IBM DB2 Alphablox

Administrator's Guide

Version 84

SC18-9432-03

IBM DB2 Alphablox

Administrator's Guide

Version 84

SC18-9432-03

Note!
Before using this information and the product it supports, read the information in "Notices" on page 153.

Fourth Edition (September 2006)

This edition applies to version 8, release 4, of IBM DB2 Alphablox for Linux, UNIX and Windows (product number 5724-L14) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. DB2 Alphablox overview

DB2 Alphablox runs in a J2EE application server environment and provides services for creating web-based analytical applications. DB2 Alphablox can be integrated with leading application servers, including IBM® WebSphere®, BEA WebLogic, and Apache Tomcat. This chapter provides an overview of DB2 Alphablox, describes how DB2 Alphablox fits into a J2EE environment, and describes the DB2 Alphablox architecture.

DB2 Alphablox overview

DB2 Alphablox provides the ability to rapidly create custom, web-based applications that fit into the corporate infrastructure and have the ability to reach a wide range of users, both inside and outside the corporate firewall. Applications built with the DB2 Alphablox platform run in standard web browsers, allowing real-time, highly customizable multidimensional analysis in a web browser.

The following are some features available in the DB2 Alphablox platform:

- access and interact with data in multidimensional and relational databases
- create structured reports sourced from relational databases
- choose from a wide variety of charts to display data
- create applications that write data back to the database, particularly useful in "what-if" financial planning applications
- with multidimensional data sources, allow users to interact with the different levels of data (for example, filter, drill down, etc.) to interactively display the exact view of the data desired
- users access an intuitive user interface making analysis of the data easy and powerful
- a single application can access multiple data sources
- integrates into a wide variety of enterprise infrastructure components, including application servers (IBM WebSphere and BEA WebLogic)

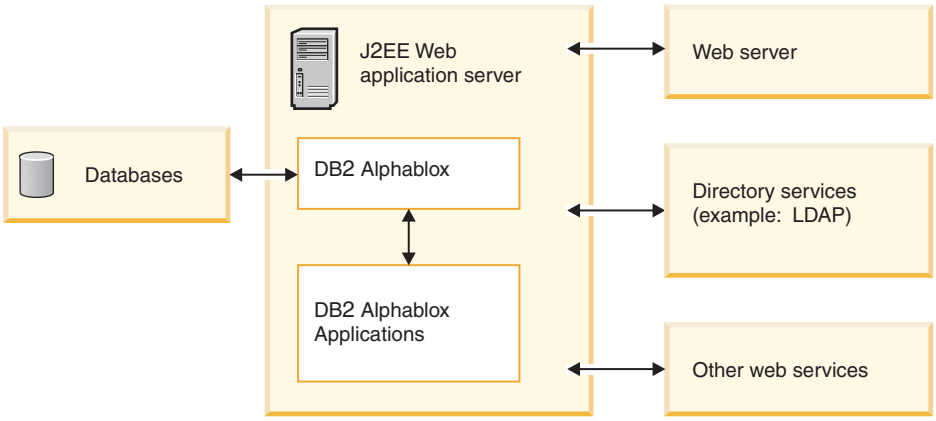
DB2 Alphablox provides a wide variety of application programmer interfaces (APIs) so developers can create custom applications. The DB2 Alphablox APIs are written in the Java™ programming language, and application developers can access them using Java that is executed on the server or via JavaScript™ that is interpreted in the browser.

The remainder of this chapter describes how DB2 Alphablox fits into a J2EE environment, explains the components of DB2 Alphablox, and describes the architecture of DB2 Alphablox, the core component of DB2 Alphablox. For detailed information about creating DB2 Alphablox applications, see the *Developer's Guide*. For syntax and reference information about the DB2 Alphablox API, see the *Developer's Reference*.

DB2 Alphablox in a J2EE environment

DB2 Alphablox runs within a J2EE application server (IBM WebSphere or BEA WebLogic) or Apache Tomcat. For a list of supported application servers and their versions, see the *Installation Guide*.

The following figure shows how DB2 Alphablox fits into a J2EE environment



Advantages of running within an application server

Running within an application server has several advantages:

- it allows easy integration of open standards-based J2EE components into the DB2 Alphablox environment
- it provides access to the services provided by the various web services
- it provides access to the all the services provided by the Java Runtime Environment, as well as any Java extensions available in a particular application server
- it allows DB2 Alphablox to focus on providing a platform for analysis, leaving things like web serving, security, etc. to vendors who specialize in these areas
- it provides a platform for J2EE developers that they are familiar with using J2EE technologies, such as JavaServer Pages (JSP), Java, JavaBeans™ components, and XML.

The tight integration with the application server provides DB2 Alphablox with all of these advantages while allowing developers access to the rich set of DB2 Alphablox APIs.

WebSphere and WebLogic application server configurations

DB2 Alphablox can be configured to run in IBM WebSphere or BEA WebLogic application servers. The basic operation of DB2 Alphablox is the same when running with WebSphere, WebLogic, or Tomcat; only a few minor administration details are different (for example, you must register a new application in WebSphere after creating it in DB2 Alphablox and startup occurs with the application server startup).

When running with a commercial application server such as WebSphere or WebLogic, you have access to all the tools, scalability, and services available on that platform, as well as all the services available in the DB2 Alphablox platform.

The installation provides options to configure WebSphere, WebLogic, and Tomcat. For details, see the *Installation Guide*.

Because DB2 Alphablox applications are J2EE applications, the applications will work in different configurations. Therefore, it is possible to develop and test your application in a Apache Tomcat configuration and then deploy it in your enterprise application server configuration. Occasionally, there are minor runtime differences

between when running on different platforms, so it is necessary to test an application in a different configuration before deploying it, but as long as there are no platform-specific services used in the application, there are few issues migrating from one configuration to another.

Apache Tomcat configuration

As one option, DB2 Alphablox can use the Apache Tomcat application server. In the Apache Tomcat configuration, the DB2 Alphablox installer uses the specified Apache Tomcat server (version 3.2.4 only). For information on Apache Tomcat, see http://jakarta.apache.org/tomcat/. To obtain a copy of the Tomcat 3.2.4 version, see http://archive.apache.org/dist/tomcat/tomcat-3/archive/v3.2.4/.

When using the Apache Tomcat with DB2 Alphablox, there are no Tomcat-specific administration tasks needed in order to use the DB2 Alphablox platform. The Apache Tomcat configuration is a scalable solution, able to use the clustering solution described in Chapter 16, "Using a Database Repository," on page 109.

The Apache Tomcat configuration includes full security with support for users, groups, and roles. You can use any of the supported external web servers (for example, Microsoft® IIS, Sun iPlanet, or the Apache web server) in the Apache Tomcat configuration, or you can have Tomcat handle HTTP requests.

Tip: The HTTP server included in the Apache Tomcat configuration is good for development or for small production systems, but for larger production systems, you may want to consider using an external web server, which has many caching and page serving features, to improve HTTP performance.

Components of the DB2 Alphablox platform

The DB2 Alphablox platform consists of the following elements, which are discussed in this section:

DB2 Alphablox

DB2 Alphablox has a robust architecture specifically designed for rapid development, deployment, and use of analytic applications. Key features provided by the architecture include:

Data Adapters

DB2 Alphablox includes a Data Manager specifically designed for connectivity to a variety of databases. The Data Manager is responsible for accessing, browsing, querying, and retrieving data from both relational and multidimensional databases (including DB2 Alphablox cubes). Connection to each database is implemented through plug-in adapters. For relational databases, the plug-in adapters are

typically JDBC drivers. Each adapter encapsulates database-specific connection information and processing, greatly reducing the effort required to connect to additional databases.

The Data Manager and associated data adapters provide support for the following:

- Browsing a collection of preconfigured, named database connections (called data sources).
- Providing access to the available databases within each data source to DB2 Alphablox applications.
- Publishing the compatible query types for a specific data source.
- Allowing the traversal of a database's metadata.
- Managing database connections for user sessions.
- Translating query objects into the underlying native query language.
- Executing queries against a database.
- Interrogating a result set's data and schema.
- Processing a result set by displaying, pivoting, expanding, sorting, and drilling.
- Creating applications that take user input and write the data back to the underlying database (often used in "what if" scenarios for budget applications).

Blox Components

DB2 Alphablox applications use building blocks, called Blox components, to access and present real-time enterprise data through a standard web browser. Blox components are reusable software components that are combined, or *assembled*, on a standard JSP page, and result in interactive applications accessed through web browsers, either within a corporate intranet or over the Internet.

These tested software components provide applications with the following areas of functionality:

- access to data
- interactive analysis of the data
- flexible presentation of the data
- access to administrative details (for example, users, groups, application names)

For example, a DataBlox component can populate an HTML list with product categories from an underlying database. In turn, the selection the user makes from that list can drive a query to the underlying database, so a user sees a different set of data when she chooses the category *Compact Cars* than when she chooses *SUVs*.

Blox components have extensive application programmer interface (API) calls that can be accessed using JSP files. The various API calls allow for a large amount of customization to the data being accessed, control over the degree of interactivity to which the user is exposed, and the ability to customize the data presentation to the user.

By assembling Blox components into web-based applications, developers can quickly provide users with immediate access to business information. Using the same Blox components for a spectrum of analysis applications not only reduces application creation, delivery, and maintenance efforts, but also reduces a user's learning curve. Regardless of the data being presented, Blox behavior is consistent across applications and platforms.

Furthermore, the skills required to assemble DB2 Alphablox applications are significantly lower than those required for traditional application development.

Rather than first having to master the intricacies of a programming language, the application developer can quickly begin using HTML, JavaScript, multimedia objects, and Blox components. With these elements, application developers can create appropriate and visually compelling user interfaces for delivering powerful functionality to the user community.

DB2 Alphablox provides the following Blox components from which to assemble applications:

Blox Component	Use
DataBlox	• Provides access to supported multidimensional databases and develops a client representation of a data set. • Provides access to supported relational databases • Accepts and executes query requests • Provides query result sets to Blox components responsible for data presentation
ChartBlox	• Presents a graphical view of multidimensional data • Permits users to manipulate the data in a wide variety of chart formats (including pie, bar, and line) • Enables users to drill down in sequence through the hierarchical data, or pivot the data view
DataLayoutBlox	• Presents grouped lists of available data dimensions and the axes on which they currently reside • Enables users to move dimensions between the page, row, column, and "other" (unused) axes
GridBlox	• Presents multidimensional or relational data in an advanced grid format • Enables users to analyze and manipulate multidimensional data • Enables users to drill down in sequence through the hierarchical data, or pivot the data view
PageBlox	• Presents drop-lists of the dimensions residing on the page axis (thus filtering the data that appears in ChartBlox and GridBlox) • Enables users to change the dimensions and/or members on which to filter data
ToolbarBlox	• Presents buttons for user access to Blox functionality, such as: – toggling between chart and grid presentations – selecting chart types – saving and retrieving application views – moving the toolbar for easier access – opening a Blox in a separate window – pivoting rows and columns • Permits the assembler to select which buttons appear on the toolbar, thus tailoring user access to Blox functionality.
PresentBlox	Combines the functionality of the preceding six Blox components (DataBlox, DataLayoutBlox, ChartBlox, GridBlox, PageBlox, and ToolbarBlox) into a single Blox component, simplifying application assembly and conserving web page real estate
RepositoryBlox	Provides application assemblers with access to stored objects, including saved application views; and server, application, group, and user properties

See the *Developer's Reference* for detailed descriptions of the Blox API available within DB2 Alphablox. See the *Developer's Guide* for guidance on using Blox components to create applications.

DHTML Client

With DB2 Alphablox, application developers can create applications that users can access with a web browser, without additional plug-ins, because it uses standard DHTML technologies. DB2 Alphablox renders the application in the mode in which it is requested, requiring no additional coding by the application developer.

The **DHTML client** is based on the Dynamic HTML technology, utilizing JavaScript and Cascading Style Sheet (CSS) to support the full range of data analysis functionality with a highly usable and customizable graphical user interface. It does not require any plug-ins or download of Java class files. The DHTML client requires the use of supported versions of the Mozilla Firefox and Microsoft Internet Explorer.

Remote Administrative Capabilities

DB2 Alphablox provides comprehensive system administration capabilities through web-based administration pages, a console command window, or a telnet window. Through these administrative interfaces, administrators can perform the following kinds of tasks:

- Create DB2 Alphablox console logs and specify the levels of events and messages to record.
- Monitor DB2 Alphablox and application activity.
- Create DB2 Alphablox objects such as data sources, users, groups, and applications.
- Monitor the use of DB2 Alphablox objects.
- Start and stop DB2 Alphablox services, sessions, and applications.

For more information, see Chapter 4, “Basic administration tasks,” on page 23 and Chapter 13, “Configuring DB2 Alphablox Properties,” on page 71.

Centralized Application Management

DB2 Alphablox provides an Application Manager responsible for centralized application management. The Application Manager maintains and tracks the following:

- A list of available applications.
- A list of active applications.
- Activity within an application.
- Users logged in to applications.
- Blox components used by applications.
- Data sources used by applications.
- An application instance history.

N-Tier Architecture

DB2 Alphablox draws fully on the latest Java technologies to implement a web-based, N-tier architecture. A typical application may use three tiers:

- Data resides on and is retrieved from a database server (relational or multidimensional)
- Application web pages reside in web applications on your application server and served through DB2 Alphablox
- Web page display and user interface presentation occur on a client machine through a web browser

On demand, Blox components passed from the web server to web browsers throughout intranets or over the Internet. This Java-based architecture can eliminate the need to install, configure, and maintain client-side application software.

DB2 Alphablox Applications

To users, an DB2 Alphablox application appears as a collection of web pages to browse like other web sites. These web pages serve as containers for the following application components:

- Standard HTML tags and page elements (such as logos, text, images, icons, video clips, sound clips, and animations) to enhance the user interface
- Blox components necessary to deliver the required application functionality and user interface
- JavaScript or Java scriptlets for extended application and UI logic

For example, a sales analysis application might feature an image map of sales regions. The user clicks on a region to view a chart of data for that region. The user interface permits users to change the chart format, display the data in an interactive grid format, and perform multidimensional operations (such as drilling and pivoting). Depending on the application, the user could also render the Blox output for printing or for export to other applications (such as a spreadsheet). For more information on the types and the components of DB2 Alphablox applications, see Chapter 2, "DB2 Alphablox applications," on page 13.

Application Studio

Application Studio, a companion product of DB2 Alphablox and part of DB2 Alphablox, further reduces the time and effort required to deliver complete, custom analytic applications to users. Application Studio includes the following components:

- DHTML Query Builder, a workbench tool which can be used for developing and testing queries against application data sources and accessing several sample data sources for application prototyping and testing
- Blox Sampler - DHTML, containing working code examples that highlight DB2 Alphablox functionality and help developers understand the Blox APIs.

Application Studio enables rapid application prototyping. Accompanying the Studio is complete documentation that provides step-by-step instructions for configuring templates. For more information about Application Studio and its components, see the online help for the various templates within Application Studio.

DB2 Alphablox architecture

The following DB2 Alphablox components are discussed in this section:

DB2 Alphablox is composed of several discrete services that run independently. In addition, it provides for comprehensive administration, including the following:

- An administrative interface to DB2 Alphablox resources (such as applications, data sources, users, and groups).
- An administrative console that can be accessed by web browsers, telnet, or a command window.
- The ability to start, pause, resume, and stop services, sessions, and applications.
- The ability to monitor DB2 Alphablox and application activity.
- Logs for tracking events and messages.

As part of initial installation, DB2 Alphablox is automatically configured with a default set of properties required for its operation. Subsequently, system administrators can change or add to the default DB2 Alphablox properties. For details, see Chapter 13, "Configuring DB2 Alphablox Properties," on page 71.

Service Manager

The Service Manager starts, controls, and provides access to the other services (including required third-party services). The Service Manager processes all service requests and facilitates distribution of the service components.

Request Manager

The Request Manager controls all requests for application pages and peer services. DB2 Alphablox creates a *peer* for each connection. The peer keeps track of the state of the connection. The Request Manager creates, monitors, and manages the threads for each request, validates session IDs against its list of valid sessions, and creates the appropriate request objects to handle each request.

Session Manager

The Session Manager controls and monitors users' interactions with DB2 Alphablox applications. When a user requests an application, the Session Manager first passes the request to the User Manager for authentication. After the user successfully authenticates, the Session Manager creates a new session. A single user may have multiple simultaneous sessions (multiple browser instances). The Session Manager creates and manages session objects and tracks which applications a user visits.

The Session Manager also terminates dormant sessions (after first saving the current state of each application), and releases session resources. If a user attempts to work in the browser window after its session has expired, a reconnect message appears. If the user requests a reconnection (by clicking the browsers **Refresh** button in Internet Explorer, or **Reload** button in Mozilla Firefox), the Session Manager creates a new session and restores the application to its state at the time the previous session expired.

User Manager

The User Manager in Alphablox controls all users of DB2 Alphablox services and is responsible for authenticating application and administrative users, monitoring their resources, administering user access to data and applications, maintaining the

list of active users, and maintaining information about each user (such as when and for how long a user is attached to a specific application).

In DB2® Alphablox 8.4 and earlier versions, the User Manager is built on top of a personalization engine called Extensible User Manager. The External User Manager is based on Servlet 2.2, handles security, and provides LDAP-based and repository-based user management functionality as well as APIs for custom security for other external repositories. Starting in DB2 Alphablox 8.4.1, DB2 Alphablox supports the Servlet 2.4 specification. The Extensible User Manager is deprecated, replaced by the new Personalization Manager based on Servlet 2.4. This Personalization Manager provides methods to notify DB2 Alphablox when a User or Group object is created, loaded, or deleted. You can also use it to specify and modify properties on a user or group, or to modify users and groups contained in a group. Security is now a separate model. User authentication information is configured in the underlying application server, while DB2 Alphablox provides Java Authentication and Authorization Service (JAAS)-based APIs for user authentication.

The User Manager is aware of and manages the following objects that provide the DB2 Alphablox security scheme:

- A User is either an individual end-user or application administrator.
- A Group is a collection of users and other groups, and provides a convenient mechanism for administrators to manage multiple users as a single unit. Groups are hierarchical.
- Permissions determine what access (no access, read, read/write) a particular user has in a role.
- A role is a list of users and/or groups with a specific set of permissions.

Note: Note the following about permissions and roles: If a user appears more than once in the same role (perhaps as both an individual user and as a member of a group), the user is given the union of all permissions of all entries. For example, if the user has read/write access and the user's group has read access, the user is granted read/write access.

Application Manager

The Application Manager creates an application definition when an assembler or administrator creates or modifies an application from the DB2 Alphablox home page, **Administration** tab, **Applications** page.

The Application Manager also accepts user requests for an application, verifies that users have application access, and creates an application instance for each request. The Application Manager monitors all application instances, enables the administrator to shut down an instance, automatically terminates dormant instances, and maintains the following information:

- A list of available applications
- A list of active applications
- An activity within applications
- Users logged in to applications
- Blox components used by applications

Data Manager

The Data Manager controls access to each data source that an application uses. The Data Manager enables the browsing, querying, and retrieval of data from a variety of relational and multidimensional data sources, as specified in the *Installation Guide* for this release.

Specifically, the Data Manager works with a database adapter for:

- Browsing a collection of preconfigured, named data sources
- Exposing the available cubes within each data source (if available)
- Publishing the compatible query types for a specific data source
- Allowing the transversal of a database's metadata
- Managing database connections for user sessions
- Translating query objects into the underlying native query language
- Executing queries against a database
- Interrogating a result set's data and schema
- Modifying a result set by pivoting, expanding, and drilling
- Creating applications that write user input back to the database for what-if analysis

Repository Manager

The Repository Manager controls the DB2 Alphablox Repository and its contents, which include saved application views and all other application, group, and user properties. The DB2 Alphablox Repository can be stored either in operating system files or in a relational database. For details, see Chapter 16, "Using a Database Repository," on page 109.

File Manager

The File Manager controls temporary files and files used in the DB2 Alphablox administration pages.

Console Manager

The Console Manager provides multiple interfaces (web browser, telnet, and command window) for the system administrator to monitor and control the DB2 Alphablox environment. For more details, see Chapter 19, "DB2 Alphablox Console Commands," on page 127 and "Console" on page 18.

DB2 Alphablox Cube Manager

The DB2 Alphablox Cube Manager provides interfaces for stopping, starting, and performing other administrative tasks for DB2 Alphablox cubes. For information on the DB2 Alphablox Cube Server, see the *DB2 Alphablox Cube Server Administrator's Guide*.

Cluster Manager

The Cluster Manager controls communication between the different nodes of DB2 Alphablox when it is run in a clustered environment. A clustered configuration is used to increase the scalability of your DB2 Alphablox system to support very large numbers of users. For details about setting up DB2 Alphablox to run in a clustered environment, see Chapter 18, "Using Clustered Environments," on page 123.

List of tasks to get started with DB2 Alphablox

The following is a simplified list that shows the basic tasks needed to get started with DB2 Alphablox. For more details about each task, see the book or section listed in the *Cross Reference* column.

Task	Cross Reference
1. Understand the environment in which DB2 Alphablox runs and decide on your configuration details (for example, IBM WebSphere, BEA WebLogic, or Apache Tomcat).	Chapter 1, "DB2 Alphablox overview," on page 1
2. Install and configure DB2 Alphablox.	*Installation Guide*
3. Familiarize yourself with the DB2 Alphablox home pages where you can perform administrative tasks such as creating applications, creating users, and setting up your security policy.	Chapter 3, "DB2 Alphablox Home Page," on page 17
4. Understand the database environment in which your data resides.	Your database environment and your knowledge workers
5. Understand the development environment for building DB2 Alphablox applications.	Chapter 2, "DB2 Alphablox applications," on page 13 and the *Developer's Guide*.
6. Create an application definition.	Chapter 6, "Application Definitions," on page 29
7. Develop and test the application using JSP technology, Blox custom tag libraries, and DB2 Alphablox APIs.	See the *Developer's Guide* and *Developer's Reference*
8. Deploy the application to your enterprise.	Your users, your corporate intranet, and the Internet.

Chapter 2. DB2 Alphablox applications

You can create DB2 Alphablox applications to provide access to data for analysis through a browser, whether in a corporate intranet environment or in the broader internet. Because DB2 Alphablox applications are highly customizable and can integrate into many types of enterprises, there are a wide variety of applications that can be called "DB2 Alphablox applications." This chapter briefly describes those types of applications and describes the components that make up an DB2 Alphablox application. For more information about DB2 Alphablox applications, see the *Developer's Guide*.

Types of DB2 Alphablox applications

DB2 Alphablox applications are J2EE-compliant web applications that run in the DB2 Alphablox environment. Because DB2 Alphablox applications run in a web browser and can be combined with virtually any other web application technologies, there can be a very large variety to what these application do and how they look. At the most basic level, DB2 Alphablox applications provide a way to interactively display analytic data in a web page so users can use the data to gain valuable knowledge about the business underlying the data. The data resides in various databases throughout an enterprise.

For the DB2 Alphablox components in these web applications, this section briefly describes the following categories of DB2 Alphablox applications:

- "Data Presentation Modes"
- "Relational Reporting Applications" on page 14

You can also combine several different DB2 Alphablox elements in a single application. For example, you can display a chart and a spreadsheet in the same application. For more details about what is possible with and how to develop these applications, see the *Developer's Guide* and the *Relational Reporting Developer's Guide*.

Data Presentation Modes

There are many ways that data can be presented in DB2 Alphablox applications. You can show data in a grid, in a chart, or both. You can also use data from a database to populate elements of your applications such as drop down lists, menus, etc.

The DB2 Alphablox DHTML client uses the latest Dynamic HTML technology, including HTML, JavaScript, and Cascading Style Sheets (CSS), to support a full range of data analysis functionality with a highly usable and customizable graphical user interface, including right-click context menus. A major advantage of the DHTML client is the extensibility available using the Blox APIs with the Blox UI Model. The DHTML client mode can be used only with the more recent Microsoft Internet Explorer browsers, but does not require additional web browser plug-ins, client machine installations, or the download of Java class files.

Convert to PDF

You can create DB2 Alphablox applications that allow you to print your reports to PDF files, creating very high quality output. The Print to PDF feature uses a PDF template which can contain any elements you require (for example, company logos, boilerplate text, etc.) and then take your live data and create PDF file

containing a top quality report. You can then print, e-mail, or post on a web site the PDF file for distribution. PDF functionality can be used with all of the client rendering modes

Custom Render Modes Using XML

DB2 Alphablox provides APIs to access the data in XML format, allowing developers to create custom layouts of the data in any way needed, or to manipulate the data using whatever custom techniques you want. This is useful if you have some specialized need that the other presentation methods do not address.

Relational Reporting Applications

You can use ReportBlox components to create DB2 Alphablox applications that provide reports sourced from relational databases. These reports are highly customizable and are displayed in HTML tables, using Cascading Style Sheets (CSS) technology to provide easy customization and a flexible look-and-feel.

Reports created using ReportBlox components can be used in an interactive mode to provide a user interface to customize them, as well as a comprehensive API to programmatically change them. For details about ReportBlox components, see the *Relational Reporting Developer's Guide*.

Components of an DB2 Alphablox application

Because DB2 Alphablox applications are J2EE applications, they can have any components available to J2EE applications. This section describes the following basic components of DB2 Alphablox applications:

- "JavaServer Pages (JSP)"
- "Blox Components"

For a description of the directory structure of DB2 Alphablox applications, see "DB2 Alphablox Application Definitions" on page 29.

