; for timestamps converted to **varchar**, permit truncation to occur.

To configure properties of the alert condition

1 In the Alert Condition Editor, right-click the blank canvas and, on the menu, click: **Properties**

2 In the Alert Condition Properties dialog, change the Notification Interval from 10 Minutes to **1** Minute; change the Resend Frequency from 1 Day to **5 Minutes**; and for Keys, select the **ID** check box. See Figure 29.

Figure 29   Setting Properties for the Alert Condition
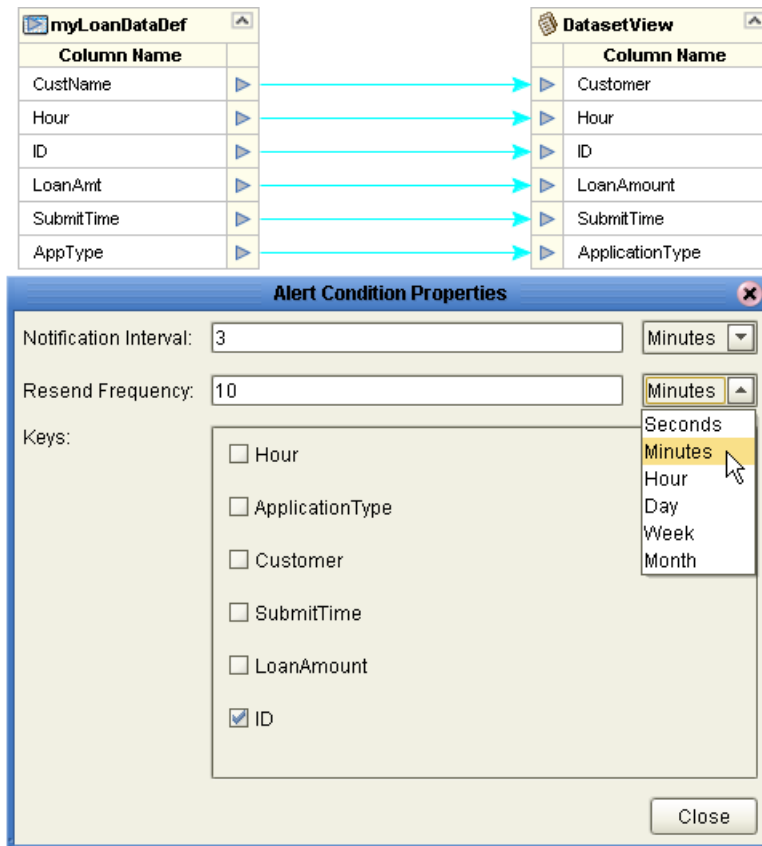
3 Click **Close**.

Result: You have created an alert condition such that any time the loan amount exceeds one million, a notification will be sent, based on the ID field of the data definition, with a notification interval of three minutes and resend frequency of ten minutes.

For further information on any of the material in the preceding section, refer to **"Setting Up the Data Definition for an eBAM Application" on page 19** and **"Setting Up the Application's Alert Conditions" on page 28**.

5.2.6 **Creating and Configuring Charts**

In this section, you will set up the data definition conditions and chart properties for a pie chart that organizes the raw data on loan applications into hourly slices. For further information on any of the material in this section, refer to **"Setting Up the Data Definition for an eBAM Application" on page 19** and **"About Charts" on page 32**.

To create a new pie chart

1 In the project tree, right-click myBamLoan and, on the popup context menu, click: **New Chart**

2 In step 1 of the Chart wizard, specify Type as **Pie Chart** and a name: **Loans by Hour**

3 In step 2 of the wizard, select the check box for **myBamLoan:myLoanDataDef**

4 In step 3 of the wizard, adjust the parameters for Common, Chart, and Pie-Specific to suit your taste (see tables in **"About Charts" on page 32**), and then click **Finish**.

Result: The project tree displays **Loans by Hour** under Charts; on the canvas, the Chart Editor displays myLoanDataDef and an empty dataset view.

To configure a view that groups by a field (Hour) to count a key field (ID)

1 In the Chart Editor, right-click **dataset-view** and, on the menu, click: **Configure**

2 In the Configure Dataset View dialog, click **Add** once, yielding two fields.

3 Double-click each of the two fields, edit the field name appropriately (to **Hour** and **TotalCount**), and the click OK.

4 In the Chart Editor, right-click the blank canvas and, on the menu, click: **Properties**

5 In the Chart Properties dialog, in the tree on the left side, click: **GroupBy**

6 Under Available Group-By Columns, click dataset-view.**Hour** and then click **Add**. See Figure 30.

Figure 30 Setting Chart Properties: Available and Selected "GroupBy" Columns
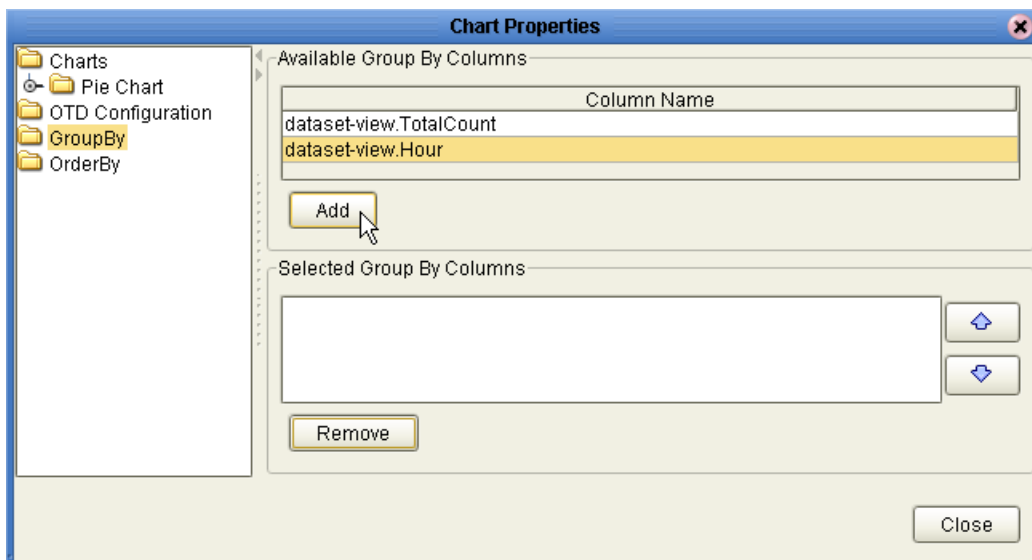
The selected column name moves into the **Selected Group-By Columns** pane.

7   Click **Close**.

8   In the Chart Editor, on the tool palette across the top, click the chevron (») to open the Operator Palette. Then, in the Operator Palette, from the Number tab, drag **count** to the canvas. See Figure 31.

Figure 31   Dragging the "count" Operator from the Operator Palette
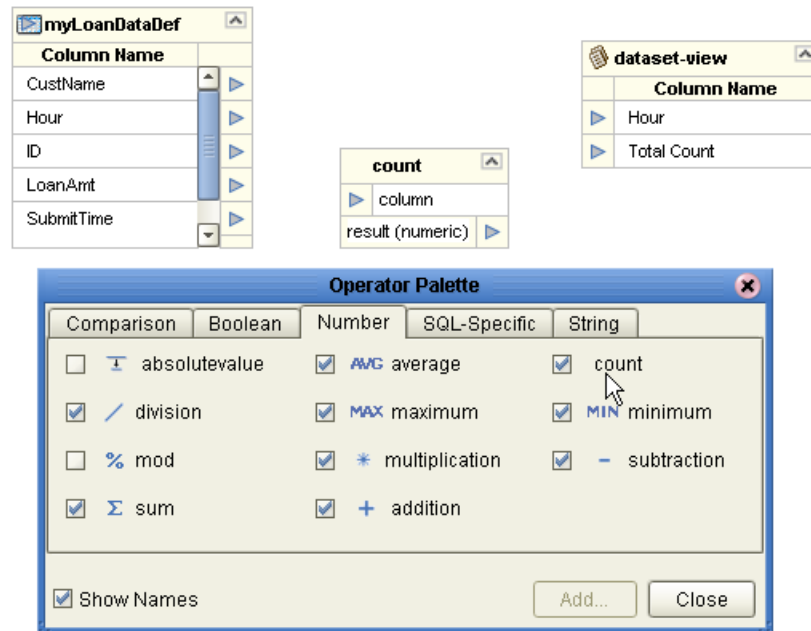
9   Click **Close**.

10   From the data definition, drag **Hour** to [view]**Hour** and drag **ID** to [count]**column**.

11   From the **count** operator, drag **result** to [view]**TotalCount**. See Figure 32.

Figure 32   Completed Mapping for Chart Editor
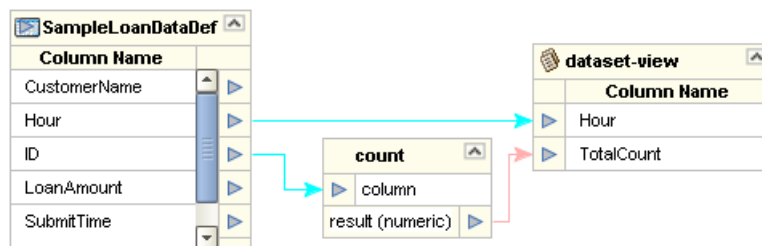
12   Save your work and close all canvases.