JavaServer Pages (JSP)

JavaServer Pages (JSP) technology is the mechanism from which an application developer typically provides DB2 Alphablox functionality to an application. Interactive web pages are developed using JSP, Blox custom tag libraries, HTML, and any Java or JavaScript code as needed for your particular applications. JSP pages are compiled at runtime on the application server (for performance reasons, the application server only compiles the page the first time it is accessed; subsequent page requests use the precompiled page in most application server configurations), then DB2 Alphablox processes any DB2 Alphablox requests, and then the web server returns dynamic content to the user.

Blox Components

Blox components are DB2 Alphablox components that provide functionality such as accessing data sources, displaying data in grids and charts, etc. You add Blox components to a JSP page using the Blox custom tag libraries. The Blox tag libraries allow you to control many aspects of the Blox. For example, you can change an attribute of ChartBlox to specify the type of chart displayed.

You can also manipulate Blox components programmatically, using either Java or JavaScript. When you use JavaScript to access Blox APIs, the code is interpreted on

the browser. When you use Java to access Blox APIs, the code is compiled by the JSP engine (the application server) and then executed on the server.

For more information about the types of things Blox are used for, see "DHTML Client" on page 6. For details about using Blox components in DB2 Alphablox applications and the types of components available when using the DHTML client, see the *Developer's Guide*, the *Developer's Reference*.

Chapter 3. DB2 Alphablox Home Page

This chapter describes the DB2 Alphablox home page. For information about the command line interface, see Chapter 19, "DB2 Alphablox Console Commands," on page 127.

DB2 Alphablox Home Page overview

DB2 Alphablox is automatically configured during initial installation. Following installation, administrators can set or change the DB2 Alphablox startup configuration, properties, and ports. To administer DB2 Alphablox, use the **Administration** tab on the DB2 Alphablox home page or the Console command line interface. To access the DB2 Alphablox administration pages, enter one of the following URL addresses in your browser window:

```
http://<servername>/AlphabloxAdmin/home/
http://<servername>/AlphabloxAdmin/
http://<servername>/AlphabloxAdmin (except iPlanet)
```

where `<servername>` represents the name of the server and port number on which DB2 Alphablox runs.

Applications tab

When you open to the DB2 Alphablox home page, the Application tab is selected by default.

The **Applications** tab displays by default when a user accesses the DB2 Alphablox home page. The **Applications** tab lists the applications and any available saved application states to which the user has access. Some users might only see the **Applications** tab because they do not have the necessary privileges to view or access the other tabs.

To launch an application, click the desired link. The applications launch in DHTML mode by default. Selecting the **Default Application State** from the drop list loads the application in the state defined as the default state in the application definition.

Other Applications

Any other applications that are defined in DB2 Alphablox also appear under the **Applications** tab. To launch an application, click its link. The applications launch in DHTML mode by default. Selecting the **Default Application State** from the drop list loads the application in the state defined as the default state in the application definition.

Administration tab

The **Administration** tab appears on the DB2 Alphablox home page only if the authenticated user has administrative privileges. Clicking the tab opens the **General** page where administrators can define or modify aspects of DB2 Alphablox. The following sections are under the **Administration** tab:

- General
- "Groups" on page 19

General

The **General** link under the **Administration** tab provides the interface for administrators to configure core DB2 Alphablox functions.

General Properties

The **General Properties** section is where you configure the parameters for starting up DB2 Alphablox and setting the maximum number of DB2 Alphablox cubes allowed for the DB2 Alphablox Cube Server. For details on performing these tasks, see "DB2 Alphablox Administration Tasks" on page 71. For information about DB2 Alphablox Cube Server, see the *DB2 Alphablox Cube Server Administrator's Guide*.

Custom Properties

For details on defining custom properties, see "Custom Property Definitions" on page 75.

Runtime Management

Use the **Runtime Management** section to manage your DB2 Alphablox license, application sessions, comments collections, PDF reports (DHTML only), and DB2 Alphablox cubes. For information on creating and maintaining DB2 Alphablox cubes, see the *DB2 Alphablox Cube Server Administrator's Guide*.

Console

Clicking the **Start Console Session** link opens the console window, where administrators can enter textual commands to perform most of the administrative functions that are normally performed through the user interface. For information on using the Console, see Chapter 19, "DB2 Alphablox Console Commands," on page 127.

Note: The telnet console user (as defined in the DB2 Alphablox home page, **Administration** tab, **General** link, **Telnet Console** link) can also access the Console through any telnet terminal software. You can access telnet software on most systems running a Microsoft Windows® operating system from the Windows **Start** menu, **All Programs** folder, **Accessories** group, **Telnet** shortcut, or open a telnet window from the **Start** menu, **Run** shortcut, then type **telnet**.

Portal Theme Utility

Note: The Portal Theme Utility option appears in the menu only when DB2 Alphablox is installed on a WebSphere Portal Server.

Click the **Portal Theme Utility** link to create portal themes for use in WebSphere portal applications containing DB2 Alphablox Blox components. Using this utility, a new DB2 Alphablox portal theme is created by merging style properties from the specified WebSphere Portal Server theme with style properties from the specified DB2 Alphablox theme. The new theme can be used to optimally display Blox components (for example, a GridBlox or ChartBlox) in portal applications.

An optional portlet version of the Portal Theme Utility is available for use in portal applications. This portlet can be found in the `AlphabloxAdminPortlets.war` file located in the `installableApps` directory of your DB2 Alphablox installation.

For help using the Portal Theme Utility, see the online help available by pressing the Help button in this utility. On the portlet version of the Portal Theme Utility, selecting the help mode opens the online help.

Application Migration

Note: The Application Migration Utility option appears in the menu only when DB2 Alphablox is installed on an Apache Tomcat server.

Click the **Application Migration** link to access the Alphablox Application Migration Utility for Tomcat 5.5. This utility assists you in migrating selected DB2 Alphablox applications from Apache Tomcat 3.2.4 to Apache 5.5. For each application that is migrated, the utility will:

- Copy the application (if needed) to the *webapps* directory of the current Alphablox installation.
- Deploy the application to Apache Tomcat 5.5
- Import the application to Alphablox (if needed)
- Update the Alphablox tag library descriptor (TLD) files for the current release
- Update the application's *web.xml* file to the Servlet 2.4 specification

Migrated applications should run properly on Apache Tomcat 5.5 installations with DB2 Alphablox.

Groups

Use the **Groups** link under the **Administration** tab to define groups, assign users to groups, and so on. For details on setting up groups, see Chapter 9, “Group Definitions,” on page 51.

Users

Use the **Users** link under the **Administration** tab to define users, reset passwords, assign group memberships to users, and so on. For details on setting up users, see Chapter 8, “User Definitions,” on page 47.

Roles

Use the **Roles** link under the **Administration** tab to create and modify roles. Roles allow you to control which users and/or groups have access to a particular application. For details on setting up Roles, see Chapter 10, “Role Definitions,” on page 55.

Applications

Use the **Applications** link under the **Administration** tab to create and modify the state of applications that reside on the DB2 Alphablox. You can define default states, set roles, and define an image that appears on the Applications tab. For details on the tasks necessary to define an application, see Chapter 6, “Application Definitions,” on page 29.

Data Sources

Use the **Data Sources** link under the **Administration** tab to define and modify data sources that are available to DB2 Alphablox applications running on DB2 Alphablox. You can define data sources to access OLAP databases (for example, IBM DB2 OLAP Server™, Hyperion Essbase, and Microsoft Analysis Services), relational databases (for example, IBM DB2 UDB, Oracle, Sybase, and Microsoft SQL Server), and DB2 Alphablox cubes. For details on defining and modifying a data source, see Chapter 7, "Data Source Definitions," on page 39.

DB2 Alphablox cubes

Use the **Cubes** link under the **Administration** tab to define or modify DB2 Alphablox cubes. DB2 Alphablox cubes present data stored in a relational database in a multidimensional form. For more information on DB2 Alphablox cubes and the DB2 Alphablox Cube Server, see the *DB2 Alphablox Cube Server Administrator's Guide*.

Assembly tab: Application Studio

Clicking the **Assembly** tab on the DB2 Alphablox home page opens the DB2 Alphablox Application Studio. The opening Application Studio page has links to pages containing examples and tools. These pages are designed to help developers rapidly create DB2 Alphablox applications. Application Studio is a framework of easy-to-use workbench tools and working code examples for easy application creation. For more detailed information on Application Studio, see the *Developer's Guide*.

Workbench

The **Workbench** page under the **Assembly** tab contains a link to **Query Builder**. Query Builder is an application that allows developers to interactively enter queries to various data sources and view the results in a grid and/or a chart.

Examples

The **Examples** page under the **Assembly** tab contains links to example JSP applications, code samples, and other examples. Included in the examples is the *Blox Sampler*, which contains many working code examples. These code samples are extremely helpful in learning to develop applications with DB2 Alphablox, and many of them are suitable to cut and paste directly into your own custom applications.

Sample Data

Sample data for use in the *Blox Sampler* application ships with DB2 Alphablox. The sample data is a database named the Quality Chocolate Company (QCC), and there are versions of the database available for multiple databases, both multidimensional and relational. For details on installing the sample data, see "Loading the Sample Data" in the *Installation Guide*.

Links in the upper right corner

Each tab of the DB2 Alphablox home page has two links in the upper right corner of the page:

- "My Profile" on page 21
- "Help" on page 21

My Profile

Clicking the **My Profile** link opens the **Profile** page. Users can change their profile description, passwords, and e-mail address, as well as install or uninstall local class (JAR) files from this page.

Help

The **Help** menu provides links to the following:

- **Help On** link provides context-sensitive online help for the page in which the button is clicked. The online help provides information about how to use the interface for each page.
- **Online Documentation** link opens the DB2 Alphablox online documentation in a new browser window. The online documentation contains web-based (HTML) and PDF versions of the DB2 Alphablox documentation set.
- **Server-Side API Reference** link opens the Javadoc for the server-side Java APIs.
- **About DB2 Alphablox** link provides copyright information about DB2 Alphablox.

Note: Users who are not members of the `AlphabloxAdministrator` role do not see all of these links.

Chapter 4. Basic administration tasks

DB2 Alphablox includes comprehensive facilities for managing the server environment. The DB2 Alphablox home page provides a single point to administer all aspects of the server. This section describes the steps for starting, accessing, configuring, and stopping DB2 Alphablox.

Starting DB2 Alphablox

When you are running DB2 Alphablox with a commercial application server like IBM WebSphere or BEA WebLogic, DB2 Alphablox is integrated with the application server. When you start your application server, DB2 Alphablox will automatically be started. For IBM WebSphere Portal Server, DB2 Alphablox is started when you start the portal server. For information about starting WebLogic or WebSphere, see the documentation for those products.

Accessing DB2 Alphablox

DB2 Alphablox is administered through the **Administration** tab of the DB2 Alphablox home page.

To access DB2 Alphablox through a browser, enter the following URL:

```
http://<serverName>/AlphabloxAdmin/home/
```

where `<serverName>` represents the name of the server and port number on which DB2 Alphablox runs.

To access the **Administration** tab of the DB2 Alphablox home page from the Windows **Start** menu:

1. Open the Windows **Start** menu.
2. Select the **All Programs** folder.
3. Select the **DB2 Alphablox** program group.
4. Select the instance name for the instance of DB2 Alphablox (the default value is **AlphabloxAnalytics**).
5. Select the **DB2 Alphablox Home Page** shortcut.

Stopping DB2 Alphablox

On WebSphere application servers, you can stop DB2 Alphablox by stopping the application server. For WebSphere Portal Server, stopping the portal server stops DB2 Alphablox. On Apache Tomcat installations, you can stop DB2 Alphablox either using the shutdown script if it is running in a console window or, if it is running as a Windows service, from the services Control Panel.

Stopping DB2 Alphablox with the shutdown script on Apache Tomcat Installations

To stop DB2 Alphablox when DB2 Alphablox is running in a Console window (on Windows: the window that opens when you start DB2 Alphablox from the Windows **Start** menu, **All Programs** folder, **DB2 Alphablox** group, instance name

(the default value is **AlphabloxAnalytics**), **Shutdown DB2 Alphablox** shortcut; on Sun Solaris: the window in which you started the server), perform the following steps:

On Windows Systems:

- Run the **Start** menu, **All Programs** folder, **DB2 Alphablox** group, instance name (the default value is **AlphabloxAnalytics**), **Shutdown DB2 Alphablox** shortcut.

On Linux and UNIX Systems:

- Run the following command from the `<db2alphablox_dir>/bin` directory, where `<db2alphablox_dir>` is the directory in which DB2 Alphablox is installed:

  ```
  ./StopAlphablox.sh
  ```

Stopping DB2 Alphablox from the Services Control Panel on Apache Tomcat installations

If you are running DB2 Alphablox as a Windows service, perform the following steps to stop DB2 Alphablox:

1. Open the Windows Control Panel from the Windows **Start Menu**, **Settings**, **Control Panel** shortcut.
2. Double click the **Services** control panel document. The Services window appears.
3. For each running DB2 Alphablox service, select the service and click the **Stop** button.
4. Press the **Yes** button to confirm that you want to stop the service.
5. Click the **Close** button to close the Services window.

Using the Essbase Client Library Utility

To use DB2 Alphablox with DB2 OLAP Server or Hyperion Essbase data sources, you must install the appropriate Essbase client library files and make them available to DB2 Alphablox. Essbase client libraries are available from Hyperion (http://support.hyperion.com/).

To change or update the Essbase client libraries installed on DB2 Alphablox:

1. On DB2 Alphablox, open the following directory:

   db2alpahblox_dir/bin/ , where `db2alpahblox_dir` is the location of your DB2 Alphablox installation (for example, *c:\alphablox\analytics*).
2. Find the ChangeEssbase batch file (*ChangeEssbase.bat* on Windows or *ChangeEssbase.sh*on Linux® and UNIX®) and then run it. A script console window appears.
3. Type in the fully-qualified path for your installation of DB2 Alphablox (for example, *c:\alphablox\analytics*) and then press **Enter**.
4. A list of available class library options displays. Choose the appropriate option, depending on the Essbase version of the client libraries you want to use, then press **Enter**.
5. If the operation is successful, you will see a message stating that your system has been reconfigured.
6. Restart DB2 Alphablox for the changes to take effect.

Other Administrative Tasks and Information

For information on other administrative tasks, see the following sections:

- Chapter 1, “DB2 Alphablox overview,” on page 1
- Chapter 19, “DB2 Alphablox Console Commands,” on page 127
- Chapter 3, “DB2 Alphablox Home Page,” on page 17
- Chapter 6, “Application Definitions,” on page 29
- “Custom Property Definitions” on page 75
- Chapter 10, “Role Definitions,” on page 55
- Chapter 7, “Data Source Definitions,” on page 39
- Chapter 8, “User Definitions,” on page 47
- Chapter 9, “Group Definitions,” on page 51

Chapter 5. Client Administration

Administrators need to be aware of some issues related to the use of the DHTML client with DB2 Alphablox applications.

DHTML Client Administration

When using the DHTML rendering mode with DB2 Alphablox, analytic applications are displayed using the built-in capabilities of the supported Mozilla Firefox and Microsoft Internet Explorer web browsers. As a result, minimal administration is required to support DHTML-based applications.

Supported DHTML Client Configurations

See the System Requirements section of the *Installation Guide* for information about supported DHTML client configurations of Mozilla Firefox and Microsoft Internet Explorer web browsers.

DHTML Client Issues

In general, there are few issues your users are likely to encounter using Mozilla Firefox and Microsoft Internet Explorer with DB2 Alphablox applications. Below are some potential issues that you may want to be aware of.

CSS

On rare occasions, users set personal preferences for Cascading Style Sheets (CSS) settings under the advanced options available in Mozilla Firefox and Microsoft Internet Explorer web browsers. Due to the cascading nature of CSS and the extensive use of CSS in the DHTML client, it is possible that changes in these settings could affect the appearance of Blox components in these browsers.

Pop-up windows

When users click on buttons or links when using the DHTML client, expected pop-up windows displaying either further information or dialogs may fail to appear. Pop-up window blocking functionality built into web browsers might interfere with the display of pop-up windows. Also, optional browser tools (for example, the Google Toolbar) and firewall software (for example, Zone Alarm) may include settings affecting the behavior of web browser pop-up windows. To correct this problem, the user will either have to disable this functionality or, by changing their user preferences, can authorize pop-up windows on DB2 Alphablox applications.

Chapter 6. Application Definitions

This chapter describes the J2EE application structure and provides procedures for creating, modifying, and deleting application definitions in DB2 Alphablox.

DB2 Alphablox Application Definitions

A list of DB2 Alphablox applications appears on the **Applications** page under the **Administration** tab. To define a new DB2 Alphablox application, click the **Create** button. To modify or delete an application definition, select an application from the list and press the **Edit** or **Delete** button.

When you create an application using the DB2 Alphablox Admin pages, DB2 Alphablox creates an application definition in the DB2 Alphablox Repository. The application name in the DB2 Alphablox Repository consists of two parts, the enterprise application name and a unique identifier (URI). For example, if you create an application called MyApp, the application in the repository will be named `MyApp_MyApp`. For WebSphere and WebLogic application servers, DB2 Alphablox creates installable applications which must be imported into the application server.

Below is the directory structure of a typical DB2 Alphablox application imported into a WebSphere application server:

```
<application>.ear
  META-INF/
    application.xml
    MANIFEST.MF
  <application>.war
    META-INF/
      MANIFEST.MF
    WEB-INF
      timeschema.dtd
      timeschema.xml
      web.xml
      classes/
      lib/ (for Alphablox 8.4.1 applications only)
         aastaglibs.jar
      tlds/ (for Alphablox 8.4 applications only)
        blox.tld
        bloxform.tld
        bloxlogic.tld
        bloxui.tld
```

Application files, including JSP files, JavaScript files, HTML files, CSS files, and image files, are typically placed in the `<application>.war` directory. Depending on your development practice, some files may be placed in nested folders within the `<application>.war` directory. Custom Java classes, including JavaBeans components, are placed in the `classes` directory. Tag library descriptor (TLD) files required for DB2 Alphablox applications are in the *tlds* directory in DB2 Alphablox 8.4 applications; in DB2 Alphablox 8.4.1 applications, the TLD files are packaged in the *aastaglibs.jar*, located in the application's*lib* directory. Time schema-related files (`timeschema.dtd` and `timeschema.xml`), required for some specialized applications, are located directly within the *WEB-INF* directory. When DB2 Alphablox creates an installable application, the deployment descriptor file (`web.xml`) is also modified as required for DB2 Alphablox applications.

Application Names

The application name that is used in the DB2 Alphablox Repository consists of both the enterprise application name plus a unique identifier. After you create an application, you can go to the edit view of a particular application to view naming information for your application. This listing includes the DB2 Alphablox name, the J2EE application name, the module name, context path, and display name.

WEB-INF Directory

Each web application contains a `WEB-INF` directory. This directory, created automatically during application creation, contains metadata about the application. During normal operations, you should not need to modify anything under the `WEB-INF` directory. During the creation of an application, the following files are installed in the `WEB-INF` directory:

- `web.xml`
- `aastaglibs.jar` (in the *lib* directory) in DB2 Alphablox 8.4.1 or as individual TLD files (for example, *blox.tld*) in DB2 Alphablox 8.4)
- `timeschema.dtd`
- `timeschema.xml`

The application's deployment descriptor file (`web.xml`) is a standard XML file in all J2EE applications and contains markup describing the attributes of the application. The application server reads the `web.xml` file during initialization of the application.

Important: Modifications to the `web.xml` file are automatically performed by DB2 Alphablox during product upgrades; use caution if you must modify the `web.xml` file because incorrect entries can cause unexpected behavior in the application.

The Alphablox tag library descriptor (TLD) files, packaged in the *aastaglibs.jar* in Alphablox 8.4.1 (or as loose TLD files in Alphablox 8.4), give developers access to the DB2 Alphablox tag libraries used to create Blox components and DB2 Alphablox applications. Two time-related files (`timeschema.dtd` and `timeschema.xml`), stored directly within the `WEB-INF` directory, are available to developers for use in creating specialized applications dealing with fiscal and calendar time periods. For more information on these files, see the *Developer's Guide* and the *Developer's Reference*

Defining a New Application

Perform the following to define a new application:

Note: If you are using WebSphere as your application server, additional steps are required to create an application. For details, see "Defining an Application When Running With WebSphere" on page 31.

1. Log into the DB2 Alphablox home page as the *admin* user or as another user who is a member of the administrators group.
2. Click the **Administration** tab.
3. Click the **Applications** link.
4. Click the **Create** button. The **Create Application page** appears.
5. Enter a unique **Name** for your application. Allowable characters are A-Z, a-z, 0-9, underscore, and special characters (for example, accented characters) when running in a language other than English. The display of the name is

case-sensitive, but the actual name that is authenticated is case-insensitive. The names *Public*, *Private*, and *Properties* are reserved and cannot be used for an object name.

6. [Optional] Type a brief description of your application in the**Description** field.
7. [Optional] Type a URL address, relative to the application name, that identifies where the initial page of this application resides (**Home URL**), and another relative URL pointing to an identifying image for this application (**Image URL**). These two values are used to create an entry under the **Applications** tab, from which users can launch the application.
8. In the **Default Saved State** field, type the parameter name specifying the desired saved state. The parameter name is created by the application developer if they desire their users to access a saved state version of the application. If a Default Saved State is not specified, the restored state is the state that the application was in upon browser shutdown or timeout.
9. For the **Write Privileges Security Role** text box, type the name of an existing role for access to this application. If you leave this blank, the default access privileges are applied.
10. Enter a **Session Timeout** to set the J2EE session timeout, which is the idle time before a user's session times out. The timeout value is in minutes.
11. Select **Yes** or **No** from the **Restore Saved Application State Enabled** drop list to indicate whether or not to load the latest autosaved state.
12. The **Include on Applications Page** option determines whether users will see this application on the their view of the Applications page. The default value is **Yes**.
13. [Optional] Set the **Default Render Mode** for your application. By default, a new application supports the DHTML render mode and installs copies of the Alphablox tag library descriptor (TLD) files into the new application's directory.
14. [Optional] Use the **Header Links** feature to set up links from the members displayed on a grid to open a URL or to perform some JavaScript. Add unique member names and URL addresses or other valid URL text in the **Header Links** text box, inserting a line break between each entry with the following syntax:

    ```
    Member = URL
    ```

    For example, to create a link to `http://www.ibm.com` from a member named IBM, enter the following into the Header Links text box:

    ```
    IBM = http://www.ibm.com
    ```
15. Press the **Save** button to define the new application and return to the Applications page.
16. If you are running an Apache Tomcat configuration with an external web server (Apache HTTP Server, Microsoft IIS, or Sun iPlanet), you must perform additional steps before the application is available. For details, see "Registering Applications with External Web Servers" on page 35.

Defining an Application When Running With WebSphere

If you are using IBM WebSphere Application Server, an application must be defined in WebSphere either before or after defining the application in DB2 Alphablox. This section describes the basic steps needed to create a web application in WebSphere. For details about creating applications using WebSphere, see your WebSphere documentation.

There are two ways to define an DB2 Alphablox application when running with WebSphere:

- "Creating an application in DB2 Alphablox, then registering in WebSphere"
- "Importing an existing WebSphere application into DB2 Alphablox" on page 33

Either way is acceptable; choose the one that is most convenient for you.

Creating an application in DB2 Alphablox, then registering in WebSphere

If you create an application in DB2 Alphablox when you are running with WebSphere, DB2 Alphablox creates the application structure in an enterprise archive (EAR) file. The EAR file is created in the following directory:

`<db2alphablox_dir>/installableApps/`

The name of the file is `<application>.ear`, where `<application>` is the name specified for the application when creating the application.

Before you can use a newly created application, you must perform the following steps to create the application structure in DB2 Alphablox and then register it with WebSphere:

1. In DB2 Alphablox, first define a new application as described in "Defining a New Application" on page 30. An EAR file is created in the `<db2alphablox_dir>/installableApps` directory.
2. Open the WebSphere Administrative Console, then select **Applications > Install New Application**.
3. On the **Preparing for the application installation** screen, click the Browse button and choose the following path:

   `<db2alphablox_dir>/installableApps/<newApplication>.ear`

   where `<application>` is the name of the application you want to install, then press Next.

   Note: Clustered environments only: You might need to use the server path setting to browse through the network to locate the `<newApplication>.ear` file.
4. The next screen shows the **Default Bindings Options**. Leave the default settings, unless you require different bindings, then click Next.
5. The **Application Security Warning** screen appears. Scroll to the bottom of this page and click on the Continue button.
6. The next section, **Install New Application**, consists of the following five steps as displayed in the dialog:

   Step 1: Provide options to perform the installation

   You can leave the existing settings and click Next.

   Step 2: Map virtual hosts for web modules

   Accept the existing values and click Next.

   Step 3: Map modules to application servers

   Accept the existing values and click Next.

   Step 4: Map security roles to users/groups

   Two roles, `AlphabloxAdministrator` and `AlphabloxUser`, should appear. You need to add at least one user for each role.

For the AlphabloxAdministrator role, check the check box in front of that role, then press the **Lookup Users** or **Lookup Groups** buttons to add administrative users. You must select at least one user. After you have added the users or groups, they should be listed in the Mapped Users or Mapped Groups value for this role.