Result: You have created a KPI—a pie chart named **Loans by Hour** that tracks and counts how many loan applications were submitted for each hour of the day and displays the data using this view.

Now that you have finished the eBAM-specific work, the only remaining design work is to create and configure the business processes and connectivity map.

5.2.7 Creating and Configuring Business Processes

The implementation scenario requires two business processes:

- The **myLoanProcess** process, triggered by an input file record, unmarshals the data to the OTD created earlier, aggregates it (in accordance with the data definition) with other data received, and stages a "success" message to be written to a file.

- The **myAlertProcess** process, triggered by the Alert notify, marshals the data using the same OTD, flags it by prepending the current time and a user-specified string, and then stages the entire record to be written to a different output file.

**To create the business processes: myAlertProcess and myLoanProcess**

1 In the project tree, right-click myBamProject and, on the popup context menu, point at New and click: **Business Process**. Rename the new business process to: **myAlertProcess**

2 Again, right-click myBamProject > New > **Business Process**. Rename the second new business process to: **myLoanProcess**

Result: The eInsight Business Process Designer canvas is blank but for Start and End.

To populate myLoanProcess with operations: receive, unmarshal, add, write

1 In the project tree, under the SeeBeyond project, open eWays > File > **FileClient**, and drag the FileClient.**receive** operation onto the canvas, just to the right of Start.

2 Under the myBamProject project, open **myBamOtd** and drag its **unmarshal** operation onto the canvas, just to the right of receive.

3 Still under the myBamProject project, open myBamLoan > Data Definitions > **myLoanDataDef** and drag its **add** operation just to the right of unmarshal.

4 Under SeeBeyond > eWays > File > **FileClient**, drag the **write** operation onto the canvas, just to the right of add and just to the left of End. See Figure 33.

Figure 33 Operations for Activities in myLoanProcess
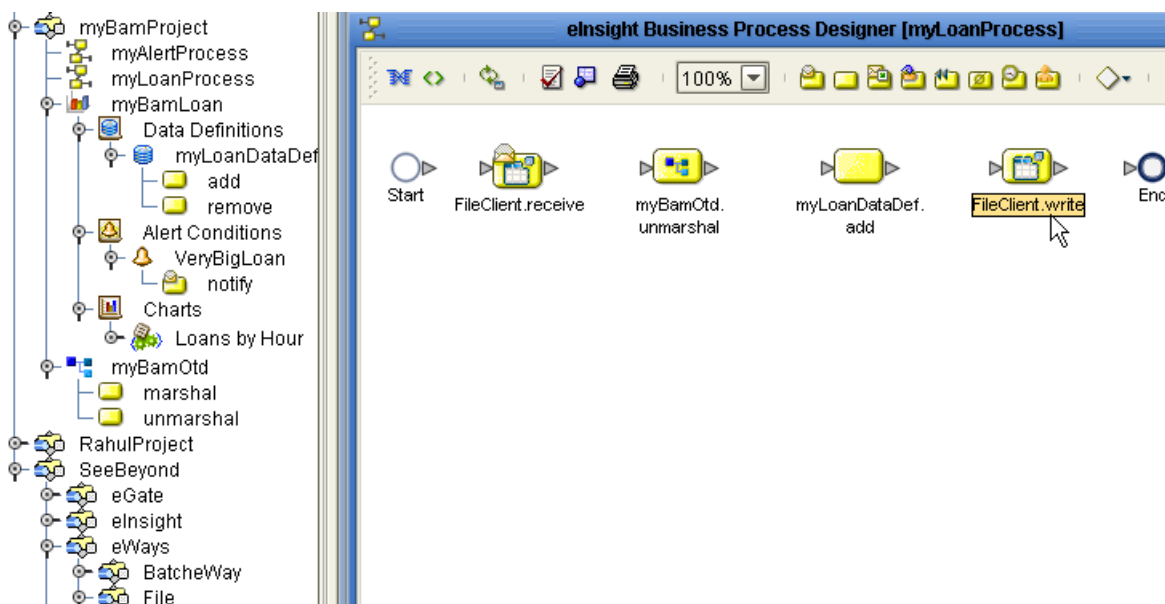
To connect the myLoanProcess activities

1 Connect each activity to its neighbor. In other words, connect Start to receive; receive to unmarshal; unmarshal to add; add to write, and write to End.

2 Right-click the connection from receive the unmarshal and, on the menu, click: **Add Business Rule**

3 In the Business Rule Designer: Open both the Output and Input OTDs, and then drag the **text** node onto the **contents** node. See Figure 34.