For the AlphabloxUser role, check the check box under the All Authenticated column for this role. This allows all authenticated users to access applications. You may also use these settings to restrict access to specific users or groups, depending on your application requirements.

When finished, click Next.

Step 5: Summary

Scroll to the bottom of this screen, then press the Finish button.

7. The application is then installed and "Application <application> installed successfully" will be displayed. Click on the **Save to Master Configuration** link.
8. The **Save to Master Configuration** dialog should appear. Click on the Save button. After a short wait period, you will be returned to the Administrative Console home page.
9. In the WebSphere Administrative Console, open **Applications > Enterprise Applications** and restart your application.

You can now add content and use the application, as well as modify any of its properties from the DB2 Alphablox home pages.

Importing an existing WebSphere application into DB2 Alphablox

To import an existing WebSphere application into DB2 Alphablox, adding required DB2 Alphablox settings and files, perform steps in "Importing an Existing J2EE Application" on page 37.

Modifying an Existing Application Definition

This section describes how to modify an existing application. Perform the following to modify an existing application:

1. Log into the DB2 Alphablox home page as the admin user or as a user who is a member of the administrators group.
2. Click the **Administration** tab.
3. Click the **Applications** link.
4. Select the application you want to change from the list of applications and click the **Edit** button. The **Edit Application** page opens.
5. Change the appropriate property values.

Note: WebSphere and WebLogic configurations: Applications can be renamed by modifying the **Name** field, but changing the application name will change the application name only in the DB2 Alphablox Repository and cannot modify the associated WebSphere or WebLogic application name or context path. To modify the associated application properties in WebSphere or WebLogic, use your application server's administration console to rename the underlying application context path to use the new name and rename the underlying application document base directory.

Note: Apache Tomcat configurations: Applications can be renamed by modifying the **Name** field. When you rename an application, the directory name for the application in the DB2 Alphablox Repository and any bookmarks that reference the application are renamed.

6. Click the **Save** button to change the application's property values and return to the **Applications** page.
7. If you are running an Apache Tomcat configuration with an external web server (Apache HTTP Server, Microsoft IIS, or Sun iPlanet) and you rename the application, you must perform additional steps necessary for your configuration. For details, see "Registering Applications with External Web Servers" on page 35.

Deleting an Existing Application Definition

Perform the following to delete an existing application:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab.
3. Click the **Applications** link.
4. Select the application you want to delete from the list of applications and click the **Delete** button.

   Important: Deleting the application removes the application definition from DB2 Alphablox, and thus the application name no longer appears in the application listings. The application folder and its associated files, however, are not removed and must be manually deleted.

   Important: Deleting DB2 Alphablox applications on WebLogic: If `autoDeploy` is enabled, to fully delete an DB2 Alphablox application on the WebLogic server, follow these steps:

   a. Use WebLogic admin console to disable the autoDeploy feature.
   b. Delete the application definition from DB2 Alphablox. DB2 Alphablox will unregister the application on the WebLogic server.
   c. Delete the application folder.
   d. If desired, enable the autoDeploy feature again.

   If autoDeploy is not enabled, only steps b and c are required.
5. If you are running an Apache Tomcat configuration with an external web server (Apache HTTP Server, Microsoft IIS, or Sun iPlanet), you must perform the steps necessary for your configuration after deleting the application. For details, see "Registering Applications with External Web Servers" on page 35.

Defining an Application When Using WebLogic Clusters

If you are using DB2 Alphablox in a BEA WebLogic clustering environments, follow these steps to create new applications:

1. Define an application on DB2 Alphablox from the Administration tab under applications as described in "Defining a New Application" on page 30.
2. Copy the entire directory of the newly created application to the WebLogic administration server. A good location would be the directory under the domain at the same level as the applications directory.
3. Add all necessary JSP files into the applications directory at the same level as the *WEB-INF* directory.

4. From the WebLogic console, configure the newly added web application. On the WebLogic server, you should be able to view the new applications directory, then press **Select**, or navigate to the proper level.
5. Select your cluster as the deployment target, then press the **Configure and Deploy** button.
6. After the application has deployed, select AlphabloxServer as the Web Application.
7. Select the Deploy tab and redeploy the AlphabloxServer application.
8. Redeploying will stop and restart DB2 Alphablox.
9. When DB2 Alphablox is restarted, you can access your applications.

Note: You do not need to copy your JSP files to each of the Managed servers since registering your web application on WebLogic takes care of this for you. For more information on this or to modify this refer to WebLogic documentation.

Registering Applications with External Web Servers

If you are running an Apache Tomcat configuration of DB2 Alphablox with an external web server, there are some steps necessary each time you create or delete an application.

Because the web server is designed to efficiently serve static pages and the application server is designed to serve dynamic pages, these steps are necessary to ensure the best performance of both the web server and the application server. Without these steps, the application server would be forced to unnecessarily process static pages.

The following configurations are described:

- "Registering applications on Apache HTTP Server" on page 35
- "Registering applications on Microsoft Internet Information Server (IIS) Web servers" on page 36
- "Registering applications on Sun iPlanet Web Servers on Alphablox 8.4" on page 36

Registering applications on Apache HTTP Server

DB2 Alphablox 8.4.1 on Apache Tomcat 5.5

When you create or delete DB2 Alphablox application definitions with an Apache Tomcat configuration, the changes will not be reflected in the Apache HTTP Server until both servers have been properly stopped and restarted. When DB2 Alphablox is restarted, it writes a new Apache configuration file (`mod_jk.conf`) to the Apache HTTP Server.

To register DB2 Alphablox application additions or deletions:

1. Shut down both the Apache HTTP Server and the Apache Tomcat server (running DB2 Alphablox)
2. Restart the Apache Tomcat server.
3. Restart the Apache HTTP Server.

After restarting both servers, the application changes will be recognized on the Apache HTTP Server.

DB2 Alphablox 8.4 on Apache Tomcat 3.2.4

If you are running an Apache Tomcat configuration using the Apache HTTP Server, you must restart the Apache web server each time you create or delete an DB2 Alphablox application definition. The restart is necessary so Apache can read the changes DB2 Alphablox makes to the Apache configuration file (`mod_jk.conf-alphablox`) when you create or delete an DB2 Alphablox application definition.

Stopping and restarting Apache HTTP Servers

To stop Apache, run the shutdown command if it is running in a console window or, if it is running as a Windows service, select the service from the **Services** control panel document and click the **Start** button.

To restart Apache, run the startup command if it is running in a console window or, if it is running as a Windows service, select the service from the **Services** control panel document and click the **Stop** button.

For more details on stopping and starting the Apache web server, see your Apache documentation.

Registering applications on Microsoft Internet Information Server (IIS) Web servers

If you are running a standalone configuration using the IIS web server, there are two tasks you must perform each time you create or delete an DB2 Alphablox application definition:

- You must add or delete the virtual directory for the DB2 Alphablox application.
- You must restart the web server.

Registering applications on Sun iPlanet Web Servers on Alphablox 8.4

If you are running a standalone configuration using the Sun iPlanet web server on DB2 Alphablox 8.4 (not 8.4.1), there are two tasks you must perform each time you create or delete an DB2 Alphablox application definition:

- "Add the Document Directory"
- "Add the Appropriate Assignments to the DB2 Alphablox Style" on page 37

Add the Document Directory

For each DB2 Alphablox application, you must add the document directory to the web server so it can serve up pages from this directory. The following steps describe how to add the application directory to your iPlanet web server.

1. Open the iPlanet Web Server Administrative pages. By default, these pages are accessible via port 8888 on the server in which iPlanet is running, but the port might be different on your system.
2. Select your server from the drop down list and click the **Manage** button.
3. Click the **Class Manager** link in the upper right hand corner.
4. Click the **Content Mgmt** tab.
5. Click the **Additional Document Directories** link.
6. In the **URL prefix** text box, enter the context name of your application followed with a forward slash (/) as in the following example:

   ```
   MyApplication/
   ```

Important: Do not put the forward slash in front of the context name or it might corrupt the iPlanet configuration file.

7. In the **Map to Directory** text box, enter the directory of your application as in the following example:

   ```
   d:/Alphablox4/webapps/MyApplication
   ```

   Tip: Use only forward slashes (/) for your path name; some versions of iPlanet do not handle backward slashes (\) correctly in path names.
8. Choose **NONE** from the **Apply Style** drop down list.
9. Click the **OK** button and then save your changes.

Add the Appropriate Assignments to the DB2 Alphablox Style

For each application, you need to assign three URL prefix wildcard characters to the **alphablox** style. These assignments forward the appropriate requests to DB2 Alphablox. The following is the procedure to add the new assignments to the **alphablox** style.

1. Open the iPlanet Web Server Administrative pages. By default, these pages are accessible via port 8888 on the server in which iPlanet is running, but the port might be different on your system.
2. Select your server from the drop down list and click the **Manage** button.
3. Click the **Styles** tab.
4. Click the **Assign Style** link.
5. Choose the `alphablox` style from the drop list.
6. If your application is a standard DB2 Alphablox application, add three **URL prefix wildcard characters** to assign to the `alphablox` style:

   MyApplication`/abx/*`
   MyApplication`/servlet/*`
   MyApplication`/*.jsp`

   where *MyApplication* is the context name of your application.
7. If your application uses 3.x compatibility, add the following **URL prefix wildcard** to assign to the alphablox style instead of the ones in the previous step:

   MyApplication`/*`

   where *MyApplication* is the context name of your application.
8. Save and apply these changes for all of the URL forwarding requests.
9. If you want to confirm that you added the assignments correctly, click the **List Assignments** link and examine the list of assignments to the alphablox style.

Importing an Existing J2EE Application

You can import an existing J2EE application to define it as an DB2 Alphablox application. After importing a J2EE application to DB2 Alphablox, an application has access to all of the DB2 Alphablox application services such as Blox rendering, bookmarks, etc.

Importing J2EE applications is a common way of making applications available when running with WebSphere. For details on defining an application when running with WebSphere, see "Defining an Application When Running With WebSphere" on page 31.

Note: When importing any existing Apache Tomcat applications that were created before DB2 Alphablox was installed, you will also need to add

`crossContext=true` in various locations, including the *$CATALINA_HOME/conf/context.xml* file. For more information, see http://tomcat.apache.org/tomcat-5.5-doc/config/context.html.

To import an existing J2EE application to DB2 Alphablox:

1. You must know the application context name of an existing J2EE application you want to import to DB2 Alphablox.
2. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
3. Click the **Administration** tab.
4. Click the **Applications** link.
5. Click the **Create** button.
6. Set the **Import J2EE Application** drop list to **Yes**. The Importing Existing J2EE Application window appears.
7. Press the **Lookup Application** button. The Import an Application window appears.
8. Using the drop-down lists, select the **J2EE Application Name** (not available in Apache Tomcat) and the **Module Name**. The read-only **URI** and **Context Path** fields display values based on your selections.
9. Press the **Select Application** button. The dialog window closes and the **Importing Existing J2EE Application** form is filled automatically.
10. Press **Save** to complete the import of the selected J2EE application. Press Cancel to return to the Applications main page or select **No** under **Import J2EE Application** to return to the **Create Application** window.

After successfully importing the J2EE application, the new application appears in the list of applications. The application will have the structure described in "DB2 Alphablox Application Definitions" on page 29.

Chapter 7. Data Source Definitions

This chapter contains administrative information about multidimensional and relational data sources, and provides procedures for creating, modifying, and deleting data source definitions in DB2 Alphablox.

Defining a New Data Source

DB2 Alphablox supports multidimensional and relational data sources. The **Data Source** page of the **Administration** tab allows you to define a data source for one or more DB2 Alphablox applications to access. DB2 Alphablox applications use these data sources to access relational and multidimensional databases. The **Data Sources** page under the **Administration** tab changes depending on the type of data source being defined

Perform the following to define a new data source:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab.
3. Click the **Data Source** link.
4. Click the **Create** button. The **Create Data Source** page appears.
5. In the **Data Source Name** text box, provide a unique name (required) and description for the data source. Allowable characters are A-Z, a-z, 0-9, underscore, or special characters (for example, accented characters) when running in a language other than English. The display of the name is case sensitive but the actual name that is authenticated is case insensitive. The names *Public*, *Private*, and *Properties* are reserved and cannot be used for an object name.
6. From the **Adapter** drop list, select the database adapter to use for connecting to the data source.

   Note: DB2 Alphablox provides a Canned data adapter for testing a connection between client and server. If you suspect a problem in this area, contact DB2 Alphablox Customer Support for assistance on using this adapter.
7. In the **Server Name** text box, specify the name of the host machine on which the data source resides. (Some of the names are different for different data adapters.)
8. Specify any other adapter-specific information for accessing the data source, for example: **Default Catalog**, **Port Number**, **Database (catalog)**, **Schema**, **SID**, **Username**, and **Password**.

   Note: For WebSphere and WebLogic application server installations, **Password** will display as **Data Source Password** since this password will not be used to authenticate for the applications. In these cases, user authentication is handled by your WebSphere or WebLogic server. When **Use DB2 Alphablox Username and Password** is enabled on your applications, the password will only be used for data source authentication.

   Note: Microsoft SQL Server users: note that the name of a database conforms to the rules for the format of identifiers in Microsoft SQL Server. There

are two classes of identifiers: regular identifiers and delimited identifiers. Any identifier that does not comply with the rules for the format of regular identifier is a delimited identifier and must always be delimited by double quotation marks or brackets. For example, if your MS SQL database is named `test-1`, the database (catalog) name in the DB2 Alphablox data source name should either be `[test-1]` (brackets included in the data source name) or `"test-1"` (quotes included in the data source name).

9. If you are using a Microsoft SQL Server data source, you must specify a database user that is defined as a SQL Server authenticated user. SQL Server users that use Windows authentication will not work correctly with DB2 Alphablox.
10. Specify the **Maximum Rows** and **Maximum Columns** if you want to limit the size of a query result set from this data source.

    Note: Do not set this value to less than `20`. The default of `1000` should be sufficient for most applications. If a large query gets truncated, the user will get an error dialog in the browser.
11. If your are using an Oracle data source, specify the number of rows that should be retrieved in preparation for display in the **Row Prefetch** field. The default value is `100`.
12. Click the **Save** button to define the new data source and return to the **Data Sources** page.

Changing or Deleting an Existing Data Source Definition

This section describes how to modify or delete an existing data source.

Changing an Existing Data Source Definition

Perform the following to modify an existing data source:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab.
3. Click the **Data Source** link.
4. Click the **Edit** button. The **Edit Data Source** page appears.
5. Change the appropriate values.
6. Click the **Save** button to change the definition and return to the **Data Sources** page.

Deleting an Existing Data Source Definition

Perform the following to delete an existing data source:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab.
3. Click the **Data Source** link. The **Data Source** page appears.
4. Click the **Delete** button to delete the data source definition and return to the **Data Sources** page.

   Important: Deleting a data source permanently removes it from DB2 Alphablox.

Setting Up Microsoft Authentication for Microsoft Analysis Services Data Sources

Microsoft Analysis Services servers, including Microsoft SQL Server 2000 Analysis Services and Microsoft SQL Server 2005 Analysis Services, uses Windows-based authentication. To use Windows-based authentication in DB2 Alphablox applications that access Microsoft Analysis Services, the `userName` and `password` properties passed by DB2 Alphablox to Microsoft Analysis Services must be Windows users and passwords. If DB2 Alphablox and Microsoft Analysis Services are in different domains, they must be trusted Windows domains.

To enable DB2 Alphablox to log users into Microsoft Analysis Services correctly, the Windows user in which DB2 Alphablox runs (that is, the Windows user who is logged in when DB2 Alphablox starts up) must have the "Act as a part of the operating system" user right. If you are running DB2 Alphablox as a Windows service, you can configure the service to run under a default Windows user. This section describes the steps needed to configure your Windows machine to use this form of Microsoft Analysis Services authentication and contains the following procedures:

- "Setting Up the Windows User Rights"
- "Configure the Windows Service" on page 42
- "Ensure Users Are Configured in Microsoft Analysis Services" on page 43

Setting Up the Windows User Rights

This section describes the steps necessary to configure your Windows machine to access data in a Microsoft Analysis Services database.

Note: If you are running DB2 Alphablox in a clustered environment and accessing Microsoft Analysis Services data sources, each node of the cluster must be configured to run under the same Windows user and each machine must have the security policy set as described in this section.

Set the Rights for the Windows User

To enable DB2 Alphablox to connect to into Microsoft Analysis Services correctly, the Windows user in which DB2 Alphablox runs (that is, the Windows user who is logged in when DB2 Alphablox starts up or, if DB2 Alphablox is running as a service, the user under which the service runs) must have the "Act as a part of the operating system" user right.

Note: The Windows user must be set up in Microsoft Analysis Services.

On Windows systems, perform the following steps to enable this user right for the user under which DB2 Alphablox runs:

1. From the Windows **Start** menu, select **All Programs > Administrative Tools > User Manager**

   The Windows User Manager window appears.
2. From the **Policies** menu, select **User Rights**.
3. Select the check box labeled **Show Advanced User Rights**.
4. In the drop list labeled **Right**, select **Act as a part of the operating system**.
5. Click the **Add** button.
6. In the **Add User and Groups** dialog box, ensure that the drop list labeled **List Names** From displays the correct domain or machine name for the account to be added.

7. Click the **Show Users** button.
8. In the **Names** list box, select the user account to be added.
9. Click the **Add** button. The user name now is listed in the **Add Names** list box.
10. Reboot the machine.

Note: The user in the DB2 Alphablox data source that accesses Microsoft Analysis Services must match the Windows user granted the "Act as part of the operating system" user right.

Set the Rights for the Windows User

On Windows systems, perform the following steps to enable this user right for the user under which DB2 Alphablox runs:

1. From the Windows **Start** menu bring up the control panel folder by selecting **Settings > Control Panel**.
2. Open **Administrative Tools** .
3. Open the **Local Security Policy** document from the **Administrative Tools** control panel folder.
4. Expand the **Local Policies** folder in the **Local Security Settings** window.
5. Select the **User Rights Assignment** folder.
6. In the right hand pane of the **Local Security Settings** window, double-click the **Act as part of the operating system** user right.
7. In the **Local Security Policy Settings** window for the **Act as Part of the Operating System** user right, click the **Add** button and select the user under which DB2 Alphablox will run. If DB2 Alphablox is running under a network user, be sure to select the correct network user (for example, CORPNET\alphablox for a network user named *alphablox* on the CORPNET Windows network).
8. Check the **Local Policy Setting** box and click the **OK** button.
9. Close the **Local Security Policy** window.
10. Reboot the machine.

Note: The user in the DB2 Alphablox data source that accesses Microsoft Analysis Services must be the same Windows user for which you granted the "Act as part of the operating system" user right.

Configure the Windows Service

To configure DB2 Alphablox to run under a different user than the user who is logged in, you must run DB2 Alphablox as a service. If DB2 Alphablox connects to Microsoft Analysis Services, the Windows user must be set up in Microsoft Analysis Services and must have the "Act as part of the operating system" user right.

To configure the service to start under a different Windows user, perform the following steps:

1. From the Windows Start menu, select **Settings**.
2. Select **Control Panel** from the pull down menu.
3. Double-click on the **Services** icon.
4. In the **Services** window, double-click the **DB2 Alphablox (AlphabloxAnalytics)** service.
5. In the **Service** dialog box, select the **This account** radio button.
6. Enter a user name in the text box to the right.

7. Enter a password in the **Password** text box.
8. Verify the password by typing it again in the **Confirm Password** text box.
9. Click the **OK** button.

Ensure Users Are Configured in Microsoft Analysis Services

Any user specified in an DB2 Alphablox application to connect to Microsoft Analysis Services, whether configured through a DB2 Alphablox data source or specified with user name and password parameters (or the associated methods) in the application, must be a Windows user and must be properly configured in Microsoft Analysis Services. Note that the Windows domain group `Everyone` includes only domain users, not local machine users.

Note: If you attempt to log in to Microsoft Analysis Services as a user that is local to your machine, that user must be part of the Administrators group as well as an OLAP Administrator in order to be able to reliably connect. If the user is not in both administrator groups, the user will be able to log in the first time, but a subsequent attempt will fail with an exception such as:

```
com.alphablox.util.NotAuthorizedException: Provider cannot be
found. It may not be properly installed
```

Working With JDBC Data Sources

This section includes the following sections related to JDBC data sources:

Setting Up the Environment for the Sybase JConnect Relational Driver

If you are using the Sybase JConnect driver to connect to a Sybase relational database, you must run a SQL script file on the Sybase database server. The SQL script files are located in the following directory:

```
<db2alphablox_dir>/tools/sql
```

where `<db2alphablox_dir>` is the directory in which DB2 Alphablox is installed. The following table shows the name of the SQL script and the Sybase database server on which the script must be run.

SQL Script	Sybase Database Server on Which to Run Script
sql_asa.sql	SQL Anywhere 5.5.02 or higher
sql_server.sql	SQL Server versions below version 12
sql_server12.sql	SQL Server version 12 and later

Determine which Sybase database server you are running and run the appropriate script on the database server. Each script sets up some stored procedures and system objects that the Sybase JConnect JDBC driver requires. For details on running a SQL script file on your Sybase database server, see your Sybase documentation.

You must execute the appropriate SQL script on your Sybase database server to be able to connect to it from DB2 Alphablox.

Setting Up JDBC Tracing

For any of the relational data source drivers in DB2 Alphablox, you can enable JDBC tracing. With JDBC tracing enabled, the JDBC driver logs information messages to the DB2 Alphablox Console and log file. All JDBC tracing messages are logged at the INFO message level.

To enable JDBC tracing for a relational data source, set the **JDBC Tracing Enabled** drop list to **Yes**. This option is only available on data sources that use one of the supported relational database drivers. After you have enabled JDBC tracing, set the message level to INFO or higher. For example, in a Console window, enter the following command:

```
report info
```

Use JDBC tracing to debug any problems with connections to relational databases. When JDBC tracing is enabled, the following information is logged:

- JDBC driver logging information
- Connection information from DB2 Alphablox connecting to relational databases
- Information about column metadata from each result set that the database returns to DB2 Alphablox

If you are having connection problems or need more information about understanding JDBC tracing, contact DB2 Alphablox Customer Support.

Updating a Supported JDBC Driver to a Different Version

If you need to update one of the supported JDBC drivers used by DB2 Alphablox to access a relational database, perform the following steps:

Important: DB2 Alphablox tests only the supported JDBC drivers listed in System Requirements section in the *Installation Guide*; other drivers have not been tested and might cause unexpected results. For more information, contact DB2 Alphablox Customer Support.

1. Obtain the Java classes for the new JDBC driver. Typically, these files have either a `zip` or `jar` file extension (for example, `ojdbc14.jar`).
2. Find the directory specified during installation for your JDBC drivers.
3. Rename the JDBC driver file you are updating (for example, changing it `<driverName>.zip.old`) to keep a copy available in case you need to revert to the supported version.
4. Rename your new driver file to be the same name as one of the existing supported drivers, then copy it to the `<db2alphablox_dir>/lib/` directory.
5. Update the DB2 Alphablox startup environment to access the new classes for the JDBC driver as described in the following section.
6. After updating the classpath information, restart the server for the changes to take effect.

Adding Additional JDBC Drivers

Relational data sources available to DB2 Alphablox applications by default, upon installation or upgrade, are defined in the `jdbcdrivers.xml` file. You should not modify this file—it is automatically overwritten during upgrades of DB2

Alphablox. If you need to create additional DB2 Alphablox data sources, you can use an optional jdbcdrivers_additional.xml file.

Perform the following steps to create additional data sources for DB2 Alphablox:

1. Find the jdbcdrivers_additional.xml file located in the following directory:
   <db2alphablox_dir>/repository/servers/
2. You can edit this file, but you should make a backup copy of this file in the same directory and name it jdbcdrivers_additional.xml. A backup copy of the default jdbcdrivers_additional.xml file is in the same directory, named jdbcdrivers_additional.xml.example.
3. Using the JDBC driver examples contained within this file, create any additional data sources that you need, making sure to:
   a. In each new JDBC driver definition that you want to make available, change the disabled attribute from true to false. By default, the example JDBC drivers are disabled.
4. You must restart the application server for these changes to take effect.
5. Edit all DB2 Alphablox classpaths to include your new driver, as described below in "Modifying Classpath Settings" on page 45.

Modifying Classpath Settings

After you add additional drivers to the jdbcdrivers_additional.xml file, you must also update classpath information so that the new JDBC drivers can be found. Follow the steps below for your configuration.

WebSphere

To add or remove JDBC drivers on WebSphere, you do not need to modify any classpath settings. Add additional JDBC drivers by placing them in the <websphere_dir>/AppServer/lib/ext directory, or delete unused or old JDBC drivers from the directory. Restart the application server for the classpath changes to take effect.

WebLogic

For BEA WebLogic application server installations, classpath information can be modified in the following batch command file: <db2alphablox_dir>/bin/jdbcsetup.bat (Windows) or <db2alphablox_dir>/bin/jdbcsetup.sh (UNIX). Restart the application server for the changes to take effect.

Tomcat

Perform the following steps to modify classpath information for DB2 Alphablox running under Apache Tomcat only:

Windows: If Tomcat is running in a Windows console, open the <db2alphablox_dir>/bin/jdbcsetup.bat file in a text editor and modify the classpath settings as required.

If the application server is running as a Windows service, open the <tomcat_dir>/conf/wrapper.properties file (in your Tomcat directory, not the DB2 Alphablox directory) in a text editor and modify the classpath settings as needed. Restart the application server for the changes to take effect.

Linux and UNIX: If you are using Apache Tomcat with DB2 Alphablox on a Linux or UNIX system, open the <tomcat_dir>/bin/analytics.sh file in a text editor and modify the classpath settings as required. Restart the application server for the changes to take effect.

Chapter 8. User Definitions

The DB2 Alphablox **Users** administration page allow you to create, change, and delete users.

Information on **Applications** and **Data Source** pages is also available by pressing the **Help** link in the upper right corner of each page.

Note: For changing existing users, be sure click the **Save** button before going to a different page. Otherwise, the changes made on the page will not be saved.

Creating a New User

The **Users** link under the **Administration** tab opens pages that are used to create and edit new users. During installation, two users (Admin and Guest) are created. Click the **Create** button on the **Users** page to define additional users. Select an existing user name and click the **Edit** or **Delete** button to modify or delete it. If a user becomes damaged, you can repair them by restarting DB2 Alphablox. This section shows the steps to define a new user from the DB2 Alphablox home page.

Perform the following to define a new user:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Users** Link. The **Users** page appears.
4. Click the **Create** button. The **Create User** page appears.

   Note: Depending on your security configuration, this screen might not contain the password information.
5. In the **Username** text box, type the ID for the user (required). The user name is the name used to log into DB2 Alphablox. Allowable characters are A-Z, a-z, 0-9, underscore, or special characters (for example, accented characters) when running in a language other than English. The display of the user name is case sensitive but the actual user name that is authenticated is case-insensitive. The names *Public*, *Private*, and *Properties* are reserved and cannot be used for an object name.
6. Optionally, enter a **Full Name**, **Description**, and **eMail Address** for the user.
7. In the **Password** text box, enter the user's initial password. The password is case-sensitive.
8. In the **Confirm Password** text box, reenter the password to confirm it.
9. In the **Allow User To Edit Profile** drop list, indicate whether the user has privileges to edit the profile. Setting this property to **Yes** lets the user access and modify her profile using the **My Profile** link on the DB2 Alphablox home page.
10. Provide values for any custom properties. (If there are any custom user properties defined, they appear below the **Allow User to Edit Profile** drop list.)
11. Click the **Save** button to define the new user and return to the Users page.

Note: When you create a new user, the user is automatically assigned membership in the *Public* group.

Changing or Deleting an Existing Group or User

This section shows the steps to change the properties or delete an existing user or group from the DB2 Alphablox home page.

Changing the Properties of an Existing User

Perform the following to change the properties of an existing user:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Users** Link. The **Users** page appears.
4. Click the **Edit** button. The **Edit User** page appears.
5. Change any values that need changing.

   Note: In WebSphere and WebLogic installations, "Password" will display as "Data Source Password" since this password will not be used to authenticate for the applications. In these cases, user authentication is handled by WebSphere or WebLogic. This also means that when Use DB2 Alphablox Username and Password on your applications, the password will only be used for data source authentication.
6. Click the **Save** button to save your changes to the **General Properties** panel and return to the **Users** page.
7. To change the user's application properties, click the **Application Properties** tab. Make any necessary changes and click the **Save** button.
8. To change the user's group memberships or role memberships, click the **Memberships** tab. Make any necessary changes. Note that the changes to membership take effect immediately. For information about group membership and role membership, see "Changing the Groups to Which a User Belongs" on page 48 and "Changing the Roles to Which a User or Group Belongs" on page 56.

Deleting an Existing User

Perform the following to delete an existing user:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Users** Link. The **Users** page appears.
4. Select a user from the list of users.
5. Click the **Delete** button to delete the user.

   Important: Deleting a user permanently removes the user from DB2 Alphablox.

Changing the Groups to Which a User Belongs

This section shows the steps to add or modify group membership for users and groups.

Perform the following to modify the groups to which a user belongs:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Users** Link. The **Users** page appears.
4. Click the **Edit** button. The **Edit User** page appears.
5. Click the **Membership** tab.
6. To add a new group to which the user is a member, select a group in the **Available Groups** list and click the left arrow button to move it into the **Group Memberships** list.
7. To remove a group to which the user is a member, select a group in the **Group Memberships** list and click the right arrow button to move it into the **Available Groups** list.
8. To add a new role to which the user is a member, select a role in the **Available Roles** list and click the left arrow button to move it into the **Role Membership** list.

   Note: Changes made to a user's group or role membership take effect immediately.

Chapter 9. Group Definitions

The **Groups** administration pages allow you to create, change, and delete groups.

Information on the **Groups** pages is also available by pressing the **Help** link in the upper right corner of each page.

Note: For changing existing groups, be sure click the **Save** button before going to a different page. Otherwise, the changes made on the page will not be saved.

Creating a New Group

The **Groups** link under the **Administration** tab open pages that are used to create and edit new groups. During installation, two groups (Administrators and Public) are created. Click the **Create** button in the **Groups** page to define additional groups. Select an existing group name and click the **Edit** or **Delete** button to modify or delete it. If a group becomes damaged, you can repair it by restarting DB2 Alphablox. This section shows the steps to define a new group from the DB2 Alphablox home page.

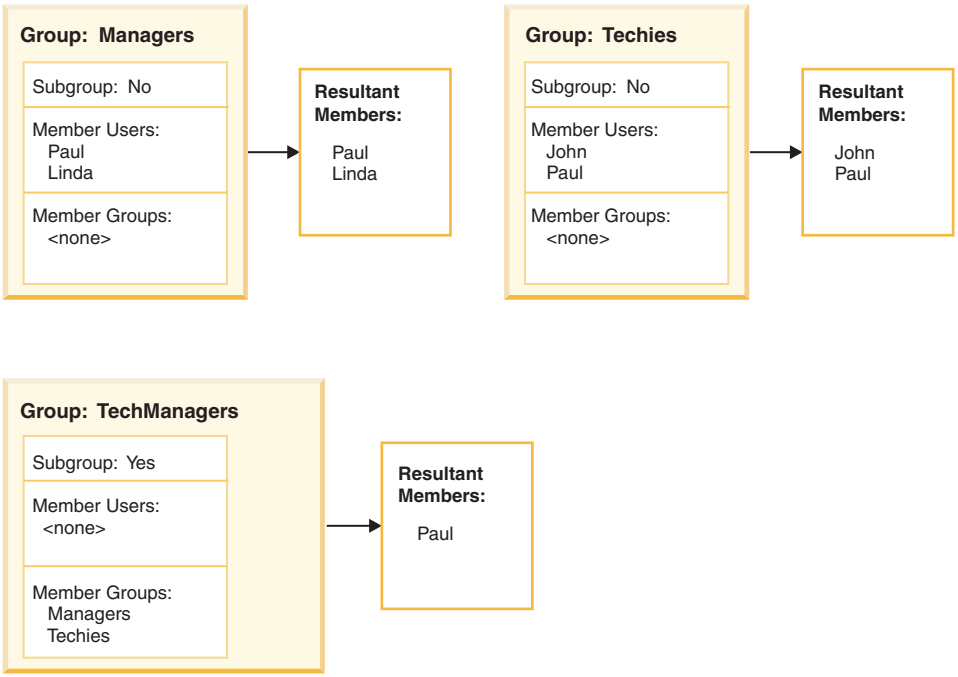
Perform the following to define a new group:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Groups** Link. The **Group** page appears.
4. Click the **Create** button. The **Create Group** page appears.
5. In the **Group Name** text box, enter the name of the group (required). Allowable characters are A-Z, a-z, 0-9, underscore, or special characters (for example, accented characters) when running in a language other than English. The display of the group name is case sensitive but the actual group name that is authenticated is case insensitive. The names *Public*, *Private*, and *Properties* are reserved and cannot be used for an object name.
6. Optionally, enter a **Description** for the group.
7. In the **Subgroup** drop list, select **Yes** if this group will act as a subgroup or **No** if it will not. For information about subgroups, see "Understanding Subgroups" on page 52.
8. Add users and/or groups to this group in the **Member Users** and **Member Groups** selection boxes. To do so, scroll through the lists of users and groups. Hold down the Ctrl key to select multiple users or groups to add to this group.
9. Provide values for any custom properties. (If there are any custom group properties defined, they appear below the **Member Groups** selection box.)
10. Click the **Save** button to define the new group and return to the **Groups** page.

Understanding Subgroups

A *subgroup* is a mechanism in which users who are in multiple groups can automatically become a member of another group. When a group has the **Subgroup** drop list set to **Yes**, its resultant user members are comprised of the intersection of the users in each of the groups assigned as members to the new group.

The following figure shows an example using a subgroup. In this example, the group *TechManagers* uses a subgroup to automatically include the members that are in both the *Managers* and the *Techies* groups.



By using the subgroup for the *TechManagers* group, an administrator creates a group that automatically includes the set of all common members of the *Managers* and the *Techies* groups. In this example, the only user in both of those groups is *Paul*. If other users were added to both the *Managers* and the *Techies* groups, they would automatically become members of the *TechManagers* group.

Note: When using subgroups, you can still manually add individual users to the group by selecting them in the **Member Users** selection box.

Changing or Deleting an Existing Group

This section shows the steps to change the properties or delete an existing user or group from the DB2 Alphablox home page.

Changing an Existing Group

Perform the following to modify an existing group:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Groups** Link. The **Groups** page appears.

4. Select a group from the list of groups.
5. Click the **Edit** button. The **Edit Group** page appears.
6. Change any values that need changing.
7. Click the **Save** button to save your changes and return to the **Users** page.
8. To change the user's application properties, click the **Application Properties** tab.
9. Make any necessary changes.
10. Click the **Save** button.
11. To change the user's group or role memberships, click the **Memberships** tab.
12. Make any necessary changes. Note that the changes to membership take effect immediately. For information about group membership and role membership, see "Changing the Groups to Which a User Belongs" on page 48 and "Changing the Roles to Which a User or Group Belongs" on page 56.
13. Click the **Save** button.

Deleting an Existing Group

Perform the following to delete an existing group:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Groups** Link. The **Groups** page appears.
4. Select a group from the list of groups.
5. Click the **Delete** button to delete the group.

Important: Deleting a group permanently removes the group from DB2 Alphablox.

Chapter 10. Role Definitions

The **Roles** administration pages allow you to create, change, and delete roles. The Roles page only appears if you are running DB2 Alphablox in a Apache Tomcat configuration; if you are using WebSphere or WebLogic, the application server provides this functionality. User roles are usually specified in the web application server's application deployment tool. Once the roles are defined there (and added to the deployment descriptor file), these roles can then be used within DB2 Alphablox correctly.

Information on the **Roles** pages is also available by pressing the **Help** link in the upper right corner of each page.

Note: For changing existing roles, be sure click the **Save** button before going to a different page. Otherwise, the changes made on the page will not be saved.

Defining New Roles

DB2 Alphablox can control access to applications through roles. A role contains a list of users and/or groups, and the access permission associated with each entry in the list.

Perform the following to create a new role:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Roles** link. The **Roles** page appears.
4. Click the **Create** button. The **Create Role** page appears.
5. In the **Role Name** text box, enter the name of the role (required). Allowable characters are A-Z, a-z, 0-9, underscore, or special characters (for example, accented characters) when running in a language other than English. The display of the name is case sensitive but the actual name that is authenticated is case insensitive. The names *Public*, *Private*, and *Properties* are reserved and cannot be used for an object name.
6. Optionally, enter a **Description** for the role.
7. Click the **Save** button to define the new role and return to the **Roles** page.

Changing and Deleting Existing Roles

Perform the following to change an existing role:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Roles** Link. The **Roles** page appears.
4. Select a role from the list of roles.
5. Click the **Edit** button. The **Edit Roles** page appears.
6. If necessary, change the description for the role.
7. Click the **Save** button before proceeding, otherwise the changes will not be saved.

8. To change the groups who are assigned to the role, click the **Member Groups** tab. A page appears that lists the groups currently members of the role, and a list of available groups not currently members the role. Use the left and right arrow buttons to move selected items from the **Available Groups** list to the **Member Groups** list and from the **Member Groups** list to the **Available Groups** list.
9. Click the **Save** button to save the group membership properties for the role.
10. To change the users who are members of the role, click the **Member Users** tab. A page appears that lists the users currently members of the role, and a list of available users not currently members the role. Use the left and right arrow buttons to move selected items from the **Available Users** list to the **Member Users** list and from the **Member Users** list to the **Available Users** list.
11. Click the **Save** button to save the user membership properties for the role.

Changing the Roles to Which a User or Group Belongs

Perform the following to change the roles to which a user or group is assigned:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the either **Users** link or the **Groups** link. The **Roles** page appears.
4. Select the user (or group) from the list of users (or groups) on the **Users** or **Groups** page.
5. Click the **Edit** button. The **Edit User** (or **Edit Group**) page opens.
6. Click the **Memberships** tab.
7. The bottom half of the **Memberships** panel has one box for **Role Memberships** and one for **Available Roles**. Use the left and right arrow buttons to move selected items from the **Available Roles** list to the **Role Memberships** list and from the **Role Memberships** list to the **Available Role** list.

Deleting an Existing Role

Perform the following to delete an existing role:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. Click the **Roles** Link. The **Roles** page appears.
4. Select a role from the list of roles.
5. Click the **Delete** button to delete the role.

   Important: Deleting an role permanently removes it from DB2 Alphablox.

Chapter 11. Security and authentication

These topics describes security and authentication using DB2 Alphablox running in a Apache Tomcat configuration, using either no external Web server or using another Web server (Sun iPlanet, Microsoft IIS, or Apache HTTP Server).

For information about security and authentication when running with a commercial application server, such as WebSphere or WebLogic, see the documentation for those products.

Note: DB2 Alphablox 8.4 and 8.4.1 support different external Web servers. DB2 Alphablox 8.4.1 does not support Sun iPlanet. DB2 Alphablox 8.4.1 supports Apache HTTP Server 2.0, and DB2 Alphablox 8.4 supports Apache HTTP Server 1.3. See the system requirements in the *Installation Guide* for details.

DB2 Alphablox authentication and security modes

Before DB2 Alphablox can deliver a page to a user, the user must be authenticated. This is true regardless of the underlying Web server. However, when you use an external Web server, the authentication to DB2 Alphablox is usually transparent.

By default, HTML pages are not routed to the application server in DB2 Alphablox configurations behind Web servers and are not challenged until a JSP page or a page with DB2 Alphablox content is received by the application server. Web servers can be configured to authenticate HTML pages, but a DB2 Alphablox configuration does not support this authentication by default. If you want to authenticate when DB2 Alphablox is configured behind a Web server, either configure your Web server to support authentication of all HTML pages or ensure that your welcome page is a JSP file.

DB2 Alphablox supports two types of user authentication:

Authentication type	Description
Application server authentication	When DB2 Alphablox is operating in an application server without an external web server, authentication and authorization is managed by the application server.
Web server authentication	When you use Sun iPlanet, Apache HTTP Server, or Microsoft IIS as the Web server, the Web server performs authentication. When DB2 Alphablox attempts to authenticate a user, the Web server passes DB2 Alphablox a user ID. **Note:** The Web server does not pass the user ID and password to DB2 Alphablox, because this would constitute a security breach.

Security model in Alphablox 8.4.1

DB2 Alphablox 8.4.1 provides a new security model that is based on the standard Servlet 2.4 security API. Because Alphablox 8.4.1 and 8.4 support different versions of Apache Tomcat, the support for security also differs because the two Tomcat version support different security models and servlet specifications. In DB2 Alphablox 8.4.1, security information is now configured directly in Apache

Tomcat's *conf/server.xml* file. To configure security in Tomcat 5.5, see the Apache Tomcat documentation on configuring realms (http://tomcat.apache.org/tomcat-5.5-doc/realm-howto.html).

Alphablox APIs that support this security model are provided in three packages:

- com.alphablox.security.jaas
- com.alphablox.security.jndi
- com.alphablox.security.ntlm

Java Authentication and Authorization Service (JAAS)-based Alphablox Login Module

The com.alphablox.security.jaas package provides a JAAS-based Alphablox login module. The com.alphablox.security.jaas.AlphabloxLoginModule class authenticates users by accessing authentication information from the DB2 Alphablox Repository. If the `debug` option is set to `true` (the default) in the *conf/alphablox-jaas.config* login configuration file in your Tomcat 5.5 installation directory, debug messages are output to the Alphablox server log.

JNDI realm configuration in Tomcat 5.5

The Apache Tomcat JNDI realm does not support LDAP dynamic groups, but DB2 Alphablox offers the Alphablox JNDI realm that can be used to access user and group information from LDAP dynamic groups. The com.alphablox.security.jndi.AlphabloxJNDIRealm class extends the basic support for JNDIRealm in Tomcat 5.5 by additionally searching for role membership through LDAP dynamic groups. This class defines four additional attributes to those provided in JNDIRealm.

Attribute	Description
`userName`	Optional. Sets the attribute name to be read as the user's unique name. This value should match the attribute that is used in the JNDI realm's `userPattern` or `userSearch` strings. The default is `uid`.
`memberURL`	Optional. Sets the dynamic group URL attribute name. The default is `memberURL`.
`cacheEnabled`	Optional. Enables or disables caching of dynamic groups to an associated user name. The default is `true`.
`dynamicGroupSearch`	Required. Sets the JNDI filter that is used to search the `roleBase` for LDAP dynamic groups. The default is null. Enabling the cache (`cacheEnabled="true"`) helps in performance because dynamic groups do not need to be queried repeatedly.

See the Javadoc documentation in the DB2 Alphablox Information Center for detailed API information.

Adding the Alphablox JNDI realm to Apache Tomcat 5.5

To add an Alphablox JNDI realm in Tomcat, configure the Apache Tomcat *conf/server.xml* file with the `className` set to com.alphablox.security.jndi.AlphabloxJNDIRealm:

```
<Realm
   className="com.alphablox.security.jndi.AlphabloxJNDIRealm"
   connectionURL="ldap://your_ldap.your_server.com:port"
   userBase="..."
   userSearch="..."
   userRoleName="..."
```

```
  roleBase="..."
  roleName="..."
  roleSearch="..."
  dynamicGroupSearch="(&amp; (cn={0}) (objectclass=groupofurls))"
/>
```

Note:

- The "&" character needs to be encoded as "&" in the XML.
- You need to restart Tomcat after you make the changes to the *server.xml* file.

Admin versus user rights

When DB2 Alphablox is installed, it creates a single user profile with administrative privileges. The name for this profile is *admin* and cannot be changed. The default password for *admin* is *password*. When you use Web server security, the user named *admin* must exist under the Web server's authentication scheme. (When you use IIS and Windows® NTLM, *admin* can be a local user instead of a domain user.)

The *admin* user has read/write access to DB2 Alphablox objects, including user properties, through the DB2 Alphablox **Administration** pages. DB2 Alphablox users have read/write access only to their own user properties. The *admin* user or another DB2 Alphablox user who is a member of the administrators group can grant access to one or more applications to specific users, groups of users, or all users.

Removing guest login rights for applications

By default, DB2 Alphablox installs with two default users, Guest and Admin. To prevent the Guest user from being able to access an application, you can modify an application's `web.xml` file by replacing `AlphabloxUser` with `AlphabloxAuthenticatedUser` in the `role-name` elements for both the `auth-constraint` and `security-role` elements.

Application server security realms and applications

Frequently, applications are created on a development server and later moved to production servers when applications are deployed. When moving applications between two different servers that do not share the same instance name, developers and administrators need to be aware that the security realm that is specified in the migrated application's `web.xml` file might not match the security realm in the new location. On IBM WebSphere and BEA WebLogic application servers, when applications attempt to access other applications on the same server with different security realms, the users will be prompted for authentication. To prevent this from occurring, the application's `web.xml` file needs to be modified so that the security realms match.

Another situation in which users might be prompted for authentication is when an application received as a WAR or EAR file is imported into WebSphere and WebLogic application servers. The instance name is not checked, and it could be a different instance name.

Web server authentication versus DB2 Alphablox authentication

DB2 Alphablox allows only authenticated users to access applications. If not properly authenticated, a user cannot connect to a data source, retrieve data, or view data on application pages. Because DB2 Alphablox does not lock access at the directory level, a user who authenticates successfully can use the browser to attempt to open a restricted application. Although the user could open the application page, data would not appear in the page.

For information about preventing users from browsing directories and files beneath the directory where applications reside, see "Disable directory browsing" on page 66.

In a default installation, DB2 Alphablox performs its own authentication. If you use another Web server, however, you can set up DB2 Alphablox to have the Web server perform the authentication. Web server authentication can ease the administrative burden on the DB2 Alphablox administrator because it can authenticate based on user accounts that might already exist on the local area network. This section describes how to set up DB2 Alphablox to have the Web server perform authentication..

Note: When you select the WebServerRealm adapter, the security of the web server will take precedence over anything else. In these cases, it doesn't matter what the password is in DB2 Alphablox.

Using the Sun iPlanet Web Server security options on Alphablox 8.4 with Apache Tomcat 3.2.4

The Sun iPlanet security model works well for most user and browser combinations in basic or clear text mode. To administer DB2 Alphablox via a URL from iPlanet, add the user named *admin* to the Web server's user database. Otherwise, access the DB2 Alphablox URL directly (for example, use `http://<servername>:<port>/AlphabloxAdmin/home`).

To use DB2 Alphablox security with iPlanet, disable security on the iPlanet Web server. All users are then authenticated by DB2 Alphablox.

In addition, be sure to establish the correct DB2 Alphablox settings to ensure a functional and secure environment. For more information, see "Configuring DB2 Alphablox to use Web server-based security" on page 65.

Setting Microsoft security options for IIS NTLM

When using DB2 Alphablox with a Microsoft IIS web server, you can set up the security authentication so that IIS performs the authentication when a user logs into DB2 Alphablox (instead of DB2 Alphablox performing the authentication). If IIS is configured to accept Windows logins in a trusted domain, then those trusted users can access DB2 Alphablox without any additional authentication; that is, they do not have to enter a user name or password when they are accessing DB2 Alphablox while logged in on the trusted domain. A *trusted* user is a Windows user that has been authenticated on a Windows trusted domain, typically from a large corporate Windows server.

In this configuration, IIS handles all request authentication. DB2 Alphablox requires authenticated requests with the user name before the requests can be handled. As a result, if a Microsoft Internet Explorer user changes the browser's

default security setting (in the Advanced Settings) to not prompt for user name and password, DB2 Alphablox cannot handle requests from these browsers.

To set up DB2 Alphablox to operate securely with Microsoft Internet Information Services (IIS) using NTLM, perform the following tasks:

- "Installing Microsoft IIS"
- "Installing DB2 Alphablox and choosing Microsoft IIS as the Web server"
- "Configuring security settings in Microsoft IIS"
  - "Enabling security settings in Microsoft IIS"
  - "Restricting anonymous user rights in Microsoft IIS" on page 62
- "Creating a Windows local user named admin for NTLM" on page 63
- "Configuring NTLM security in Tomcat 5.5 for Alphablox 8.4.1" on page 63
- "Logging into DB2 Alphablox" on page 64

In addition, be sure to establish the correct DB2 Alphablox settings to ensure a functional and secure environment. For details, see "Configuring DB2 Alphablox to use Web server-based security" on page 65.

Installing Microsoft IIS

You must install Microsoft IIS *before* installing DB2 Alphablox. For information about installing IIS, see your Microsoft documentation.

Installing DB2 Alphablox and choosing Microsoft IIS as the Web server

To run DB2 Alphablox using Microsoft IIS as the Web server, you must create a new Microsoft IIS configuration by selecting the **Create new Microsoft IIS configuration** option on the Choose Web Server Configuration page during installation. The installation will install the necessary files under the Web server root directory.

To use DB2 Alphablox with IIS, you must install IIS before you install DB2 Alphablox.

Configuring security settings in Microsoft IIS

Configure the following using the Microsoft IIS Microsoft Management Console.

Enabling security settings in Microsoft IIS: To enable the appropriate security settings:

1. Open the Microsoft IIS Management Console.
2. In the left window pane, right-click on the appropriate Web server (generally, **Default Web Site**) and select **Properties**. The Default Web Site Properties window opens.
3. Click the **Directory Security** tab.
4. Press the **Edit** button. To use Windows trusted authentication through NTLM, check the **Windows Integrated authentication** check box (ensure that the other check boxes are cleared). Note that only Microsoft Internet Explorer users will be able to log in this way.

   This method works only if you specify the host name for the client.

   Note:

   - In DB2 Alphablox 8.4, to specify the authorized host names, click the **Administration** tab in the DB2 Alphablox home page, click the **System** link

under the **General Properties** section, and enter any authorized clients in the **Authorized Client List** text box of the **System** page. To limit access to clients only on the machine in which DB2 Alphablox is running, type `localhost` in the **Authorized Client List** text box.

- In DB2 Alphablox 8.4.1, specify the authorized clients in Tomcat 5.5 directly. Tomcat 5.5 lets you specify a Remote Host Filter or a Remote Address Filter to identify IP addresses or host names to allow or to deny. For example, to block all requests coming in except those from the localhost:

  ```
  <Valve
     className="org.apache.catalina.valves.RemoteAddrValve" allow="127.0.0.1" />
  ```

  For more information about the Remote Host Filter and Remote Address Filter, see http://tomcat.apache.org/tomcat-5.5-doc/config/valve.html.