Figure 34   myLoanProcess (Connection 2: receive to unmarshal)
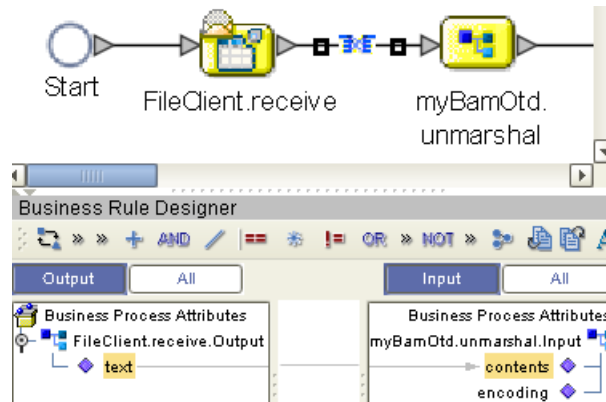
4 For the third connection: Add a business rule on the line from **unmarshal** to **add**. In the Business Rule Designer, fully open both the Output and Input nodes, and then drag each of the six nodes onto its corresponding node. See Figure 35.

Figure 35   myLoanProcess (Connection 3: unmarshal to add)
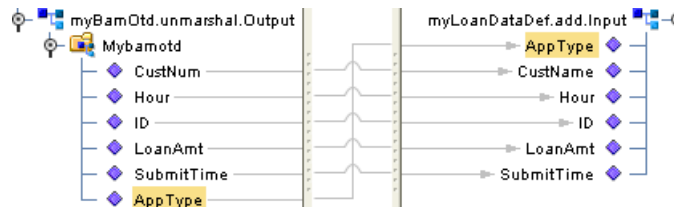
5 For the fourth connection: Add a business rule on the line from **add** to **write**. Then, in the Business Rule Designer (left side): click All; find unmarshal.Output, open myBamOtd, and highlight its **ID** node. On the right side: Open the Input node and highlight its **text** node. From the method palette, drag a **A** **string literal** method into the upper left and give it the value "**Processed Loan ID:**" (with the colon, but without the quotes). From the method palette, drag a 🐱 **concat** method into the middle right; drag the literal onto its **string1**, and then drag the highlighted **ID** onto its **string2**. Finally, from the concat, drag its **return string** onto the **text** node on the right. See Figure 36.

Figure 36 myLoanProcess (Connection 4: add to write)
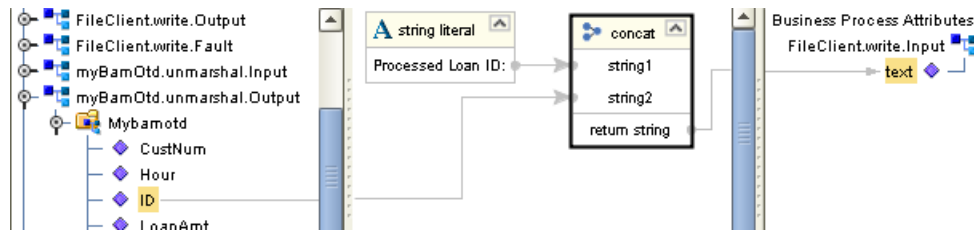
6 Validate the model (if you receive warning messages, you can delete unused attributes such as Fault if you want), save your work, click the myAlertProcess tab, and close the tab for myLoanProcess.

The Business Process Designer now displays the **myAlertProcess** tab and its canvas, which is blank except for the Start and End activities.

To populate myAlertProcess with operations: notify, marshal, write

1 Under the myBamProject project, open myBamLoan > Alert Conditions > **VeryBigLoan** and drag its **notify** operation just to the right of Start.

2 Under the myBamProject project, open **myBamOtd** and drag its **marshal** operation onto the canvas, to the right of notify.

3 In the project tree, under the SeeBeyond project, open eWays > File > **FileClient**, and drag it **write** operation onto the canvas, to the right of marshal. See Figure 37.