DB2 Alphablox prompts the user for a user name and password, which is then authenticated by DB2 Alphablox.

Security type	Description
Allow Anonymous	This option permits users to log in using the anonymous account. Do not check this option if you are using NTLM security with IIS. After users are authenticated, they can browse to directories and files beneath the `webapps` directory where applications reside. To prevent directory browsing, see "Disable directory browsing" on page 66. Selecting this check box grants users unrestricted access to resources. For details, see "Restricting anonymous user rights in Microsoft IIS" on page 62.
Basic Authentication (clear text)	This option, for both Mozilla Firefox and Microsoft Internet Explorer browsers, results in user IDs and passwords being transmitted as clear text. **Note:** Be sure to check this option in environments where Mozilla Firefox browsers access DB2 Alphablox applications on an Microsoft IIS. Select both this option and Windows Integrated authentication in environments where both Mozilla Firefox and Internet Explorer browsers access DB2 Alphablox applications on Microsoft IIS.
Windows Integrated authentication	This is the recommended method of using IIS security. This option, only for environments using a combination of IIS web servers and Microsoft Internet Explorer browsers, causes user IDs and passwords to be transmitted in a proprietary Microsoft format, not in clear text.

5. If you checked the **Allow Anonymous Access** box (and therefore are not using trusted NTLM security), click the **Edit** button to the right of the **Allow Anonymous Access** check box. The anonymous user name (generally of the format IUSR_<HOSTNAME>) appears. Make note of the anonymous user name and skip to the instructions in "Restricting anonymous user rights in Microsoft IIS" on page 62.
6. Click the **OK** button to close the Authentication Methods window.
7. Click the **OK** button to close the Default Web Site Properties window.

Restricting anonymous user rights in Microsoft IIS: This procedure is necessary only if you are *not* using trusted NTLM security. Only advanced Microsoft IIS and Windows administrators should perform these steps. Restricting anonymous user

access only to the webapps directory does not necessarily restrict access to objects on the Web server. Therefore, this method is recommended.

If Allow Anonymous is set under IIS, perform the following steps to restrict the rights of the anonymous user:

1. Open the Windows File Explorer.
2. Locate and right-click the *Inetpub* directory to open its menu. The *Inetpub* directory is the directory that IIS uses to store documents that the Web server provides access to, and its default directory is *C:\Inetpub*.
3. Choose **Properties** from the drop list to open the **Directory Properties** window.
4. Click the **Security** tab.
5. Press the **Add** button at the bottom of the window to open the **Add Users and Groups** window.
6. Click the **Show Users** button to display the list of users.
7. Select the name of the anonymous user (for example, IUSR_*hostname*).
8. Press the **Add** button to add the anonymous user.
9. From the **Type of Access** list, scroll to and select **No Access**.
10. Press the **OK** button to return to the Directory Permissions window and select the **Replace Permission on Subdirectories** check box.
11. Press the **OK** button. Windows updates the entire directory tree under the *InetPub* directory, which takes a few minutes.

Creating a Windows local user named admin for NTLM

If you added a Microsoft IIS administrative account during the installation of DB2 Alphablox, you can skip this step and use that administrative user instead of creating an admin account on your local machine.

Trusted NTLM security requires creating a Windows account for the DB2 Alphablox *admin* user. By default, when authenticating to NTLM on Microsoft IIS, the user name corresponds to the user name in the domain of the Web server (typically a network domain).

In most cases, the DB2 Alphablox *admin* user is not a user in the network domain. Instead, add the *admin* user to the IIS host's local domain. When authenticating as *admin* under IIS, the user name takes the following format:

```
machine-name\admin
```

Configuring NTLM security in Tomcat 5.55 for Alphablox 8.4.1

1. Open the *conf/server.xml* file in your Tomcat 5.5 directory
2. Locate the definition of the AJP/13 Connector, which looks as follows:

   ```
   <Connector port="8009"
      enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
   ```

3. Add the tomcatAuthentication attribute to the Connector tag and set it to false:

   ```
   <Connector port="8009" tomcatAuthentication="false"
      enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
   ```

4. Replace your existing Realm tag by commenting it out and adding a new Realm tag to use the DB2 Alphablox Alphablox NTLM realm:

   ```
   <!--
     <Realm className="org.apache.catalina.realm.JAASRealm" debug="10"
     appName="Alphablox"
     userClassNames="com.alphablox.security.jaas.UserPrincipal"
   ```

```
    roleClassNames="com.alphablox.security.jaas.RolePrincipal"
    useContextClassLoader="false" />
  -->
    <Realm className="com.alphablox.security.ntlm.AlphabloxNTLMRealm" />
```

Logging into DB2 Alphablox

When connecting to DB2 Alphablox as the *admin* user through Microsoft IIS, you must:

- Use the correct format for the login name.

  If the IIS machine is logged into a different domain than the client from which the *admin* user is authenticating, you must pass the client domain name with the user name. For example, if the *admin* user is logging in from a client machine in the WEBDEV1 domain, the format for the login becomes:

  `WEBDEV1\admin`
- Ensure that browser authentication is set correctly.

By default, the Microsoft Internet Explorer Web browser authenticates users to NTLM with their Windows user names and passwords. To log into Microsoft IIS as a different user (for example, *admin*) requires changing this default behavior. (Because the *admin* user name and password will not match the user name and password, the login attempt will fail.)

Complete the following steps to force Microsoft Internet Explorer browsers to prompt for a user name and password during authentication:

1. In Internet Explorer, from the **Tools** menu, select **Internet Options**.
2. Select the **Security** tab.
3. If the application is used via Intranet (LAN), select **Local Intranet for Zone**.
4. Press the **Custom Level** button to set the security level.
5. In the **User Authentication** section, **Login** section, select the **Prompt for User Name and Password** button.
6. Click the **OK** button to close the Security Settings window.
7. Click the **OK** button to close the Internet Options window.

Windows Integrated authentication means that a browser will be passed a Negotiate header from the server, which specifies whether to use Kerberos authentication or Windows Integrated authentication (formerly called NTLM, or NT Challenge/Response). To skip the Negotiate process, follow the following steps:

1. Stop DB2 Alphablox and open a command prompt and navigate to the `Inetpub\AdminScripts` directory.
2. Type the following command:

   ```
   cscript adsutil.vbs set w3svc/NTAuthenticationProviders "NTLM"
   ```
3. To check the authentication, type:

   ```
   cscript adsutil.vbs get w3svc/NTAuthenticationProviders
   ```

   The default value is Negotiate, NTLM. After you set to NTLM only, it should show "NTLM".
4. Type the following command: `iisreset` *computerName* `/RESTART` This will restart IIS.
5. Restart DB2 Alphablox and you should be able to log into DB2 Alphablox through IIS.

Configuring DB2 Alphablox to use Web server-based security

Regardless of the Web server being used, be sure to review the following DB2 Alphablox settings for both convenience and security:

- "Configuring automatic generation of user accounts"
- "Filter IP addresses"
- "Set directory rights" on page 66
- "Disable directory browsing" on page 66

Configuring automatic generation of user accounts

Before the Web server can respond, even if the user has successfully authenticated on Microsoft IIS, the user must have a DB2 Alphablox user ID to make a request. Rather than having to create user accounts for every user already known to IIS or iPlanet, DB2 Alphablox can automatically generate user accounts.

To configure DB2 Alphablox to automatically create user IDs:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The General page opens.
3. In the General Properties section, click the **System** link.
4. Select the **Automatically Generate User Profiles** check box.
5. Press the **Save** button.

Subsequently, when a new user attempts to log in, DB2 Alphablox prompts for a user ID and password, and automatically creates that user's account. If you are using Microsoft IIS as your web server and want to use Windows authentication for DB2 Alphablox, you must perform the additional configuration that is described in "Setting Microsoft security options for IIS NTLM" on page 60.

Restriction: If you need to use a Mozilla Firefox browser to administer DB2 Alphablox when using the IIS NTLM security, be sure to log in with a Mozilla Firefox browser the first time you log in. Logging in with a Mozilla Firefox browser creates the user with a clear text password; otherwise, authentication will fail for Mozilla Firefox browsers.

Important: Filtering IP addresses (described in the next section) can circumvent potentially malicious users from bypassing Web server security.

Filter IP addresses

In DB2 Alphablox 8.4 (not 8.4.1), you can configure DB2 Alphablox to accept requests from only specified clients. Under Web server-based security, DB2 Alphablox should accept only requests that come from the Web server host (that is, the web server itself or administrators on the Web server host machine). On the DB2 Alphablox home page, **Administration** tab, **General Properties** page, **System** link, you can use the **Authorized Client List** property to create a list of acceptable DNS names or IP addresses. If you want only connections from the local machine to be accepted, set the **Authorized Client List** to *localhost*.

In DB2 Alphablox 8.4.1, specify the authorized clients in Tomcat 5.5 directly. Apache Tomcat 5.5 lets you specify a Remote Host Filter or Remote Address Filter to identify IP addresses or host names to allow or to deny. For example, to block all requests coming in except those from the local host:

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="127.0.0.1" />
```

For more information on the Remote Host Filter and the Remote Address Filter, see http://tomcat.apache.org/tomcat-5.5-doc/config/valve.html.

Set directory rights

When using DB2 Alphablox with Microsoft IIS, you should prevent users from accessing restricted files under the Web server `docroot` or the `webapps` directory of DB2 Alphablox. For details about setting directory right in IIS, see your IIS documentation.

Disable directory browsing

Internet browsers permit users to randomly access directories on the Web server. To prevent users from locating and accessing files this way unless they know the exact path and file name, you must disable directory browsing on the Web server.

Chapter 12. Extending DB2 Alphablox

This chapter describes how you can extend the built-in functionality of DB2 Alphablox with customizations in the custom calculations, DHTML client, and the Extensible User Manager.

Overview

While DB2 Alphablox has a multitude of features supporting the needs for most analytic applications, developers can also extend the platform to use custom calculations, the User Manager, and the DHTML client. Custom calculations may be required when the standard calculation APIs included are not sufficient. Extensions to the User Manager enable customization of authentication, authorization, and personalization. The DHTML client can also be customized to handle your unique needs. Following are brief sections on calculation extensions, User Manager extensions, and DHTML client extensions. Also, included is information on configuring DB2 Alphablox to handle your customizations.

Calculation Extensions

DB2 Alphablox has built-in calculation APIs supporting standard calculation requirements for most analytic applications. If you find, however, that the included functionality is not sufficient for your needs, you can create your own custom calculation extensions by following these steps:

1. Write your calculation extension to extend the `Function` class .
   a. Implement the following method:
      ```
      public double getResult(double[] variables)
      ```
      The `getResult` method returns the result of the user-defined calculation on the list of numbers which are the variables.
   b. Ensure that this file belongs to the following package:
      ```
      com.alphablox.util.calculator
      ```
   c. The name of the file must have the first letter in uppercase and the following letters in lowercase.
2. Compile your file.
3. Add the file to a JAR file.
4. Place the JAR file in the following directory:
   ```
   <alphabloxDirectory>\lib
   ```
5. Modify startup files to include this JAR in the classpath. See “Setting Class Path” on page 68 for details about how to properly set this class path.

More information about custom calculations can be found in the DataBlox Reference section of the Developer’s Reference for the DHTML Client.

User Manager Extensions

The DB2 Alphablox User Manager can be extended to support custom user authentication, authorization, and personalization requirements you might have that cannot be accomplished using DB2 Alphablox out-of-the-box functionality. For information on the available Java interfaces and server commands for extending

the DB2 Alphablox User Manger, see the “Extensible User Manager” on page 90. For information on defining a class path for access to your custom Java classes, see “Setting Class Path” on page 68.

DHTML Client Extensions

Out of the box, the DB2 Alphablox DHTML client offers a rich set of features for users. For many application pages, this functionality may be sufficient. As a result of the Blox UI model behind the DHTML client, however, developers can add customizations on their pages, use custom JavaBeans components, and create custom JSP tag libraries to further enhance analytic applications.

For details about extending the capability of DB2 Alphablox using the Blox UI Model, see the Developer’s Guide for the DHTML Client and the *Developer’s Reference for the DHTML Client*.

Configuring DB2 Alphablox to Support Custom Java Classes

As described earlier, customizations to the DHTML client, calculations capabilities, and the User Manager can be created using custom Java classes. When creating and using your custom classes, you should be aware of the following considerations:

- Custom classes, typically packaged in JAR files, can be stored in any location on your server that you prefer.

  Important: If you store your Java classes within an DB2 Alphablox installation directory, your files may be deleted during upgrades. As a result, Alphablox recommends that you store your classes in either a location outside of the DB2 Alphablox directory, or include them within your application’s directory:

  ```
  <alphlabloxDirectory>/webapps/<applicationDirectory>
  ```

- Modifications made to the DB2 Alphablox startup batch file (described in the next section, “Setting Class Path” on page 68) will be lost during DB2 Alphablox upgrades.

Setting Class Path

For DB2 Alphablox to use your custom classes, you need to add the classes directory to the DB2 Alphablox class path specified in the startup batch file. Follow the steps below based on your application server.

For Apache Tomcat implementations:

1. In a text editor, open the `<db2alphablox_dir>/appserver/bin/aas.bat` file (Windows platforms) or the `<db2alphablox_dir>/appserver/bin/aas.sh` file (Linux and UNIX platforms), where `<db2alphablox_dir>` is the directory in which DB2 Alphablox is installed.
2. Find the line that sets the class path and append a line using the same syntax to point to where your classes are. For example, if your classes are in the c:/myclasses/classes directory:

   ```
   set CLASSPATH=%CLASSPATH%;%AAS_LIB%\xerces.jar
   // Add your class path below
   set CLASSPATH=%CLASSPATH%; c:/myclasses/classes
   ```

If you are running DB2 Alphablox as a Windows service, also set the class path in the service parameters file at <db2alphablox_dir>\appserver\conf\wrapper.properties:

```
wrapper.class_path=$WRAPPER_AAS_HOME$\lib\aasserver.jar
// Add your class path below
wrapper.class_path=c:/myclasses/classes
```

3. Save your changes.
4. Restart DB2 Alphablox for the change to take effect.

For WebLogic application servers, follow these steps:

1. In a text editor, open the <db2alphablox_dir>/bin/aassetup.bat (Windows platform) or <db2alphablox_dir>/bin/aassetup.sh (Linux and UNIX platforms)
2. Find the line that sets the class path and append a line using the same syntax to point to where your classes are. For example, if your classes are in the c:/myclasses/classes directory:

   ```
   set CLASSPATH=%CLASSPATH%;%AAS_CP%
   // Add your class path below
   set CLASSPATH=%CLASSPATH%; c:/myclasses/classes
   ```

   If you are running DB2 Alphablox as a Windows service, also set the class path in the service parameters file at <db2alphablox_dir>\bin\aassetup_nt_service.bat
3. Restart WebLogic. If DB2 Alphablox is running as a Windows service, run the installSvc.cmd under <BEA>/weblogic700/server/bin again.

If you are using WebSphere, you must create a JAR file that contains your classes (you cannot have loose classes), then put the JAR file under <websphere_dir>/AppServer/lib/ext. Restart WebSphere for the change to take effect.

Chapter 13. Configuring DB2 Alphablox Properties

The **General** link under the **Administration** tab of the DB2 Alphablox home page provides links to a set of web pages for performing common administrative tasks. Note that all these tasks, as well as several others, can also be performed using console commands, as described in Chapter 19, "DB2 Alphablox Console Commands," on page 127. To access DB2 Alphablox, enter the following URL in a browser:

```
http://<servername>/AlphabloxAdmin/home
```

where <servername> represents the name of the server and port number on which DB2 Alphablox runs. This chapter includes procedures for defining the properties in which DB2 Alphablox runs and describes the DB2 Alphablox log.

DB2 Alphablox Administration Tasks

The **General Properties** section on the **General** page under the **Administration** tab provides links to the interface to perform the following DB2 Alphablox-related tasks:

- "Configuring Startup Properties"
- "Configuring System Properties" on page 72
- "Specifying the Telnet Port" on page 74
- "Configuring the DB2 Alphablox Cube Manager" on page 75

Configuring Startup Properties

The initial installation and configuration process established values for the properties on the **Startup** page. This property set represents the absolute minimum required for DB2 Alphablox to start up.

Note: Review and establish the correct values for other DB2 Alphablox properties, as described in "Configuring System Properties" on page 72, before deploying applications.

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the Administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **General Properties** section of the page, click the **Startup** link to view and modify DB2 Alphablox startup properties:

   On the **Startup** page, you can provide values for the following startup properties:

Property Name	Default Value	Description
Instance Name	`AlphabloxAnalytics`	The name for this instance of DB2 Alphablox. This name is determined during installation and cannot be changed unless you reinstall DB2 Alphablox. **Note:** This is NOT the domain name (DNS) of the DB2 Alphablox machine.

Property Name	Default Value	Description
Log File Name	`Server.log`	The name of the log file for this session of DB2 Alphablox. It resides in the `<repository>/servers/<instanceName>/logs` directory.
Command File Name	`abc.console`	The name of an optional command file. DB2 Alphablox looks for a file by this name during startup, and reads from it a series of DB2 Alphablox commands. The commands in this file use the same syntax as those entered through a DB2 Alphablox Console.
Default Message Level	INFO	The minimum (least severe) level of messages to display (and write to the log file). Valid values, in increasing severity, are: DEBUG, VERBOSE, INFO, SYSTEM, WARNING, ERROR, and FATAL. For a description of the message levels, see "Message Levels" on page 134.
Enable DB2 Alphablox Log	Yes	Whether to enable DB2 Alphablox to write messages to the log.
DB2 Alphablox Idle Duration	15 minutes	Specifies the number of minutes of DB2 Alphablox inactivity before DB2 Alphablox suspends. Suspending is a state where DB2 Alphablox uses less resources. Any request automatically resumes server activity.
DB2 Alphablox Console Default Message Level	FATAL	The minimum (least severe) level of messages to be sent to `System.out` (for DB2 Alphablox Apache Tomcat implementations, the output is displayed to the Tomcat console). Valid values, in increasing severity, are: DEBUG, VERBOSE, INFO, SYSTEM, WARNING, ERROR, and FATAL.
Load All User Objects	enabled	Load into memory all user objects. By default, the User Manager stores user objects in the DB2 Alphablox Repository. Used in conjunction with the "Keep Users Loaded (in minutes)" setting in the System properties, can be used for more efficient memory management and faster User Manager startup when you have many users.

4. Enter the required values for these properties.
5. Click the **Save** button to apply the changes.

Note: You must stop and restart DB2 Alphablox for these changes to take effect.

Configuring System Properties

After initial installation and before deploying applications, review and edit the system properties for DB2 Alphablox.

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **General Properties** section of the page, click the **System** link to view and/or modify DB2 Alphablox system properties:

In the **System** properties page, you can provide values for the following properties:

Property Name	Default Value	Description
Web Server URL Prefix	<empty>	Correct value depends on the web server in use. Do not change this value after installation without first contacting DB2 Alphablox Customer Support.
New Log Start Message Level	INFO	Establishes the least severe message level to write to a new log file.
New Log End Message Level	FATAL	Establishes the most severe message level to write to a new log file. Setting the same value on this and the **New Log Start Message Level** property (for example, ERROR) creates a log containing messages of only the specified level.
Enable Authentication	Yes	Sets whether users must authenticate before accessing DB2 Alphablox resources and DB2 Alphablox applications. When the value is "No," all users log in as guest with no requirement for a password.
Automatically Generate User Profiles	No	Determines whether user profile is automatically created when a user logs into DB2 Alphablox. When set to **Yes**, it causes any user request, regardless of the password, to be accepted and the user to be granted access to DB2 Alphablox.
Authorized Client List	<empty>	Lists the users authorized to access DB2 Alphablox. The value is a comma-separated list of IP addresses and/or host names; wildcard characters are not permitted. An empty list indicates that all users are permitted to access DB2 Alphablox. **Note:** When using web server security, it is important to set authorized clients so only the web server machine can access DB2 Alphablox.
Message History Size	100	Sets the number of messages saved to the message history area for viewing in the console. When the area fills, the DB2 Alphablox wraps to the beginning, overwriting the oldest messages (the messages are still saved in the log file).
Save on Exit	Yes	Sets whether to save the current state of DB2 Alphablox (and all its properties) when its session ends. If administrators or assemblers made changes during a session and failed to save them, leaving this value as "Yes" ensures that the changes are not lost.
Default HTML Client Theme	coleman	This sets the default theme to use when rendering application pages into HTML format. (For more information see the*Developer's Guide*.)
SMTP Server	specified during installation	Sets the name of an SMTP server so applications that use JavaMail to send mail can operate successfully. Set this to the name of a valid SMTP mail server.

Property Name	Default Value	Description
Keep Users Loaded (in minutes)	-1	Sets the number of minutes to keep user objects in memory after last instance of a user session. The default, -1, keeps users loaded indefinitely (while the DB2 Alphablox is running). In conjunction with the "Load All User Objects" setting in the Startup properties, this setting can be used for more efficient memory management and faster User Manager startup when you have many users.

1. Enter the required values for these properties.
2. Click the **Save** button to apply the changes.

Note: You must stop and restart DB2 Alphablox for property changes to take effect.

Specifying the Telnet Port

Perform the following to specify ports in which DB2 Alphablox listens on:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **General Properties** section of the page, click the **Telnet Console** link to view and modify DB2 Alphablox ports.

   On the **Telnet Console** page, the following properties values can be modified:

Property Name	Default Value	Description
Telnet Console Port	23	The port on which the telnet version of the DB2 Alphablox console operates. If DB2 Alphablox is the default telnet device, it is not necessary to enter a port number when accessing the telnet consoles. Set this value to 0 (zero) to disable telnet access to DB2 Alphablox Console.
Telnet User Name	admin	The name of a user who can run a telnet session.
Telnet Password	password	The password for the telnet user name.
Confirm Telnet Password		When you enter a password in the **Telnet Password** text box, you must confirm it in this text box.
Telnet Timeout	15 minutes	Specifies the number of minutes before a telnet session to DB2 Alphablox times out. When a telnet session times out, the session is terminated.
Restart Console Manager	not checked	When checked, restarts the console manager when you press the **Save** button.

4. Enter the required port numbers.
5. Click the **Save** button to apply the changes.

Note: You must stop and restart the corresponding service managers for these changes to take effect. To restart the service manager(s), check the box below the port(s) that you have changed and click the **Save** button.

Configuring the DB2 Alphablox Cube Manager

Clicking the **DB2 Alphablox Cube Manager** link brings up a page that allows administrators to limit the number of DB2 Alphablox cubes that can be loaded in the server. Each DB2 Alphablox cube can potentially use a large amount of system resources, so administrators might want to limit the number of cubes based on the memory available on the machine in which DB2 Alphablox runs.

To limit the number of DB2 Alphablox cubes on the system:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **General Properties** section of the page, click the **DB2 Alphablox Cube Manager** link to view and/or modify DB2 Alphablox system properties:
4. Check the **Maximum Cubes** box.
5. Enter the maximum number of DB2 Alphablox cubes in the **Maximum Cubes** text box.
6. Click the **Save** button to apply the changes.

For detailed information about creating and managing DB2 Alphablox cubes using the DB2 Alphablox Cube Server, see the *DB2 Alphablox Cube Server Administrator's Guide*.

Custom Property Definitions

Custom properties can be used to associate user login details with DB2 Alphablox user profiles. For details regarding the user profile, see Chapter 8, "User Definitions," on page 47. DB2 Alphablox provides an example custom properties file to make this easier for you.

Valid values for the property, as well as its default value, are taken from the property definition. The user property is available to developers programmatically, for example, using the RepositoryBlox `getUserProperty()` method.

Note: When defining a Custom Property that requires spaces in the property value, use ` ` instead of typing spaces. Typed spaces are not recognized, and are removed from the property value.

Defining a New User Property

Perform the following to define a new user property:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **Custom Properties** section of the page, click the **User Definitions** link. The **User Definitions** page appears.
4. Click the **Create** button. The **Create User Custom Properties** page appears.
5. Provide a name (required) in the **Property Name** text box and a description in the **Display Label** text box. The description appears on the application definition on the pages that display from the **Applications** link under the **Administration** tab.

   Note: If you are using LDAP-based User Manager or other external user manager (such as NTLM) that has its own editor, you will need to

select between External Repository and Local Repository under **Property Name**. Click the External Repository radio button and select from the drop list in order for DB2 Alphablox to load the custom property.

6. Click the **Multiple Select** button if you want the property to accept a multiple values.
7. Select the **User Access** level from the drop list. (Administrative privileges are always required to define, change, or delete Application properties.)
   - **Hidden:** The property does *not* appear on the New User and Group, Change User and Group, and the Manage User Profile pages. Its value can be changed in an application (for example, through JavaScript) or through this page only.
   - **Visible:** The property and its default value appear on the New User and Group, Change User and Group, and the Manage User Profile pages, but the user cannot change the value.
   - **Edit:** The property and its default value appear and can be changed on the New User and Group, Change User and Group, and the Manage User Profile pages. If the property has more than one valid value, the user selects a value from a list. If the property has no valid value list, the user can enter a value.
8. Enter a **Default Value** for the property, if any.
9. In the **Value List** text box, enter an optional comma-separated string of valid property values. Values entered here are used to populate the list from which users make their selection. If no values are entered, and if **User Access** is set to **Edit**, a text box appears on the respective pages in which users can enter a value.
10. Click the **Save** button to save the new property and return to the **General** page.

Changing a User Property

Perform the following to change a user property:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **Custom Properties** section of the page, click the **User Definitions** link. The **User Definitions** page appears.
4. Select the definition to change and click the **Edit** button. The **Edit User Custom Property** page appears with its current values appearing in the input fields.
5. Make the necessary changes.
6. Click the **Save** button to make the changes and return to the **General** page.

Deleting a User Property

Perform the following to delete a user property:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **Custom Properties** section of the page, click the **User Definitions** link. The **User Definitions** page appears.
4. Select the property definition you want to delete from the list of existing properties.

5. Click the **Delete** button to delete the property and return to the **General** page.

   Note: Deleting a property definition is permanent; you cannot recover a deleted property definition.

Defining a New Custom Application Property

Perform the following to define a custom application definition:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **Custom Properties** section of the page, click the **Application Definitions** link. The **Application Definitions** page appears.
4. Click the **Create** button. The **Create Application Custom Property** page appears.
5. Provide a name (required) in the **Property Name** text box and a description in the **Display Label** text box. The description appears on the application definition on the pages that display from the **Applications** link under the **Administration** tab.
6. Click the **Multiple Select** button if you want the property to accept a multiple values.
7. Select the **User Access** level from the drop list. (Administrative privileges are always required to define, change, or delete Application properties.)
   - **Hidden:** The property does *not* appear on its respective pages. Its value can be changed in an application (for example, through JavaScript) or through this page only.
   - **Visible:** The property and its default value appear on its respective pages, but the user cannot change the value.
   - **Edit:** The property and its default value appear and can be changed on its respective pages. If the property has more than one valid value, users select the desired value from a list. If the property has no valid value list, users can enter a value.
8. Enter a **Default Value** for the property, if any.
9. In the **Value List** text box, enter an optional comma-separated string of valid property values. Values entered here are used to populate the list from which users make their selection. If no values are entered, and if **User Access** is set to **Edit**, a text box appears on the respective pages in which users can enter a value.
10. Click the **Save** button to save the new property and return to the **General** page.

Changing an Application Property

Perform the following to change an application property definition:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **Custom Properties** section of the page, click the **Application Definitions** link. The **Application Definitions** page appears.
4. Select the definition to change and click the **Edit** button. The **Edit Application Custom Property** page appears with its current values appearing in the input fields.

5. Make the necessary changes.
6. Click the **Save** button to make the changes and return to the **General** page.

Deleting an Application Property

Perform the following to delete an application property definition:

1. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
2. Click the **Administration** tab. The **General** page appears.
3. In the **Custom Properties** section of the page, click the **Application Definitions** link. The **Application Definitions** page appears.
4. Select the property definition you want to delete from the list of existing properties.
5. Click the **Delete** button to delete the property and return to the **General** page.

   Note: Deleting a property definition is permanent; you cannot recover a deleted property definition.

Note: In this figure, everything below the **Allow User To Edit Profile** drop list appear because of the contents of the *UserPropDesc.properties* file.

Creating and Managing Comments Collections

DB2 Alphablox can be used to manage user comments to be available on data cells in multidimensional grids. The following sections describe how to use the Comments Management page, available under the DB2 Alphablox Administration tab to configure a Comments Collection for use with a particular data source. Comments are viewable in the DHTML client on supported web browsers.

Note: Comments collections requires additional configuration steps to work with the Microsoft JDBC Driver for SQL Server, which is an option included on WebSphere. Contact DB2 Alphablox Customer Support for assistance.

Accessing the Comments Management Dialog

To access the Comments Management page, which opens in a new window, follow these steps:

1. Open the DB2 Alphablox home page.
2. Click the Administration tab.
3. Under Runtime Management in the navigation menu on the left, click the Comments Management link.
4. The Comments Management dialog should appear in a new browser window.

   Note: To use the Comments Management Dialog, you need to have rights for creating and dropping relational tables. For using the CommentsBlox API in developing custom commentary applications, you may need rights for selecting, inserting, updating, deleting, creating, and dropping tables.

Defining And Accessing a Data Source

To define a data source to be available for storing your comments collections, follow these steps:

1. Under the Datasource heading, click on the Name selection list and select the defined DB2 Alphablox data source where you want your comments collections to be stored. [Note: To learn more about defining data sources in DB2 Alphablox, see Data Source Definitions]
2. In the **Username** and **Password** entry fields, enter a login and password for a user authorized to create tables in the data source. [Note: If these fields are blank, the default datasource user name and password, if defined in the DB2 Alphablox data source definition, will automatically be submitted upon connecting.]
3. Press the Connect button. If the defined data source and user information are valid, a connection to your data source should occur and the Collection section on the Comments Management dialog will become enabled.

Defining Comments Collections

Before users can begin adding comments to data cells in a grid, you must first create a comments collection to store related comments.

Note: Dimensions in your comments collection may be added or removed, as long as no comments exist in the collection.

1. To create a new comments collection, follow these steps:
2. Click the Create button below the Collection list box. On the right side of the browser window, three new sections will appear.
3. In the Name section, enter the name of your comments collection. [Note: Names must not include spaces and must conform to any naming requirements for tables in the database being used to store comments.]
4. In the Fields section, you can add or delete fields in your comment collection. By default, three required fields are included: the Author, Timestamp, and CommentText fields. You can add additional fields, defining their names, and if desired, descriptions for each field you define. An optional CellValue field can also be added. The CellValue field will result in the current cell value for the selected cell to be automatically added to the comment associated with that cell.
5. In the Dimensions section, you must select a multidimensional data source from the list of defined DB2 Alphablox data sources. A list of dimensions available in the selected cube should appear in the text box below the listed data source.
6. Select which dimensions you want users to be able to add comments to. To select more than one dimension, you must press the Control key and click on each additional dimension.

   Note: Dimensions selected affect the scope of the added comments by your users, as shown in the following two examples:

   - Example 1: If you have a cube with three dimensions (Year, Product, and Region), and select only two of the dimensions (Product and Year) in the Dimensions list, then comments added to any member of the Region dimension (unselected in the Dimensions list) will be displayed in all other Region members.
   - Example 2: If you have a cube with three dimensions (Year, Product, and Region), and select all three dimensions in the Dimensions list, then comments will appear only in the cell to which the comments were added.
7. Press the Save button. At this point, the name of the newly defined collection should appear in the Collection listing.

Comments Collections Using Microsoft SQL Server or Sybase Databases

If you plan to use Microsoft SQL Server or Sybase databases to store comments that may include non-ASCII characters, commonly found in foreign languages, you will need to perform the following steps:

1. For each comments collection you define, you need to change the data types of the columns in the automatically-generated tables in from `varchar` to `nvarchar`.

   The following five tables are shared by all of the defined comments collections using the data source:

   - Collections: `CollectionName`
   - Comments: `CommentText`
   - Context: `ContextName`
   - Dimensions: `DimName`
   - FieldDefinitions: `descr`, `FieldName`
   - FieldValues: `fieldvalue`

   Also, for each comments collection, a unique table is created:

   - ABXMBRT_<collectionName>

   where <collectionName> is the name you specified for the comments collection. You need to change the data type to `nvarchar` for the following column:

   - ABXMBRT_<collectionName>: MemberName
2. All JSP pages within your application should have the Unicode character set specified in a JSP page directive on each page, like this:

   ```
   <%@ page contentType="text/html; charset=UTF-8" %>
   ```

Displaying Comments Collection Definitions

To display the definition of an existing collection, follow these steps:

1. When you are connected to a defined data source, a list of predefined comments collections appears in the Collection text box and the Create, Display, and Delete buttons are enabled. If you are not connected, first follow the steps above under Defining and Accessing a Data Source to connect to a data source.
2. Click on one of the comments collections listed in the Collection text box, then press the Display button. The collection definition should appear in the right half of the window.

Deleting Comments Collections

To remove a defined comments collection, follow these steps:

1. Connect to a defined data source (see Defining and Accessing Data Sources above).
2. In the Collection text box, click on the name of the comments collection you want to remove.
3. Press the Delete button. The collection name should be removed from the listing.

Adding and Displaying Comments

After a comments collection has been defined, users can add comments to the data cells of grids that have been enabled for comments. When a user right-clicks on a data cell that is comment enabled, a menu appears. In the Comments submenu, users can choose to either Add Comments or Display Comments.

For more information on CommentsBlox, see the CommentsBlox section of either the *Developer's Reference*.

Creating a Remote PDF Processor

For performance, memory management, or to share PDF processing for multiple DB2 Alphablox hosts, you may decide to run your PDF engine on an remote dedicated server.

Configuring Remote PDF Processor

To use a remote PDF processor instead of the internal server:

1. Create a root directory on your remote server called *AlphabloxPDFRemote*.
2. In this root directory, create two new directories: *lib* and *repository*.
3. Inside of the *repository* directory, create two directories: *font* and *theme*.
4. From the *db2alphablox_dir/lib/* directory of your DB2 Alphablox installation, copy the following JAR files into the remote server's *lib* directory:
   - bforeport.jar
   - pdfserver.jar
   - xalan.jar
   - xercesImpl.jar
   - xml-apis.jar
   - icu4j_3_4_1.jar
5. Copy all of the files from the alphablox *db2alphablox_dir/repository/theme* directory into the *theme* (*repository/theme*) directory on the remote server.
6. Copy all of the files from the alphablox *db2alphablox_dir/repository/font* directory to the *font* (*repository/font*) directory on the remote server.
7. From the *db2alphablox_dir/bin/* directory of your DB2 Alphablox installation, copy the StartPDFRemoteServer script file (*StartPDFReportServer.bat* for Windows systems or *StartPDFReportServer.sh* for Linux and UNIX systems) to the *AlphabloxPDFRemote* directory on the remote server.
8. Edit the StartPDFRemoteServer script file, making the following changes:
   - Change the *LIB* setting to point to the remote server *lib* directory.
   - Make sure the JAVA setting points to a valid JRE or JDK.
   - Change the *LISTEN_PORT* setting, if necessary
   - Make sure that the script starts in the current directory (by default the script file goes up one directory).
9. Run the batch file. The remote PDF processor should now be available.
10. To use it with your DB2 Alphablox applications, follow the steps in the next section to configure DB2 Alphablox to use the remote server.

Configuring Remote PDF Reports Administration

PDF reports processing is administered by using the PDF Reports Runtime Management settings in the DB2 Alphablox Admin Pages. To configure a remote server, follow these steps:

1. Open the DB2 Alphablox Admin Pages, then click on the Administration tab.
2. On the left side of the page, under Runtime Management, click on the PDF Reports (DHTML) link to open the PDF Reports dialog.
3. To configure a remote PDF server, make the following changes:
   a. Click on the Edit button.

b. Change the Server setting from `Use Internal Server` to `Use PDF Report Server`.
c. Define the Host Name and Host Port for the remote server.
d. Choose a Compression option, if desired, or leave it set to the default of `No Compression`. The options are based on the compression settings defined in the JDK. In reality, there is little difference in compression between the low and high settings. The low setting significantly compresses the files, yet the highest setting offers only minimally greater compression than the lowest setting.
e. Save your changes. If you have successfully configured the remote PDF server, the Status should indicate `Available`.

DB2 Alphablox Log Files

The DB2 Alphablox logs various DB2 Alphablox events and errors to a log file. Every message that is written to the DB2 Alphablox console is also written to the log file. The log files remain until they are deleted or moved, and help to keep a history of the activity on DB2 Alphablox. The history is useful for administrative auditing purposes and for use by DB2 Alphablox Customer Support.

Log File Rollover Interval Settings

By default, DB2 Alphablox creates a new log file every day and also every time the application server starts up. As an administrator, you can choose to change the log file rollover settings in either the DB2 Alphablox Admin Pages or by editing the `server.properties` file.

To set the interval in the DB2 Alphablox Admin Pages, open a browser to the Administration tab in the DB2 Alphablox Admin Pages, then click on System under General Properties. By default, the New Log Rollover Interval is set to rollover each day. You can edit this setting to never rollover, or to specified minutes, hours, days, weeks, or kilobytes.

To set the log file rollover interval in the `server.properties` file, open the `server.properties` file, located in the following directory:

`<repository_dir>/servers/<instanceName>/`

Find the setting for `LogRollOverInterval` and change it to the desired setting using the following values:

Set this to one of the following values (default is `1D`):

`#M` - Rollover after number (#) of minutes

`#H` - Rollover after number (#) of hours

`#D` - Rollover after number (#) of days

`#W` - Rollover after number (#) of weeks

`#K` - Rollover after number (#) of K bytes written

`NONE` - Never rollover

Log File Names

The log file is located in the following directory:

`<repository_root>/servers/<instance_name>/logs`

The default name of the active log file is `Server.log`. The name is configured on the page accessed by clicking the **Administration** tab, **General Properties** section, **Startup** link.

DB2 Alphablox creates a new log file each time it starts up. When DB2 Alphablox starts up, the previous log file is renamed to the following:

`Server<timestamp>.log`

where <*timestamp*> is a time from the last entry in the log. The timestamp has the following format:

`YYMMDD_HHmmSS`

For example, if the DB2 Alphablox was last shut down at 6:22:35 PM on June 3, 2000, when it next starts up the old log file will be renamed *Server000603_182235.log* and a new log file named Server.log will be created to store the current log entries. Note that the timestamp is based on a 24 hour clock (in this example, the *18* represents 6 PM). If the name of the log file is changed on the **Administration** pages, the timestamp is appended to the new name.

Note: The active log file does not have the timestamp appended to its name; its name is exactly what is specified on the **Administration** pages.

Managing the Log Files

Old log files can remain around indefinitely without causing any problems, but you might want to establish a process to archive them after a certain amount of time. The disadvantage of keeping old log files around is that they take up disk space. How much disk space they occupy depends on the activity on DB2 Alphablox and the default message level set on the Administration pages. If the message level is set to DEBUG or VERBOSE, the log file can grow large fairly quickly.

Since the old log files are renamed with a timestamp as part of their filename, you can create utilities to help manage the log files. For example, you can create a script to delete or move the log files once they become older than three months.

Chapter 14. User Manager and personalization (Alphablox 8.4.1)

The Personalization Manager manages personalization capabilities for customizing application content. In DB2 Alphablox 8.4.1, the security functionality and the personalization functionality provided in User Manager are separated into two models.

DB2 Alphablox 8.4.1 provides a security model that is based on Servlet 2.4 and Java Authentication and Authorization Service standards. This security model is discussed in "Security model in Alphablox 8.4.1" on page 57.

For personalization, the new Personalization Manager provides an interface to notify DB2 Alphablox when a user or a group is created, loaded, or deleted. The Personalization Manager also lets you specify and modify properties on a user or group, modify users and groups that are contained in a group, and create your own user or group and add it to the Alphablox Repository.

Personalization Manager

The new Personalization Manager in DB2 Alphablox 8.4.1 allows you to read user and group properties from an external user repository. The Personalization Manager interface in the com.alphablox.personalization package is notified when a user or group object is created, loaded, or deleted. This PersonalizationManager object also lets you augment DB2 Alphablox users and groups with properties that are defined in a user repository.

If you are using a Personalization Manager, add a *config.xml* file in the *alphablox_dir/repository/servers/instance* directory. When DB2 Alphablox starts up, it does the following:

1. Looks for the *config.xml* file in the repository.
2. Creates an instance of the PersonalizationManager object.
3. Sets the supplied properties.
4. Assigns the PersonalizationManager object to the User Manager service.

The following is an example of the `<Server>` tag:

```
<Server>
  <Service name="User Manager">
    <PersonalizationManager
      className="myPackage.CustomPersonalizationManager"
    />
  </Service>
</Server>
```

The Personalization Manager has the following characteristics and behaviors:

- User or group properties in your external user repository are available through the DB2 Alphablox Admin Pages only after you define the same custom properties through the DB2 Alphablox Admin Pages.
- Once the same properties are defined as custom properties, DB2 Alphablox only performs only read operations. Changes made through the DB2 Alphablox

Admin Pages are local to DB2 Alphablox in memory only and are not written back to the external user repository. These values are overwritten the next time the user object is loaded.

- If you need to change the connection information or other attributes in the *config.xml* file, you need to restart Apache Tomcat, which restarts DB2 Alphablox.

Accessing JNDI user and group properties

The information provided in this section applies to DB2 Alphablox 8.4.1.

User and group properties in your JNDI-based repository can be made available to DB2 Alphablox and accessible through the RepositoryBlox API. To make these properties available, you need to specify the properties in the *alphablox_dir/repository/servers/instance/config.xml* file:

1. Add a *config.xml* file in your *alphablox_dir/repository/servers/instance* directory:

```
<Server>
    <Service name="User Manager">
     <PersonalizationManager
    className="com.alphablox.personalization.jndi.JNDIPersonalizationManager"
        connectionURL="ldap://machineName:port"
        userBase="ou=People,dc=company,dc=com"
        userSearch="(&amp; (uid={0}) (objectclass=person))"
      userProperties="Mail, Phone, Fax"
         groupBase="ou=Groups,dc=company,dc=com"
        groupSearch="(&amp; (cn={0})(objectclass=groupofuniquenames))"
      groupProperties="Description"
      />
    </Service>
</Server>
```

   The following table includes the attributes that are available for accessing JNDI user and group properties.

Attribute	Description
connectionURL	Required. The JNDI connection URL. Example: ldap://machine1.ibm.com:10456
contextFactory	Optional. The context factory to use when creating the initial directory context. The default is com.sun.jndi.ldap.LdapCtxFactory.
groupBase	Required. The JNDI base to use when searching for groups.
groupProperties	Required. The user properties to assign from the JNDI repository to the DB2 Alphablox user object. This property overrides any custom user properties defined in the DB2 Alphablox Admin Pages.
groupSearch	Required. The JNDI search string to use when searching for a group. The group search might contain the MessageFormat replacement syntax {0}, which will be replaced by the unique name of the group being searched.
groupSubtree	Optional. Determines if subtrees of the groupBase are used when searching for a group.
password	Optional. The password to use when connecting to the connection URL.

Attribute	Description
userBase	Required. The JNDI base to use when searching for users.
userName	Optional. The user name to use when connecting to the connection URL.
userProperties	Required. The user properties to assign from the JNDI repository to the Alphablox user object. This property overrides any custom user properties defined in the DB2 Alphablox Admin Pages.
userSearch	Required. Sets the JNDI search string to use when searching for a user. The user search might contain the MessageFormat replacement syntax {0} which is replaced by the unique name of the user being searched.
userSubtree	Optional. The subtrees of the userBase to use when searching for a user.

To access the values of the user properties in your JSP files using the RepositoryBlox API:

1. Add the same custom user properties by using the DB2 Alphablox Admin Pages. Go to the **Administration** tab > **General properties** page > **Customer Properties** section. Detailed steps are described in "Custom Property Definitions" on page 75.
2. In your JSP files, access this custom property and its value through the RepositoryBlox API:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
...
<blox:repository id="myRepository" />
...
<html>
<head>
    <blox:header />
</head>
<body>
Your region is: <%= myRepository.getUserProperty("region") %>
...
```

3. Restart Apache Tomcat, which restarts DB2 Alphablox, for your *config.xml* file changes to take effect.

Chapter 15. User Manager (Alphablox 8.4)

This chapter describes how to configure and use the DB2 Alphablox User Manager functionality for LDAP integration and how to extend the Extensible User Manager personalization engine to implement custom security.

DB2 Alphablox User Manager Overview

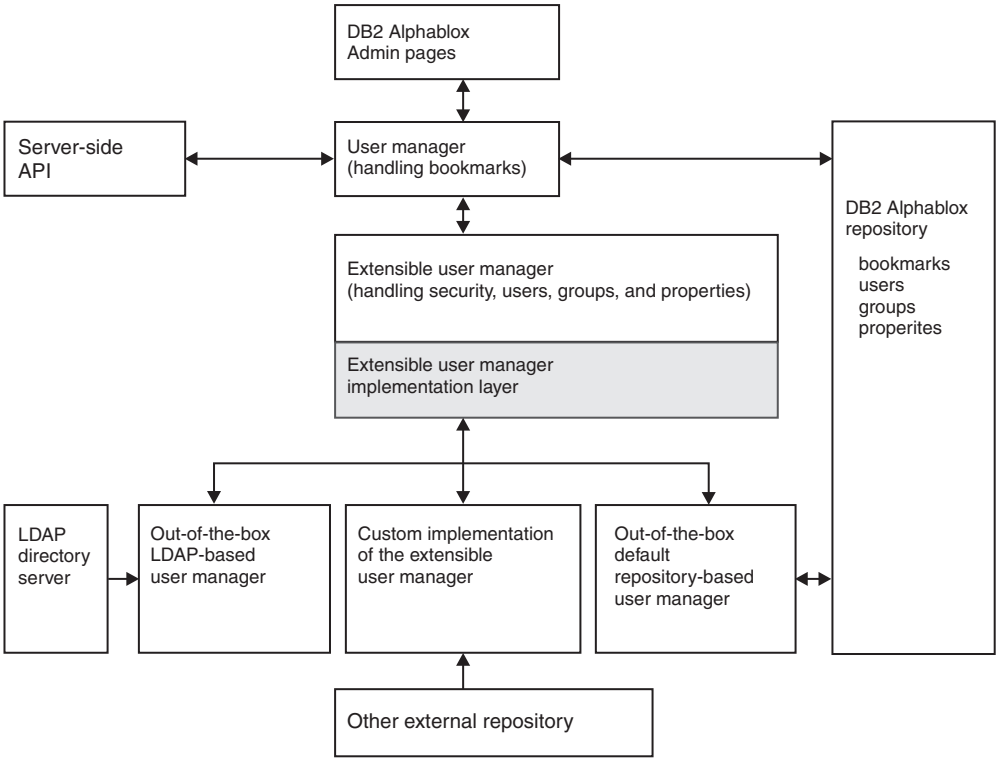
Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

The DB2 Alphablox User Manager manages user authentication and authorization as well as provides personalization capabilities for customizing application content. By default, DB2 Alphablox uses the DB2 Alphablox Repository and the J2EE Security API to manage user and group information. Two J2EE security methods (`isUserInRole()` and `getUserPrinciple()`) are used to identify who the user is and whether the user is in a specific role.

DB2 Alphablox also provides an out-of-the-box Lightweight Directory Access Protocol (LDAP) integration solution. This solution allows DB2 Alphablox to authenticate and authorize the users by using an LDAP directory server to recognize DB2 Alphablox users, groups, and custom properties.

The User Manager is built on top of the personalization engine called Extensible User Manager. For environment where custom security is desired, the Extensible User Manager personalization engine provides interfaces that let you extend either of the two out-of-the-box security solutions (DB2 Alphablox Repository and LDAP). Or you can plug in another external user manager such as NTLM or some existing Enterprise JavaBeans (EJBs).

The following diagram shows the architecture of User Manager and its Extensible User Manager personalization engine:



Notice from the diagram that:

- Users, groups, and related property information are accessible programmatically through server-side API through the RepositoryBlox.
- DB2 Alphablox provides two out-of-the-box solutions for security and personalization: DB2 Alphablox Repository-based User Manager and LDAP-based User Manager.
- The default Repository-based User Manager reads from and writes to the DB2 Alphablox Repository.
- The out-of-the-box LDAP-based User Manager only performs READ operations to the LDAP server.
- Through the Extensible User Manager API, you can use other external repository such as NTLM or some existing Enterprise JavaBeans (EJBs).
- Regardless of the repository used (DB2 Alphablox, LDAP, or other external sources), you can always access the user properties through the RepositoryBlox API.

Extensible User Manager

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

At the heart of the DB2 Alphablox User Manager lies the Extensible User Manager personalization engine. It gives you the ability to:

- customize the out-of-box user authentication and/or authorization,
- plug in an external user manager to access user/group information from other sources outside DB2 Alphablox Repository, or
- use security settings other than J2EE Security API

The two out-of-the-box DB2 Alphablox security and personalization implementations—DB2 Alphablox Repository-based User Manager and LDAP-based User Manager—are both built on this engine. To use the LDAP-based User Manager, some configuration steps are required. These steps are described in "LDAP-Based User Manager" on page 91. To implement custom security, or to customize either of the two out-of-the-box security implementations, you can extend the Extensible User Manager. The details on extending the Extensible User Manager interface are described in "Extensible User Manager Interfaces" on page 94.

LDAP-Based User Manager

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

DB2 Alphablox provides an out-of-the-box implementation of the Extensible User Manager to use an LDAP directory server to recognize DB2 Alphablox users, groups, and custom properties:

- DB2 Alphablox only performs READ operations to the LDAP server.
- Administrators can only create DB2 Alphablox users or groups that are already defined in the LDAP Server.
- Administrators can not rename or modify memberships from the **Administration** tab on the DB2 Alphablox home page.
- Deleting users or groups via the **Administration** tab on the DB2 Alphablox home page does not remove the items from the underlying repository nor affect the LDAP server.
- The "admin" user is no longer available by default unless it is an LDAP user.
- User "guest" and Group "public" are available even if they do not exist in LDAP.