Figure 37 Operations for Activities in myAlertProcess

To connect the myAlertProcess activities

1 Connect each activity to its neighbor. In other words, connect **Start** to **notify**; **notify** to **marshal**; **marshal** to **write**, and **write** to **End**.
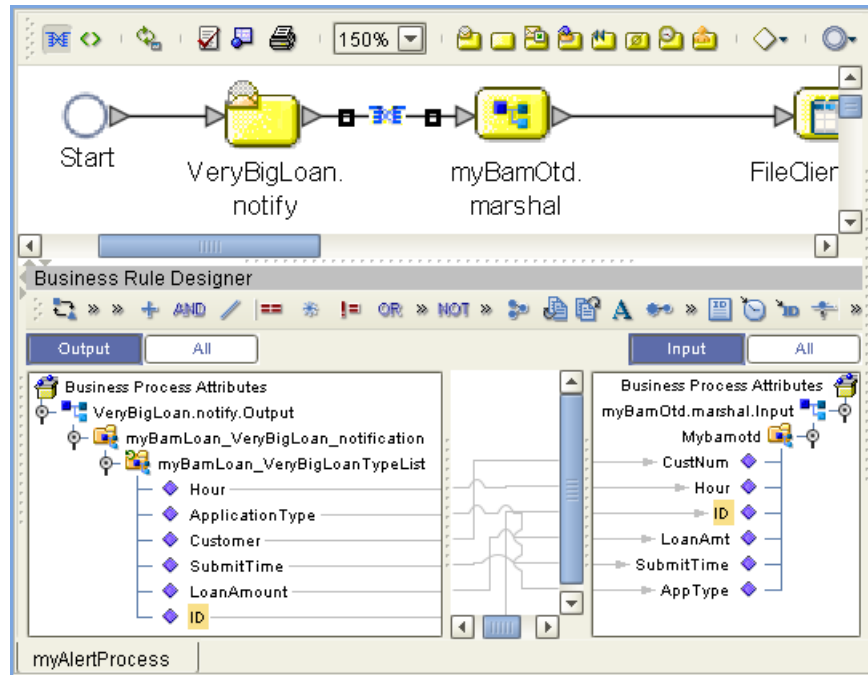
2 For the second connection: Add a business rule on the line from **notify** to **marshal**. In the Business Rule Designer, fully open both the Output and Input nodes, and then drag each of the six nodes onto its corresponding node. See Figure 38.

Figure 38 myAlertProcess (Connection 2: notify to marshal)

3 For the third connection: Add a business rule on the line from **marshal** to **write**. Next, in the Business Rule Designer, open the OTD (myBamOtd.marshal.Output) on the left and highlight its **contents** mode, and then open the FileClient.write.Input OTD on the right and highlight its **text** node. Then, from the method palette, drag a ☐ **get current time** method into the middle left, and then drag a **A** **string literal** method into the lower left and give it the value "**::GreaterThanOneMillion::**" (with the colons but without the quotes). Drag a ☐ **concat** method into the middle center; drag **current time** onto its **string1**, and then drag the literal onto its **string2**. Drag another ☐ **concat** method, this time into the upper right; drag **return string** onto its **string1**, and then drag [marshal.Output.]**contents** onto its **string2**. Finally, from the second concat, drag its **return string** onto the [write.Intput.]**text** node on the right. See Figure 39.

Figure 39 myAlertProcess (Connection 3: marshal to write)
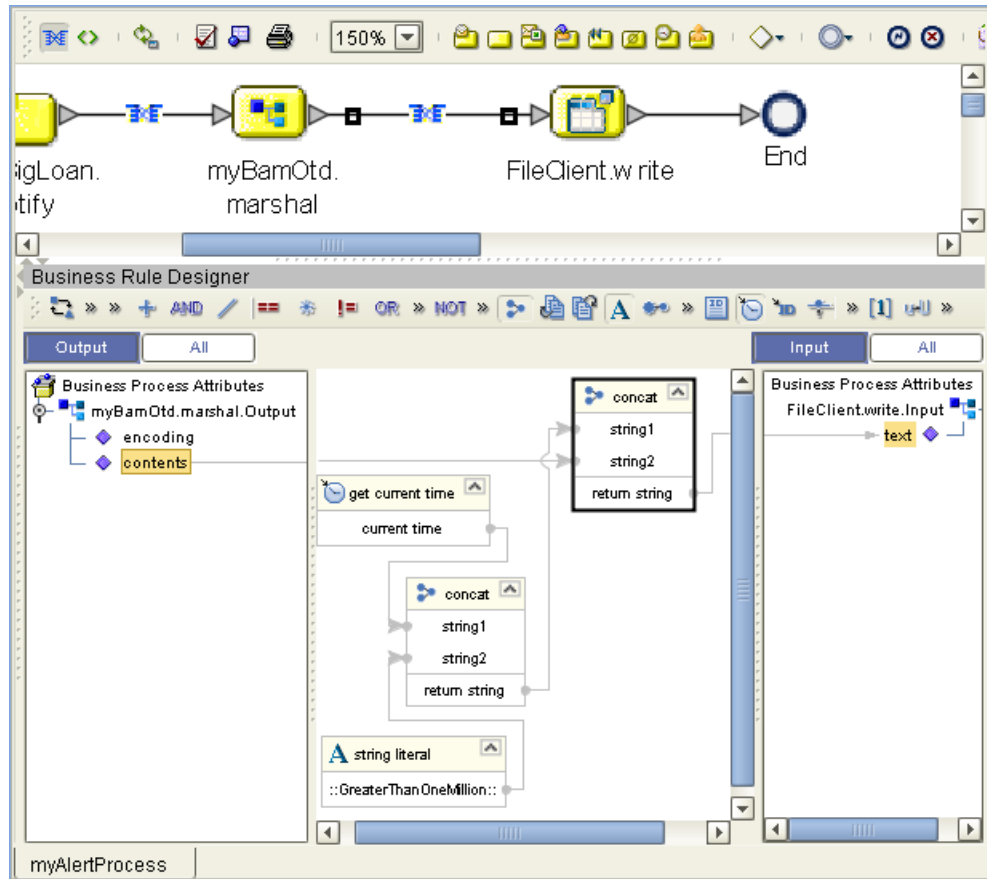
4   Validate the model (if you receive warning messages, you can delete unused attributes such as Fault if you want), save your work, and close all canvases.

5.2.8 Creating and Configuring the Connectivity Map

The implementation scenario requires one connectivity map, named **myLoanCMap**, containing one input file, two output files, and the following three services:

- **myLoanProcess1** receives from the input file, talks to myBamLoan1, and sends its output to a simple output file.

- **myBamLoan1** listens to myLoanProcess1 and talks to myAlertProcess1.

- **myAlertProcess1** listens to myBamLoan1, and its output goes to an alert file.

To create the connectivity map

1   In the project tree, right-click myBamProject and, on the popup context menu, point at New and click: **Connectivity Map**

2   Rename the new business process to: **myLoanCMap**

Result: In the canvas, the Connectivity Map Editor is blank.

To populate myLoanCMap with components

1 In the Connectivity Map Editor, open the tool for ▢▾ External Applications and enable **File External Application**.

2 From the tool palette, drag a File to the upper left of the canvas and rename it to: **myFileInLoan**

3 From the tool palette, drag a File to the upper right of the canvas and rename it to: **myFileOutLoan**

4 From the tool palette, drag a File to the lower right of the canvas and rename it to: **myFileOutAlert**

5 From the project tree, drag **myLoanProcess** to the upper center of the canvas and keep the default name: myLoanProcess1

6 From the project tree, drag **myAlertProcess** to the lower center of the canvas and keep the default name: myAlertProcess1

7 From the project tree, drag **myBamLoan** to the middle center of the canvas and keep the default name: myBamLoan1

Result: See Figure 40.

Figure 40   Components in Connectivity Map myLoanCMap
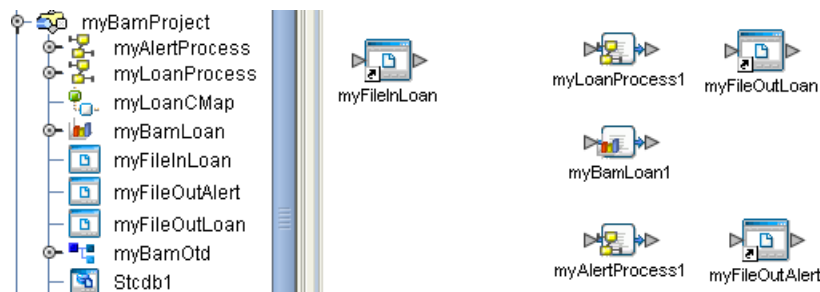
**To connect the components in myLoanCMap**

1 In the Connectivity Map Editor, double-click **myLoanProcess**1 to open it, and then:

   A From Implemented Services, drag **FileSender** leftward onto **myFileInLoan**.

   B From Invoked Services, drag **FileReceiver** rightward onto **myFileOutLoan**.

   Keep myLoanProcess1 open; you will minimize it after using it in the next step.

2 Double-click **myBamLoan1** to open it, and then:

   A From Implemented Services, drag **add** upward onto **myLoanDataDef**. You can now minimize myLoanProcess1.

   Keep myBamLoan1 open for now; you will close it after using in the next step.

3 Double-click **myAlertProcess** to open it, and then:

   A From Implemented Services, drag **VeryBigLoan** upward onto **VeryBigLoan_notify**. You can now minimize myBamLoan1.

   B From Invoked Services, drag **FileReceiver** rightward onto **myFileOutAlert**.

Result: All components are connected, but the connections are not yet configured; see Figure 41.