Configuring DB2 Alphablox to Use LDAP User Manager

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

The LDAP-based User Manager can be used with WebSphere, WebLogic, and Tomcat application servers. Follow these steps to configure DB2 Alphablox to take advantage of the out-of-the-box integration with your LDAP directory servers:

1. Use the `ExtUserManager` telnet console command to set the server to use LDAP, as described below in "Setting LDAP-based User Manager Properties" on page 92.
2. Make sure that the server property `autoCreateUsers` is set to `true` by using the telnet console command:

   ```
   set server autoCreateUsers true
   ```

   Note: The default value of `autoCreateUsers` is `false`. To enable DB2 Alphablox to automatically create the users if they are successfully authenticated against the LDAP server, you must set the property to `true`.

   Note: The External User Manager must have the Group "public" and the User "guest" defined (these can be just dummy instances).

Additional Requirements for Apache Tomcat Configurations

1. Create `AlphabloxAdministrator` group in LDAP Directory Server.

2. Add at least one user to the AlphabloxAdministrator group or add an existing group that contains at least one user as a subgroup to the AlphabloxAdministrator group.

Setting LDAP-based User Manager Properties

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

1. Open a telnet console connection to DB2 Alphablox.

   Note: If you access a telnet console through the **Administration** tab on the DB2 Alphablox home page, you must restart DB2 Alphablox before changes take effect. By using a standard telnet console, you will not need to restart DB2 Alphablox.

2. Enter the following command:

   ```
   ExtUserManager setToDefaults ldap "<ldapProps>"
   ```

   where <ldapProps> are semicolon-separated lists of property-value pairs. For example:

   ```
   "host:localhost;port:389:admin:cn=DirectoryManager;
   password:password;debug:false;base:dc=alphablox,dc=com"
   ```

   Note: The above should be entered as a single line with no line breaks.

After the ExtUserManager command has been executed, DB2 Alphablox will switch from the default DB2 Alphablox Repository-based User Manager to the LDAP-based User Manager.

Note: This switch will result in disconnecting all current users.

The following table lists all required and optional properties that can be defined in the <ldapProps> string:

Property	Description	Example
host	LDAP directory server host	host:localhost
port	LDAP directory server port	port:389
admin	LDAP directory server administrator user name	admin:cn=Directory Manager
password	LDAP directory server administrator password	password:myPassword
base	LDAP directory server search base	base:o=myCompany.com
debug	[Optional] Debug mode setting; default is false	debug:true

Tip: The search base in LDAP is used to specify the starting point for the search. It points to a distinguished name of an entry in the directory hierarchy so the search can be more effective.

Accessing Custom User Properties

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

You can define custom user properties to provide further personalization to your applications. Once a custom user property is defined, you can assign a different value to the property for each user and then programmatically access these

property values using the RepositoryBlox's API. For an LDAP-based User Manager or other external user managers, DB2 Alphablox will only load custom user properties that are marked as external properties.

Custom user properties can be defined using the **Administration** tab on the DB2 Alphablox home page. On the custom property definition page, make sure you check the box for **External Property**. For detailed steps, see "Custom Property Definitions" on page 75. Once a custom user property is marked as external, when information on the value of an external custom user property is requested (through a `RepositoryBlox.getUserProperty("myExternalUserProp")` call, for example), DB2 Alphablox User Manager will automatically go to the external source to fetch the value.

Runtime Behavior

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

The LDAP-based User Manager has the following runtime behaviors:

- DB2 Alphablox will not auto create groups at runtime. Administrators must explicitly register LDAP Group via the **Administration** tab on the DB2 Alphablox home page.
- DB2 Alphablox will auto create users if they are successfully authenticated against the LDAP server and if the autoCreateUsers server property is set to true via the telnet command (described in "Configuring DB2 Alphablox to Use LDAP User Manager" on page 91).
- DB2 Alphablox will only load LDAP user and group properties that is defined as external custom properties.

Extensible User Manager Telnet Console Command

The `ExtUserManager` telnet console command lets you instruct the DB2 Alphablox to use a different class for managing users or groups or to use an

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).
external repository.

The general syntax for ExtUserManager is as follows:

```
ExtUserManager <Property> <Value>
```

where `Property` can be:

Possible Property	Description	Value
`umclassname`	The user manager class name	The value must be a valid class name available in the class path that implements the `IUserManager` interface, otherwise an error will occur. See "Custom Security Implementations" on page 95 for more detail.
`userclassname`	The user class name	The value must be a valid class name available in the class path that implements the *IUser* interface, otherwise an error will occurs. See "Custom Security Implementations" on page 95 for more detail.

Possible Property	Description	Value
groupclassname	The group class name	The value must be a valid class name available in the class path that implements the *IGroup* interface, otherwise an error will occur. See "Custom Security Implementations" on page 95 for more detail.
customstartupproperty	A custom startup property that can be used by developers in their code to read in a value as the User Manager starts up.	The value of this property.
setToDefaults	The default repository to use	See the following section on "Setting the Default Repository."

Note: DB2 Alphablox must be stopped and restarted if a new custom startup property is set.

Setting the Default Repository

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

To set all properties to use the default LDAP-based User Manager, use the following telnet console command:

```
ExtUserManager setToDefaults ldap <ldapProps>
```

where `ldapprops` is a property that contains all the information needed to connect to LDAP. There is no need to restart DB2 Alphablox, but all current running users will be disconnected. For details on the steps to configure to use LDAP-based User Manager and on the syntax of the `ldapprops` property, see "LDAP-Based User Manager" on page 91.

To reset all properties to use the default DB2 Alphablox Repository-based User Manager, use the following telnet console command.

```
ExtUserManager setToDefaults repository
```

This command will instruct the User Manager to stop and start again to get the latest user information from DB2 Alphablox Repository. There is no need to restart DB2 Alphablox, but all current running users will be disconnected.

Removing Users and Groups No Longer in the External User Repository

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

The following command instructs Extensible User Manager to remove from the repository users and groups that are no longer in the external user repositories (such as LDAP or NTLM):

```
ExtUserManager clean
```

Extensible User Manager Interfaces

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

The Extensible User Manager framework consists of three main interfaces: `IUserManager`, `IUser`, and `IGroup`. The following table lists the purpose of and methods in each of the interfaces. It also lists the class implementing the specific interface.

Interface	Description	Implementing Class
IUserManager	Responsible for creating and finding the users and groups at runtime.	`AbstractUserManager`
IUser	Providing a read-only view for a user in the underlying repository for authentication, authorization, and user properties.	`AbstractUser`
IGroup	Providing a read-only view for membership information and group properties in the underlying repository	`AbstractGroup`

The interfaces and implementing classes are in the `com.alphablox.personalization` package. Javadoc for this package is available from the following directory:

`<db2alphablox_dir>/system/documentation/javadoc/blox/index.html`

where `<db2alphablox_dir>` is the directory in which DB2 Alphablox is installed.

Custom Security Implementations

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

The Extensible User Manager has three DB2 Alphablox properties for identifying which user, group, and user manager class to use for authentication and authorization:

- `umclassname`

  The default value for DB2 Alphablox Repository-based User Manager is `com.alphablox.personalization.repository.RepUserManager`. The default for LDAP-based User Manager is `com.alphablox.personalization.ldap.LDAPUserManager`.
- `userclassname`

  The default value for this property for DB2 Alphablox Repository-based User Manager is `com.alphablox.personalization.repository.RepUser`. The default for LDAP-based User Manager is `com.alphablox.personalization.ldap.LDAPUser`.
- `groupclassname`

  The default value for this property for DB2 Alphablox Repository-based User Manager is `com.alphablox.personalization.repository.RepGroup`. The default for LDAP-based User Manager is `com.alphablox.personalization.ldap.LDAPGroup`.

You can write your own user, group, or user manager class and then use the telnet console command `ExtUserManager` to set the new value for these DB2 Alphablox properties to point to your user/group/user manager class.

The following table shows the class you need to extend if you need to customize parts or all of the out-of-the-box security scheme.

Goal	Solution
Custom authentication or authorization in the default DB2 Alphablox Repository-based User Manager	Extending the `AlphabloxUser` class
Custom authentication or authorization in the default LDAP-based User Manager	Extending the `ldapUser` class
Single sign-on	Implementing the `getPassword()` method in the `AlphabloxUser` class
Reading users/groups properties from repository other than DB2 Alphablox or LDAP	Extending the `AlphabloxUser` or `AlphabloxGroup` class
Reading dynamic group membership information from repository other than DB2 Alphablox or LDAP	Extending the `AlphabloxUser` or `AlphabloxGroup` class
NTLM support	Full implementation of the Extensible User Manager interfaces

Note: For DB2 Alphablox to use your classes, you need to add the classes directory to the DB2 Alphablox class path specified in the startup batch file. For details on how to set the class path, see "Setting Class Path" on page 68.

Single Sign-On

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

To implement single sign-on when your DB2 Alphablox data sources are configured to use DB2 Alphablox user names and passwords, you can use the `getPassword()` method in the `AlphabloxUser` class to get the base64–encoded password. The following table shows the solution for single sign-on in different situations:

User Manager	Data Sources with LDAP?	Authentication Method
LDAP-based User Manager	Yes	Basic DB2 Alphablox authentication (browser-challenged authentication); No action needed
LDAP-based User Manager	No	Use `getPassword()`
Other repository	Either	Use `getPassword()`

Custom Security Examples

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

The following are three examples of extending the Extensible User Manager for custom security. Each of the examples shows:

- how to extend the default user, group, or user manager class used by the default DB2 Alphablox Repository-based User Manager, and then
- set the new class name using the telnet `ExtUserManager` command.

For a complete example of a full implementation of external user manager, see the Simple User Manager example in the following directory:

`<db2alphablox_dir>/system/documentation/admin/Examples/`

The Java source files are provided.

Example 1: Setting up DB2 Alphablox to Use an External User Manager

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

This example demonstrates how you can customize the default DB2 Alphablox Repository-based User Manager. In this case, you may want to read a list of custom properties from a file when the User Manager starts up. The steps involved include:

1. Write a custom user manager class in a package (in this example, the class is called `MyUserManager` in a package called `com.myCompany.user`)
   - The class extends the `AlphabloxUserManager` class used in DB2 Alphablox Repository-based User Manager
2. Set up the new value for the `umclassname` server property, using the following telnet console command:

   ```
   ExtUserManager umclassname com.myCompany.user.MyUserManager
   ```

The custom `MyUserManager` class might be the following simple example:

```
package com.myCompany.user;
import com.alphablox.personalization.alphablox.*;
import com.alphablox.personalization.*;
import java.util.*;

public class MyUserManager extends AlphabloxUserManager {
   void start(Properties props) throws PEngineException  {
      super.start(props);
      String myXmlFile = (String) prop.get("customstartupprop");
      // read the xml file and do other things
   }
}
```

Example 2: Setting up DB2 Alphablox to Use a Different User Class

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

This example demonstrates how you can customize user authorization in the default DB2 Alphablox Repository-based User Manager. In this case, you want to allow everyone to access DB2 Alphablox or an application and overwrite the isUserInRole() method in the *IUser* interface. The steps involved include:

1. Write a custom user class in a package (in this example, the class is called `MyUser` in a package called `com.myCompany.user`)
   - The class extends the default `AlphabloxUser` class used in DB2 Alphablox Repository-based User Manager
2. Set up the new value for the `userclassname` server property using the following telnet console command:

   ```
   ExtUserManager userclassname com.myCompany.user.MyUser
   ```

   where `MyUser` might be the following simple example:

```
package com.myCompany.user;
import com.alphablox.personalization.repository.*;
import com.alphablox.personalization.*;
import java.util.*;

public class MyRepositoryUser extends AlphabloxUser {
   public boolean isUserInRole(HttpServletRequest req,
   String [] roles) throws PEngineException  {
      // Everybody can get in
      return true;
   }
}
```

Example 3: Setting up the DB2 Alphablox to Use a Different Group Class

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

This example demonstrates how you can customize the default DB2 Alphablox Repository-based User Manager to print a system message each time a group membership check is done. The steps involved include:

1. Write a custom group class in a package (in this example, the class is called `MyGroup` in a package called `com.myCompany.user`)
   - The class extends the default `AlphabloxGroup` class used in DB2 Alphablox Repository-based User Manager
2. Set up the new value for the `groupclassname` server property using the following telnet console command:

   ```
   ExtUserManager groupclassname com.myCompany.user.MyGroup
   ```

where `MyGroup` might be the following simple example:

```
package com.myCompany.user;
import com.alphablox.personalization.alphablox.*;
import com.alphablox.personalization.*;
import java.util.*;

public class MyGroup extends AlphabloxGroup {
   public boolean containsUser(IUser user,
          boolean checkSubGroups) throws PEngineException {
      boolean exists =           super.containsUser(user,checkSubGroups);
      System.out.println("User membership checked.");
      return exists;
   }
}
```

Interface Methods Cross-References

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

This section provides reference materials for the three interfaces in Extensible User Manager. For each method in the interfaces, description of the method, its syntax, and notes on its usage are provided.

For a complete example of a customized user manager, see the Simple User Manager example in the following directory:

```
<db2alphablox_dir>/system/documentation/admin/Examples/
```

Interface	Methods
IUserManager	`findGroup()` `findUser()`

	`getExternalProperties()`
	`getPrincipleUserName()`
	`hasExternalEditor()`
	`resume()`
	`setCaseSensitiveGroups()`
	`setCaseSensitiveUsers()`
	`start()`
	`stop()`
	`suspend()`

IUser	`authenticate()`
	`authorize()`
	`getName()`
	`getPassword()`
	`getPropertiesSubset()`
	`isUserInRole()`
	`refresh()`

IGroup	containsGroup() containsUser() getName() getPropertiesSubset() refresh()

IUserManager Interface

Note: This section applies only to DB2 Alphablox 8.4 (not supported in 8.4.1).

The `AbstractUserManager` class implements the *IUserManager* interface to find users and groups information and figuring out user identity during runtime. To extend this class, add the following import statement in your code:

```
import com.alphablox.personalization.*;
```

For an example of extending the `AbstractUserManager` class, see "Example 1: Setting up DB2 Alphablox to Use an External User Manager" on page 97.

findGroup()

Finds a group and returns the appropriate instance of *IGroup*.

Syntax

```
IGroup findGroup(String id, boolean fromCache);
                    // throws PEngineException
```

where:

Argument	Description
`id`	The group id.
`fromCache`	Whether to find the group from the cache

Usage

This method will return null:

- if the group does not exist in memory and *fromCache* is set to `true`
- if the group does not exist in memory, the group is not a valid group in the underlying user manager repository, and *fromCache* is set to `true`

findUser()

Finds a user and returns the appropriate instance of *IUser*.

Syntax

```
IUser findUser(String id, boolean fromCache);
                   // throws PEngineException
```

where:

Argument	Description
id	The user id
fromCache	Whether to find the user from the cache

Usage

This method will return null:

- if the user does not exist in memory and *fromCache* is set to `true`
- if the user does not exist in memory, the user is not a valid user in the underlying user manager repository, and *fromCache* is set to `true`

getExternalProperties()

Returns a String array of properties defined via the external editor.

Syntax

```
String[] getExternalProperties();  //throws PEngineException
```

getPrincipleUserName()

Returns the user name that is associated with this request.

Syntax

```
String getPrincipleUserName(HttpServletRequest request);
                   // throws PEngineException
```

where:

Argument	Description
request	The current HTTP request

Usage

This method is used by DB2 Alphablox to determine the user name when a new DB2 Alphablox session is created. You can use this method to override the default behavior of DB2 Alphablox.

hasExternalEditor()

Returns if there is an external user manager editor.

Syntax

```
boolean hasExternalEditor();
```

Usage

Returns `true` if the external user manager has its own editor. For example, this method returns `true` when LDAP-based User Manager is used. It will return `false` in the case of the default DB2 Alphablox Repository-based User Manager. This method should not be implemented to return `true` unless it is a complete implementation of the Extensible User Manager with an outside editor.

resume()

Resumes the User Manager.

Syntax

```
void resume();    // throws PEngineExceptionn
```

Usage

This method is called when DB2 Alphablox User Manager resumes its service. Call this method to resume DB2 Alphablox User Manager after its service is suspended.

Examples

The following code logs a message saying the User Manager has been restarted:

```
import com.alphablox.personalization.*;

public class MyUserManager extends AbstractUserManager {
...
   public void resume() throws PEngineException {
      System.out.println("Resumed");
   }
...
}
```

See Also

"suspend()" on page 102

setCaseSensitiveGroups()

Specifies if group names should be case sensitive or not.

Syntax

```
void setCaseSensitiveGroups(boolean caseSensitive);
                    // throws PEngineException
```

where:

Argument	Description
caseSensitive	true – group names are case-sensitive

Usage

When this method is set to true, group names are case-sensitive and the User Manager will respect that.

setCaseSensitiveUsers()

Specifies if user names should be case sensitive or not.

Syntax

```
void setCaseSensitiveUsers(boolean caseSensitive);
                    // throws PEngineException
```

where:

Argument	Description
caseSensitive	true — user names are case-sensitive

Usage

When this method is set to true, user names are case sensitive and the User Manager will respect that.

start()

Starts the User Manager.

Syntax

```
void start(java.util.Properties props);
        // throws PEngineException
```

where:

Argument	Description
props	DB2 Alphablox properties related to personalization. Valid properties are: `umclassname`, `groupclassname`, `custompropery`, and `ldapprops`. See "Extensible User Manager Telnet Console Command" on page 93 for details.

Usage

This method is called every time the DB2 Alphablox User Manager starts. This method is used to establish connection with appropriate repository and instantiate users and groups objects as needed. In the case of DB2 Alphablox Repository-based User Manager, all users and groups in the repository are instantiated. In LDAP-based User Manager, only users and groups that are registered in DB2 Alphablox will need to be created. Therefore it is a good idea to create users and groups as needed, such as when findUser() and findGroup() methods are called.

stop()

Stops the User Manager and frees all resources.

Syntax

```
void stop();     // throws PEngineException
```

Usage

This method is called when DB2 Alphablox User Manager stops its service.

See Also

"start()" on page 102, "suspend()" on page 102.

suspend()

Suspends the external user manager and frees up unused resources such as database connection.

Syntax

```
void suspend();     // throws PEngineException
```

Usage

This method is called when DB2 Alphablox User Manager suspends its service.

See Also

"resume()" on page 101

IUser Interface

To extend this class, add the following import statement in your code:

```
import com.alphablox.personalization.*;
```

For an example of extending the AbstractUser class, see "Example 2: Setting up DB2 Alphablox to Use a Different User Class" on page 97.

authenticate()

Checks if the supplied user and password through the HTTP request is the same as the user and password stored in the repository.

Syntax

```
boolean authenticate(HttpServletRequest request,
                     String authorizationHeader);
                   // throws PEngineException
```

where:

Argument	Description
request	The HTTP request.
authorizationHeader	Whether to find the group from the cache

Usage

Returns true if the user saved password is the same as the parameter password. This method is applicable only in DB2 Alphablox Apache Tomcat configurations (Apache Tomcat; no Microsoft IIS). where DB2 Alphablox uses basic authentication (i.e., browser challenged authentication). This method is called by DB2 Alphablox Tomcat Interceptors when users access the session the first time. The method has two parameters: the request object and the encoded authorization header. This header contains user name and password. You can use the following method to decode the header:

```
AbstractUserManager.getDecoder().decode(authorizationHeader);
```

After you obtain the password you have the option to save it in the object memory if you intend to use it as the return value of the getPassword() method to allow single sign-on with the data source.

authorize()

Checks if user is in the provided list of roles.

Syntax

```
boolean authorize(HttpServletRequest request,
                  String[] roles);
          // throws PEngineException
```

where:

Argument	Description
request	The HTTP request.
roles	A list of roles to check if the user is in one of them.

Usage

Returns true if the user is in one of the provided roles.

getEmail()

Gets the user's e-mail.

Syntax

```
String getEmail();    // throws PEngineException
```

Usage

Return null if the DB2 Alphablox maintains the e-mail name rather than reading from the external repository. In the case of LDAP, user's e-mail and full name are obtained from LDAP and their values are not editable in the DB2 Alphablox Admin Pages. In that case, you can use this method to get the user's e-mail.

getFullName()

Gets the user's full name.

Syntax

```
String getFullName();    // throws PEngineException
```

Usage

Return null if the DB2 Alphablox maintains the full name rather than reading from the external repository. In the case of LDAP, user's e-mail and full name are obtained from LDAP and their values are not editable in DB2 Alphablox Admin pages. In that case, you can use this method to get the user's full name.

getName()

Gets the user name.

Syntax

```
String getName();      // throws PEngineException
```

Usage

The user name must be a valid DB2 Alphablox username.

getPassword()

Gets the user's password.

Syntax

```
String getPassword();     // throws PEngineException
```

Usage

Gets the base64–encoded password of the user. When the data source is set to use DB2 Alphablox username and password, DB2 Alphablox will use the encoded password saved in the repository for authentication and for accessing the data source. Use this method for custom security such as single sign-on to data sources. Any password saved via this method will be used temporarily in the memory. Note that this password should be encoded using the following code:

```
AbstractUserManager.getEncoder().encode(password);
```

before it is returned to callers.

See Also

"Single Sign-On" on page 96.

getPropertiesSubset()

Gets the subset of properties of the user that is of interest.

Syntax

```
java.util.Properties getPropertiesSubset(String[] propList);
                // throws PEngineException
```

where:

Argument	Description
propList	An array of Strings for the properties subset

Usage

Returns a Properties object for the subset of the user properties passed in. It is assumed that the values will be obtained from the memory unless the developers change that.

isUserInRole()

Identifies if the user belongs to one of the specified roles.

Syntax

```
boolean isUserInRole(HttpServletRequest request,
                     String[] roles);
            // throws PEngineException
```

where:

Argument	Description
request	The HTTP request
roles	The role or roles to which the user belongs.

Usage

Returns `true` if user belongs to one of the roles in the list. The role implementation is left to developers. The implementation of this method is based on the standard J2EE API `Request.isUserInRole(String` *role*`)` method. DB2 Alphablox uses this method to determine user access to the administrative functionality and bookmark write access.

With LDAP-based User Manager, this method calls the `IGroup.`containsUser() method since roles are equivalent to groups in LDAP.

See Also

"containsGroup()" on page 106

refresh()

Refreshes all information cached in memory and get the latest from the underlying repository.

Syntax

```
void refresh();       // throws PEngineException
```

Usage

Forces refreshing any user information saved in memory and gets the latest from external repository.

IGroup Interface

To extend this class, add the following import statement in your code:

```
import com.alphablox.personalization.*;
```

For an example of extending the AbstractGroup class, see "Example 3: Setting up the DB2 Alphablox to Use a Different Group Class" on page 98.

containsGroup()

Checks if this group contains a subgroup.

Syntax

```
boolean containsGroup(IGroup group, boolean checkSubGroups);
                  // throws PEngineException
```

where:

Argument	Description
group	An instance of IGroup.
checkSubGroups	true — check subgroups; false— do not check subgroups.

Usage

Returns true if this group contains a subgroup.

containsUser()

Checks if this group contains the specified user.

Syntax

```
boolean containsUser(IUser user, boolean checkSubGroups);
                  // throws PEngineException
```

where:

Argument	Description
user	An instance of IUser.
checkSubGroups	true — check subgroups; false — do not check subgroups

Usage

Returns true if the group contains the specified user.

getName()

Gets the group name.

Syntax

```
String getName();      // throws PEngineException
```

Usage

The group name must be a valid DB2 Alphablox group name.

getPropertiesSubset()

Gets the subset of properties of the group that is of interest.

Syntax

```
java.util.Properties getPropertiesSubset(String[] propList);
                // throws PEngineException
```

where:

Argument	Description
propList	An array of Strings containing the properties.

Usage

Returns a Properties object for at least the list of group properties passed in. It is assumed that the values will be obtained from the memory unless the developers change that.

refresh()

Refreshes all information cached in memory and get the latest from the underlying repository.

Syntax

```
void refresh();       // throws PEngineException
```

Usage

Forces refreshing any group information saved in memory and gets the latest from external repository.

Chapter 16. Using a Database Repository

This chapter describes how to configure DB2 Alphablox to use a relational database for the DB2 Alphablox Repository.

Overview of the DB2 Alphablox Repository

The DB2 Alphablox Repository is a store of objects that DB2 Alphablox uses to keep track of applications, users, groups, bookmarks, and other information. The repository can reside either in the operating system file system or in a relational database. If an DB2 Alphablox configuration is set up to run in a clustered environment, the repository must reside in a relational database. The relational database allows multiple server nodes to read from and write to the repository, ensuring a consistent state for the data in the repository.

Repository Within the DB2 Alphablox Environment

Within DB2 Alphablox, the Repository Manager controls access to the repository. The Repository Manager uses Java Naming and Directory Interface (JNDI) to communicate with the repository, whether the repository resides in a file system or in a relational database. The JNDI layer uses a different service provider for the different types of repositories: DB2 Alphablox file system or database.

The supported databases for the repository are the same as those supported for data source access, as described in the *Installation Guide*. For more details about the architecture of DB2 Alphablox, see "DB2 Alphablox architecture" on page 7.

The Repository Conversion Utility is a Java program that is used to convert the repository between file system and file system, file system and database, between database and file system, or between different databases. The conversion utility runs automatically during the installation process and can also be run manually. For details on using the conversion utility, see "Using the Repository Conversion Utility" on page 110.

Advantages of the Relational Repository

Using a relational database for the DB2 Alphablox Repository has two main advantages:

- A relational repository can be accessed by multiple servers, thus enabling a clustered environment for DB2 Alphablox and providing a means for virtually unlimited scalability.
- A relational repository provides you with all the industrial-strength tools available in a database environment for things like transactional integrity, backup and restore operations, rollback to a consistent state, database replication, and so on.