Figure 41 Connected Components in myLoanCMap
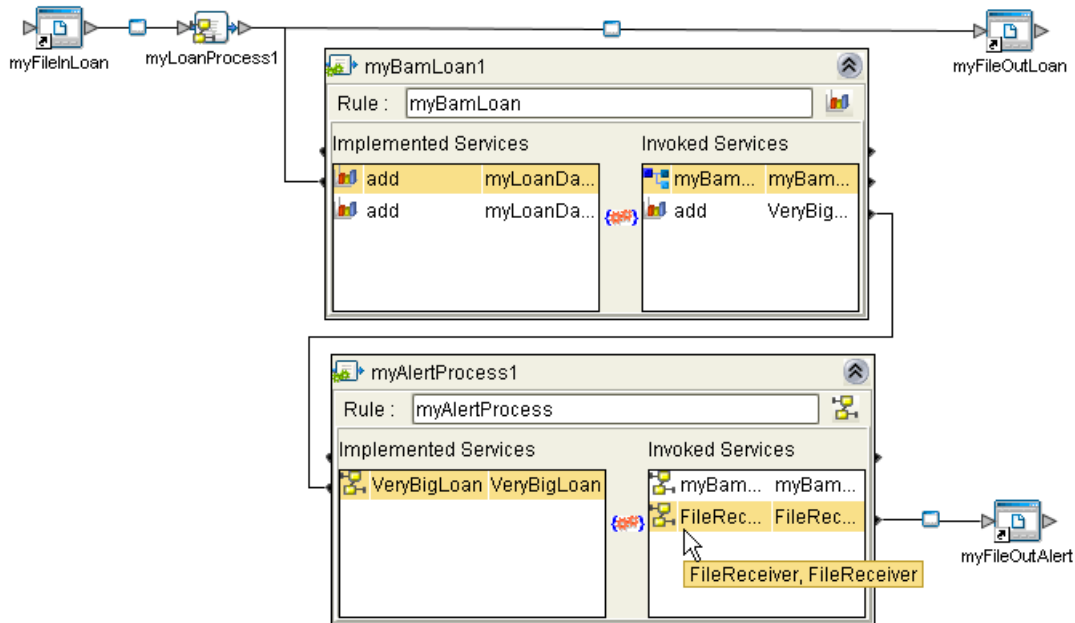
To configure the eWays

1 In the Connectivity Map Editor, double-click the eWay connecting myFileInLoan with myLoanProcess1, choose the **Inbound** File eWay template, and then make the following changes to the default parameter settings:

  ◆ For **Directory**, change to: **C:\temp\eBAM\Sample\Data\In**

  ◆ For **Multiple records per file**, change to: **True**

  ◆ For **Remove EOL**, change to: **True**

2 Verify that all parameters are correctly set, and then click OK.

3 Double-click the eWay connecting myLoanProcess1 with myFileOutLoan, choose the **Outbound** File eWay template, and then make the following change to the default parameter settings:

  ◆ For **Directory**, change to: **C:\temp\eBAM\Sample\Data\Out**

4 Verify that all parameters are correctly set, and then click OK.

5 Double-click the eWay connecting myAlertProcess1 with myFileOutAlert, choose the **Outbound** template, and then make the following changes:

  ◆ For **Directory**, change to: **C:\temp\eBAM\Sample\Data\Out**

  ◆ For **Output file name**, change to: **alert%d.dat**

6 Verify that all parameters are correctly set, and then click OK.

7 Save your work and close all canvases.

Result: All components are connected and configured. The next step is to create and activate a deployment profile for the project.

5.2.9 ## Activating and Running the Project

You will create a deployment profile named **myBamLoanDP**, which you will activate and deploy to the logical host that is currently running. (If it is not already running, see **"Starting the Logical Host" on page 45**.)

To create the deployment profile

1 In the project tree, right-click myBamProject and, on the popup context menu, point at New and click: **Deployment Profile**

2 In the dialog box, name it **myBamLoanDP**, and be sure it references the myBamEnv environment before clicking OK.

Result: The project tree displays the new object, and the Deployment Editor shows the six components and the three servers to which you will assign them.

To assign components to servers

1 One by one, drag the three services (myLoanProcess1, myAlertProcess1, and myBamLoan1) into LogicalHost1 and onto **IntegrationSvr1**.

2 Drag the inbound File eWay (myFileInLoan->myLoanProcess1) into the **myExternalFileIn** server.

3 One by one, drag the two outbound File eWays (myLoan...->myFileOutLoan and myAlert...->myFileOutAlert) into the **myExternalFileOut** server.

4 Save your work. See Figure 42.

Figure 42 Components from myBamProject Assigned to Servers in myBamEnv
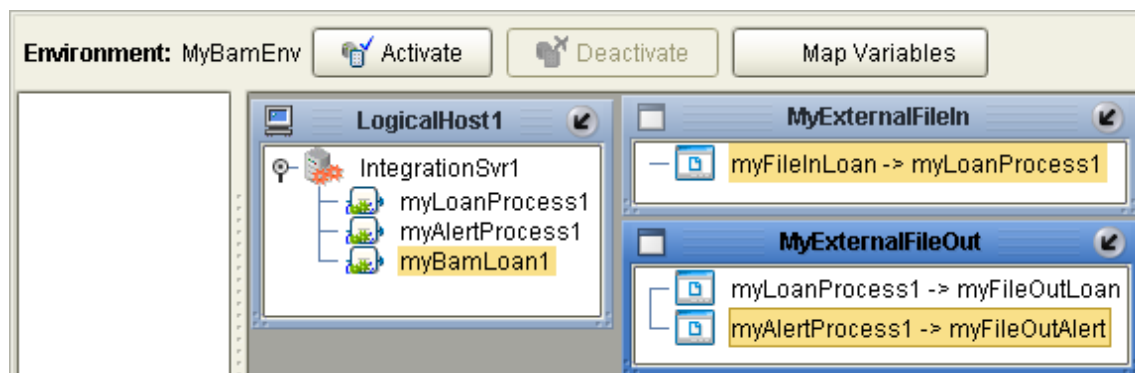
To activate and run the project

1 In the Deployment Editor, after each of the six components in myBamProject has been assigned to one of the three servers in myBamEnv, click **Activate**

  ◆ Or, if you have previously activated this deployment profile, click **Reactivate**

2 After activation is successfully completed, when the Activate dialog box asks whether you want to apply the changes to the logical host immediately, click: **Yes**

5.2.10 Monitoring the Key Performance Indicators with Live Data

Now that you have set up the data definitions and set up a pie chart and an alert condition, you are ready to display the results and see how they update in real time as you feed sample data to the project.

To start monitoring the project

1 If you have not already done so, log in to Enterprise Manager.

2 In Enterprise Manager, enter the URL for monitoring the eBAM charts for this project. This takes the following form:

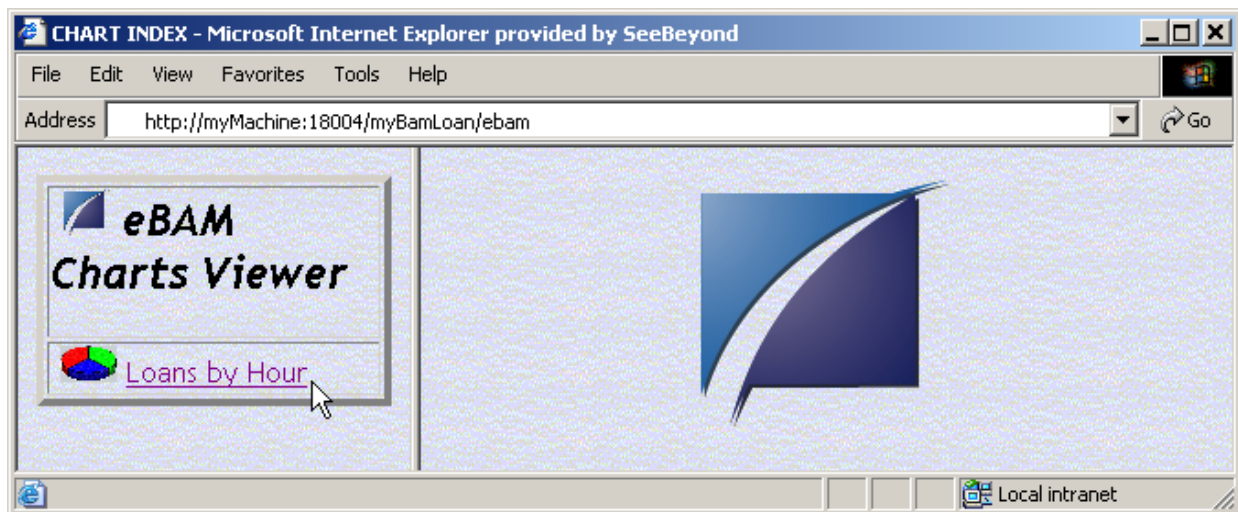> **http://***hostname***:18004/***BamApplicationName***/ebam**

(This is case-sensitive; the final four letters must be **ebam**, not eBAM.) For example:

> http://myMachine:18004/myBamLoan/ebam

If your Integration Server does not use port 18004, then substitute the correct port number in the URL.

Result: See Figure 43.

Figure 43   Initial eBAM Charts Viewer (No Data)

To feed sample data to the project

1 Browse (or open a command prompt and change directories) to the location where you installed the sample input data, and verify the presence of both sample data files (LoanDataAllSmall.txt and LoanDataOneBig.txt). For example:

```
C:
cd \temp\eBAM\Sample\Data\In
dir
```

2 Rename a copy of LoanDataOneBig.txt to: **input_OneBig.txt**

3 Watch as the file is picked up by the File eWay, renaming it to **input_OneBig.~in**

4 In the Charts Viewer, if you have not already done so, click the **Loans by Hour** link and see the pie chart display a 360-degree "slice" representing a single data item.

5   Browse the output contents, in C:\temp\eBAM\Sample\Data\Out\output*.dat

6   Rename a copy of LoanDataAllSmall.txt to: **input_AllSmall.txt**

7   Watch as the file is picked up by the File eWay, renaming it to **input_AllSmall.~in**

Result: The Charts Viewer displays a pie chart whose slices show how many loans were submitted during each work hour monitored by the sample data file. See Figure 44.

Figure 44   eBAM Charts Viewer Showing Pie Chart of Loans By Hour