The repository stores objects critical to the operation of DB2 Alphablox, so having all the tools of a database available to ensure the integrity of the data ensures a more robust and reliable system.

Configuring the DB2 Alphablox Repository

The initial state of the DB2 Alphablox Repository is established during the installation process. The DB2 Alphablox Repository has two *service providers* it uses to access the repository, depending on the repository type: the **DB2 Alphablox Filesystem Service Provider** and the **DB2 Alphablox Database Service Provider**. The default state is to use the **DB2 Alphablox Filesystem Service Provider**. All of the relational databases supported by DB2 Alphablox are supported databases for the **DB2 Alphablox Database Service Provider**. For details on specifying your repository type during the installation process and for the supported relational databases, see the *Installation Guide*.

Checking Your Repository Type

To check which repository service provider is in use for your server, perform the following steps:

1. Make sure DB2 Alphablox is running.
2. Log into the DB2 Alphablox home page as the *admin* user or as a user who is a member of the administrators group.
3. Click the **Administration** tab. The **General** page under the Administration tab appears.
4. In the **General Properties** section of the page, click the **Repository Manager** link to view DB2 Alphablox repository properties.

   The **Repository Service Provider** entry lists which JNDI service provider is in use, the **Database Adapter** entry lists which JDBC driver is used, and the other entries list configuration information for the data source. If the filesystem service provider is in use, the **Repository Location** entry lists the directory in which the repository files are stored.

Note: You cannot change the repository type in the Repository Manager properties page. To change the repository type, use the Repository Conversion Utility.

Using the Repository Conversion Utility

If you want to change the repository type (for example, from filesystem to an Oracle database), you must run the Repository Conversion Utility. The Repository Conversion Utility is a Java program that runs in a command line window (MS-DOS window on Windows systems, xterm or other command window on Linux and UNIX systems). It creates the tables and/or files necessary to move an DB2 Alphablox Repository from one place to another. You can use the Repository Conversion Utility to move a repository from filesystem to filesystem, filesystem to database, database to filesystem, or database to database.

Starting the Repository Conversion Utility

You can start the Repository Conversion Utility from the **Start** menu on Windows systems that have DB2 Alphablox installed or by running the following file on Windows systems.

Important: Always shut down DB2 Alphablox before running the Repository Conversion Utility.

```
<db2alphablox_dir>/Tools/convert/ConvertRepository.exe
```

or by running the following file on Linux and UNIX systems:

```
<db2alphablox_dir>/Tools/convert/ConvertRepository
```

where <db2alphablox_dir> is the directory in which DB2 Alphablox is installed.

Repository Conversion Utility Interactive Command Line Options

The following table describes each interactive command line option for the main menu of the Repository Conversion Utility.

Option		Description
1	Set DB2 Alphablox File Manager Root	Sets the directory for the location of the repository files used for a filesystem repository. The directory must be a valid DB2 Alphablox directory or the operation will fail with an error.
2	Set DB2 Alphablox Instance Name	Selects which instance of DB2 Alphablox the Repository Conversion Utility will access. The instance specified must be a valid DB2 Alphablox instance or the operation will fail with an error.
3	Convert One Repository to Another	Converts an existing repository to a different repository, moving all the necessary data from one repository to the other. There are options to convert the repository from file to database, from database to file, from file to file, and from database to database. You are prompted to use one of the following options when you convert a repository: • COPY: Copies the contents of one repository into another, leaving the original repository in place. • MOVE: Moves the contents of the repository out of one repository into another. The original repository is deleted. You are also prompted to use one of the following options for the destination repository: • NEW: Creates a new repository in the destination. • UPDATE: Updates the data in the destination repository from the source repository instead of replacing it with a new repository • OVERWRITE: Recreates the tables and replaces all the data in the destination repository with the information from the source repository. The OVERWRITE option does not preserve any of the data in the destination repository.
4	Create an Empty Database Repository	Creates the tables in a database needed for the DB2 Alphablox Repository. This option only creates the tables with their initial content; it does not populate them with any repository objects.
5	Verify and Repair a Repository	Inspects a repository for problems and reports any problems it finds. Includes options to verify both file and database repositories. For database repositories, this option can also correct some problems.
6	Change DB2 Alphablox to use a different Repository	Redirects an instance of DB2 Alphablox to point to a different repository. The repository you change to must exist and be accessible from the DB2 Alphablox machine.

Option		Description
7	Conversion Utility Options	Allows you to toggle between verbose and normal messages for the utility and to erase the buffer that stores the history the conversion utility uses to remember information you have entered. Also provides an option to specify different DDL schema files to create the repository tables. Do not specify different DDL files unless you have thoroughly tested them.
8	Exit	Exits the Repository Conversion Utility.

The Repository Conversion Utility keeps a log file named `repositoryconvert.log` that saves all the activity in your conversion utility session. The `repositoryconvert.log` file is located in the same directory as the conversion utility (`<db2alphablox_dir>/Tools/convert`).

Converting From Filesystem to Database

Before converting a repository from filesystem to database, gather the following information:

- The full path to your filesystem repository (for example, `d:\alphablox\repository`)
- The connection information for the database

The following procedure describes the steps necessary to convert an DB2 Alphablox Repository from filesystem to database.

1. Shut down DB2 Alphablox.
2. Start the Repository Conversion Utility (for details, see "Starting the Repository Conversion Utility" on page 110).
3. Select option 3, **Convert one repository to another**, by typing the number **3** and pressing the enter key.
4. Select option 1, **Convert file to database**, by typing the number **1** and pressing the enter key.
5. Confirm the repository root directory by pressing the enter key.

   Note: If the directory shown as the default is not your repository directory, make sure you are accessing the correct instance of DB2 Alphablox. The default instance the Repository Conversion Utility access is the name of the installed instance, which defaults to `AlphabloxAnaltyics`. If you have a different instance name, exit the current sequence and set the correct instance name from option 2 on the main menu.
6. If everything is correct, select **Continue** by typing the number **1** and pressing the enter key.
7. Select the database corresponding to your database server. For example, if you are using Oracle 8.1.7, enter **2**.
8. Enter the configuration information for your database as prompted.
9. If everything is correct, select **Continue** by typing the number **1** and pressing the enter key.
10. Enter COPY or MOVE. COPY leaves the old repository in place and creates a copy of it in the destination repository, MOVE deletes the old repository and creates a new repository in the destination repository.
11. Enter NEW, UPDATE, or OVERWRITE. Use NEW to create a new repository with new tables if one does not already exist. Use UPDATE to keep the

existing table structure of the repository and update it to include the data from the source repository. Use OVERWRITE to delete all the old data and database tables and recreate the new tables and data in its place.

12. If you want the new repository to be used by your instance of DB2 Alphablox, enter **Y** at the **Update DB2 Alphablox to use the New Repository** prompt.
13. To update all necessary properties for the DB2 Alphablox instance, select ALL at the **Update DB2 Alphablox Properties** prompt. ALL specifies that all server properties be converted, SPECIFIC specifies that only the server properties specific to the local machine (not the clustered properties) be converted, and GLOBAL specifies that only the properties shared by the cluster (not the local machine entries) be converted. NONE specifies that no properties are changed on the DB2 Alphablox instance.
14. If everything is correct, select **Continue** by typing the number **1** and pressing the enter key.
15. When the conversion is complete, you are returned to the main menu. Enter **8** to exit the Repository Conversion Utility.

Converting From Database to Filesystem

Before converting a repository from database to filesystem, gather the following information:

- The connection information for the database repository
- The full path to where you want the filesystem repository to reside (for example, `d:\alphablox\Repository`)

The following procedure describes the steps necessary to convert an DB2 Alphablox Repository from database to filesystem.

1. Shut down DB2 Alphablox.
2. Start the Repository Conversion Utility (for details, see "Starting the Repository Conversion Utility" on page 110).
3. Select option 3, **Convert one repository to another**, by typing the number **3** and pressing the enter key.
4. Select option 2, **Convert database to file**, by typing the number **2** and pressing the enter key.
5. Select the database corresponding to your database server. For example, if you are using Oracle 8.1.7, enter **2**.
6. Enter the configuration information for your database as prompted.
7. If everything is correct, select **Continue** by typing the number **1** and pressing the enter key.

   The Repository Conversion Utility will attempt to connect to the database.
8. Enter the directory for the destination repository root and then press the enter key.
9. If everything is correct, select **Continue** by typing the number **1** and pressing the enter key.
10. Enter COPY or MOVE. COPY leaves the old repository in place and creates a copy of it in the destination repository, MOVE deletes the old repository and creates a new repository in the destination repository.
11. Enter NEW, UPDATE, or OVERWRITE. Use NEW to create a new repository if one does not already exist. Use UPDATE to keep the existing structure of the repository and update it with the data from the source repository. Use OVERWRITE to delete all the old data and structures and recreate the new structures and data in its place.

12. If you want the new repository to be used by your instance of DB2 Alphablox, enter **Y** at the **Update DB2 Alphablox to use the New Repository** prompt.
13. To update all necessary properties for the DB2 Alphablox instance, select ALL at the **Update DB2 Alphablox Properties** prompt. ALL specifies that all server properties be converted, SPECIFIC specifies that only the server properties specific to the local machine (not the clustered properties) be converted, and GLOBAL specifies that only the properties shared by the cluster (not the local machine entries) be converted. NONE specifies that no properties are changed on the DB2 Alphablox instance.
14. If everything is correct, select **Continue** by typing the number **1** and pressing the enter key.
15. When the conversion is complete, you are returned to the main menu. Enter **8** to exit the Repository Conversion Utility.

Configuring an Instance to Use an Existing Repository

Before configuring an instance of DB2 Alphablox to use an existing repository, gather the connection information for the database repository to which you want to connect.

The following procedure describes the steps necessary to configure an instance of DB2 Alphablox to use an existing DB2 Alphablox Repository.

1. Shut down DB2 Alphablox.
2. Start the Repository Conversion Utility (for details, see "Starting the Repository Conversion Utility" on page 110).
3. Select option 6, **Change DB2 Alphablox to use a different repository**, by typing the number **6** and pressing the enter key.
4. Select option 2, **Database repository is the target**, by typing the number **2** and pressing the enter key.
5. Select the database corresponding to your database server. For example, if you are using Oracle 8.1.7, enter **2**.
6. Enter the configuration information for your database as prompted.
7. If everything is correct, select **Continue** by typing the number **1** and pressing the enter key.

   The Repository Conversion Utility will attempt to connect to the database and then will update the instance of DB2 Alphablox to use the specified repository.
8. Enter **8** to exit the Repository Conversion Utility.

Command Line Syntax

In most cases, you can use the Repository Conversion Utility in interactive mode, as described in the previous sections. You can, however, use command line options to specify different database DDL files or to run the conversion utility as part of an automated script. The basic syntax for the repository utility is as follows:

```
java -cp [class_path] com.alphablox.util.convert.Convert operation
     destination [source] [arguments]
```

where:

class_path — is the Java classpath for the instance of DB2 Alphablox. Look in the DB2 Alphablox startup files (for example, `AnalysisServer.bat` on Windows platforms and `AnalysisServer.sh` on Linux and UNIX platforms) for an example of a *class_path*.

operation — is the Repository Conversion Utility operation to

	be performed. For a description of each available operation, see "Operation Descriptions" on page 115.
destination	is a relative or absolute path to a properties file describing the destination repository. For sample destination files, see "Sample Source and Destination Property Files" on page 116.
source	is a relative or absolute path to a properties file describing the source repository. The *source* argument must be specified for COPY or MOVE operations. For sample destination files, see "Sample Source and Destination Property Files" on page 116.
arguments	is one or more of the arguments specified in the table in "Arguments" on page 115.

If no *operation*, *destination*, *source*, or *arguments* are specified, the Repository Conversion Utility runs in interactive mode.

Operation Descriptions: The following table describes the available operations for the Repository Conversion Utility.

Operation	Description
HELP	Displays the command line help.
CHANGE	Changes the active repository to the destination repository.
COPY	Leaves the source repository in place and creates a copy of it in the destination repository.
DELETE	Permanently deletes the destination repository.
MOVE	Deletes the source repository and creates a new repository in the destination repository.
NEW | CREATE	Creates a new repository in the destination and populates it with the default repository values.
VERIFY	Runs validation operations on the destination repository and reports the results.

Arguments: The following table describes the arguments for the Repository Conversion Utility.

Argument	Description
DEBUG	Outputs additional debug information to help diagnose problems.
LOG:*file*	Specifies the name of the conversion utility log file which logs all of the activity of the conversion utility. The default name for *file* is *repositoryconvert.log*.
SERVER:*InstanceName*	Specifies the instance name for the instance of DB2 Alphablox. The SERVER argument is required.
OVERWRITE	Used with the NEW, MOVE, COPY, or CHANGE operation, the OVERWRITE argument deletes all

	the old data and structures in the destination repository and recreates the new structures and data in its place. Without this argument, the Repository Conversion Utility will stop if it detects an existing repository at the destination.
PROPS:*option*	Used with the SERVER:*InstanceName* argument, specifies which server properties are updated on the destination repository. The values for *option* are: • ALL: specifies that all server properties be converted. • GLOBAL: specifies that only the properties shared by the cluster (not the local machine entries) be converted. • SPECIFIC: specifies that only the server properties specific to the local machine (not the clustered properties) be converted.
UPDATE	Specifies that a MOVE or COPY operation updates the destination repository with information from the source repository rather than replacing the contents of the destination repository.
DDL:*file*	Overrides the default DDL schema file used to create the tables, indexes, and initial content for the database repository. **Important:** Use extreme caution if using this argument.
USEDEST	When specified, modifies the server properties to use the destination repository. If not specified, the repository property for the server are not changed.

Sample Source and Destination Property Files: This section shows the contents of sample source and destination files used with the Repository Conversion Utility. The property files specify the type of repository, the connection information for the repository, the location of the repository property files on the computer in which DB2 Alphablox runs, and the name of the DDL file.

The following example shows a property file for a repository residing in an Oracle database on a server named *oracle817*:

```
RepositoryTarget=JDBCTarget
java.naming.factory.initial=com.alphablox.jndisp.AlphabloxContextFactory
java.naming.provider.url.server=oracle817
java.naming.provider.url.port=1521
java.naming.provider.url.sid=orcl817
database_driver=oracle.jdbc.driver.OracleDriver
fileroot=C:\alphablox\analytics\repository\servers\
commandfile=oracle.dmlsql
user=user
password=password
```

The following example shows a property file for a repository residing in a filesystem:

```
RepositoryTarget=ABXTarget
java.naming.factory.initial=com.alphablox.jndisp.AlphabloxContextFactory
java.naming.provider.url=C:\alphablox\analytics\repository\
fileroot=C:\alphablox\analytics\repository\servers\
```

Chapter 17. Using Connection Pooling

This chapter describes how to configure DB2 Alphablox to use connection pooling with multidimensional and relational data sources.

Connection Pooling - Overview

Each time a web-based application needs to interact with a multidimensional or relational database, it must first connect to it. Each of these connections incurs overhead, using resources to establish the connection, maintain it, and release it when it is no longer required. In web-based applications, user interactions with databases are typically short and, often, more time is spent during connecting and disconnecting to a database than is spent on the request themselves.

To more efficiently handle these database interactions, application servers and databases typically offer connection pooling. Connection pools are a group of preestablished database connections that can be shared with applications, allowing the needed database interactions, but not consuming as much time and resources as individual connections.

If available and properly configured, DB2 Alphablox can utilize connection pooling to improve the performance of your applications. Connection pooling with DB2 Alphablox can be used with application servers (IBM WebSphere and BEA WebLogic) for relational database connection pooling, or with the connection pooling support in IBM DB2 OLAP Server Deployment Services or Hyperion Essbase Deployment Services.

MDB Connection Pooling

Multidimensional database connection pooling is not supported using standard JDBC connections, which only work with relational data sources. For DB2 OLAP Server or Hyperion Essbase, DB2 Alphablox can use the implementations of connection pooling available in IBM DB2 OLAP Server Deployment Services or Hyperion Essbase Deployment Services.

DB2 OLAP Server and Hyperion Essbase Connection Pooling

IBM DB2 OLAP Server Deployment Services or Hyperion Essbase Deployment Services can be configured to support connection pooling. Each defined connection pool specifies the DB2 OLAP Server or Essbase user name and password to use. A connection pool's access can be defined for all users ("Allow Everyone") or for a list of specified users and groups. If a DB2 Alphablox data source specifies the IBM DB2 OLAP Server Deployment Services or Hyperion Essbase Deployment Services adapter and the connection pools are configured properly in IBM DB2 OLAP Server Deployment Services or Hyperion Essbase Deployment Services, DB2 Alphablox can use these connection pools. For details on configuring Essbase connection pools, see the IBM DB2 OLAP Server Deployment Services or Hyperion Essbase Deployment Services documentation.

Microsoft Analysis Services and Connection Pooling

The use of connection pooling in Microsoft Analysis implementations can be beneficial in improving the performance of analytic applications. By default, connection pooling is disabled.

When the connection pool is enabled and the DataBlox *disconnect()*method is called, the DataBlox will disconnect, but the ADO connection will not be closed and will be placed in a connection pool available for reuse. When a subsequent DataBlox connection requests a connection that matches one in the pool, the connection will be removed from the pool and given to the DataBlox. Otherwise, a new connection will be created. A pooled connection is keyed against the data source user name, password, provider string, catalog (the Microsoft Analysis Services database name), and schema (which would normally be null as it is not used by Microsoft Analysis Services). All five properties need to match in order for a DataBlox component to use one of the connections in the pool. Changing the provider string on different data sources leads to different unique connections in the pool. Each of these unique connections is essentially a connection pool key. For example, if there are two connections as user1 to database1 using the default provider string of "MSOLAP" and another connection as the same user and database, but with a provider string of *"MSOLAP;Default Isolation Mode=1;Execution Location=3;Client Cache Size=0;"* there will be three connections using two different connection pool keys.

Enabling the connection pool

The MSOLAP connection pool is disabled by default. Then enable it, you need to set the MSOLAPEnableConnectionPool server property to true.

To enable the connection pool:

1. Open a telnet console to your DB2 Alphablox server (or use the Admin Console window, available in the DB2 Alphablox Admin Pages).
2. Set the MSOLAPEnableConnectionPool server property by typing "*set MSOLAPEnableConnectionPool true*" and press Enter. A message appears stating "Server property 'Enable pooling of the MSOLAP connection pool' set to true."
3. Type "save" and press Enter. The "Server properties saved" message appears.

   Note: If you do not save the property, the server property will apply only during the current server session.

To disable the connection pool, repeat the steps above, but set the MSOLAPEnableConnectionPool property to false.

Using the connection pool

When using the DB2 Alphablox Microsoft Analysis Services connection pool, you need to ensure that on all of the DataBlox autoDisconnect property is set to true. If you do not, connections will not be returned to the available pool until after the user session has ended. Also, the clearClientCache() method behaves differently than it did without connection pooling. If the clearClientCache() method is called when connection pooling is enabled, it clears all of the connections from the available pool whose connection pool key matches the DataBlox connection. Additionally, you can also clear the entire connection pool using the ODBOBridgeJavaAPI.clearPool() method. This method clears all of the ADO connections in the available pool.

Constraining the connection pool

For environments with very high concurrency and few unique pooled connections (user names), you can improve both memory consumption and performance by restricting the pool size. The MSOLAPConnectionPoolLimit server property has been added that will limit the number of active MSOLAP connections that DB2 Alphablox has open per connection pool key. This includes connections that are in the pool. This feature should be used, though with caution; it is really intended only for the unique environment described here. The

MSOLAPConnectionPoolLimit property limits the number of connections per connection pool key. Use the MSOLAPConnectionPoolLimit server property to set the maximum number of connections.

For example, setting the MSOLAPConnectionPoolLimit to 4 will set the maximum number of open MSOLAP connections per connection pool key to 4. In this example case, assume that the connections to MSAS are identical except for the username property and you have only 3 user names (Exec, Manager, and Admin) that connect. If a DataBlox component attempts to connect using the userid Admin and it is the 5th connection it will need to wait until a connection for that user name is freed (or closed) and placed back in the pool, making the number of open connections fall under the limit.

Tuning the connection pool

Using the DB2 Alphablox Microsoft Analysis Services connection pooling, you can tune the pool to optimize both memory usage and performance. There are a number of factors to consider

Microsoft Analysis Services Client Side Cache: One concept commonly not recognized in connectivity is the impact of the Microsoft Analysis Services client side cache. MSAS can cache a number of things on the DB2 Alphablox host. For example, calculated measures will always be computed and cached in the ADO layer on the Alphablox host. If the information is continually reused between Alphablox sessions, then it is beneficial to have an ADO connection continually open that contains that information.

Connection pooling considerations: Connection pooling is useful when you have a limited number of connection pool keys used in connecting (a small number of distinct usernames) or when DB2 Alphablox sessions have multiple Blox connections that use the same connection pool key.

Do not use connection pooling if all users (which would required unique connection pool keys) are unique and you have only a single DataBlox per connection pool key or if you have a large number of connection pool keys and are unable to clear them when users sessions end.

Using the autoDisconnect property: You should almost always use the DataBlox autoDisconnect property when using connection pooling. Otherwise, ADO connections will not be returned to the pool for reuse.

Using the clearClientCache() method: If you are using a large number of connection pool keys (that is, every connection to Microsoft Analysis Services is a specific named user) you will want to call the DataBlox clearClientCache() method on each DataBlox component with a unique connection pool key when the user is done with the session. This is generally done through a logout page that is invoked when the user explicitly logs out or closes his browser window.

Using the clearPool() method: In most situations it probably makes sense to periodically clear the connection pool. A simple timer servlet that clears the pool every night is probably adequate for most environments.

Monitoring and managing the connection pool: The following static methods are available on the ODBOBridgeJava class for use in monitoring and managing the connection pool:

ODBOBridgeJavaAPI.poolSize()
Returns the number of connections in the available pool

ODBOBridgeJavaAPI.clearPool()
Disconnect connections in the available pool

ODBOBridgeJavaAPI.getMaximumNumberOfConnections()
Gets the maximum number of MSAS connections which can be open

ODBOBridgeJavaAPI. getNumberOfOpenConnections()
Gets the current number of MSAS connection which are open

The number of open ADO connections equals the number of connections in the available pool plus the number of connections being used.

The following JSP scriptlet example uses the methods above to manage and monitor a connection pool:

```
<% ODBOBridgeJavaAPI.setMaximumNumberOfConnections(15);
  int count = ODBOBridgeJavaAPI.poolSize();
  out.write("Number of connections in the pool: "+count+"<br>");
  count = ODBOBridgeJavaAPI.getNumberOfOpenConnections();
  out.write("Number of connections active: "+count+"<br>");
  count = ODBOBridgeJavaAPI.getMaximumNumberOfConnections();
  out.write("Connection limit: "+count+"<br>");
%>
```

RDB Connection Pooling

An RDB connection pool is a named group of identical JDBC connections to a relational database created when a connection pool is registered on an application server. A web-based application can “borrow” a connection from the pool, use this connection during an interaction with the database, then return the connection to the pool by closing the connection.

The following table highlights benefits of using connection pooling:

Benefit	Description
Improved response time	Each time a resource attempts to access a database, it must connect to that database. When a connection pool starts up, it creates a specified number of physical database connections, eliminating the overhead of creating database connections for each application. By eliminating the overhead of creating, maintaining, and releasing connections, connection pools significantly improve the response time for resultset retrievals.
Abstraction from underlying databases	Connection pooling provides a layer of abstraction from underlying databases, making it easier to switch application databases without having to deal with vendor-specific SQL exceptions. Instead, you only have to work with the connection pooling exception from the application server.
Concurrent connection management	If you have a limited number of database connections due to licensing restrictions, connection pooling might be able to be used to manage the number of concurrent database connections.

DB2 Alphablox Usage of RDB Connection Pooling

If you are using WebSphere and WebLogic application servers with DB2 Alphablox, the following can utilize connection pooling:
- DB2 Alphablox Data Sources
- Database-based DB2 Alphablox Repository
- RDB cubing

- ReportBlox
- CommentsBlox
- JDBC Connection beans

All of the DB2 Alphablox features listed above, other than DB2 Alphablox data sources and any relational database-based DB2 Alphablox Repository, will automatically take advantage of RDB connection pooling, if available and configured properly.

DB2 Alphablox Data Sources and RDB Connection Pooling

To use RDB connection pooling with DB2 Alphablox and WebSphere or WebLogic application servers, the Application Server Data Source Adapter option must be selected on your DB2 Alphablox Data Source definitions. This adapter option only appears when DB2 Alphablox is configured to use supported WebSphere and WebLogic application servers.

Note: When using WebSphere or WebLogic connection pools to access data sources, you can use DB2 Alphablox with any of the relational databases supported on the application servers.

Note: The Apache Tomcat configuration of DB2 Alphablox does not support RDB connection pooling.

DB2 Alphablox Repository and RDB Connection Pooling

To enable or disable RDB connection pooling with a relational-based DB2 Alphablox Repository, following these steps:

Enabling

1. Create an DB2 Alphablox Data Source for the defined connection pool on your application server. See the directions above for details.
2. Run the Repository Conversion Utility.

   Windows:

   ```
   <serverDirectory>/tools/convert/ConvertRepository.exe
   ```

   Linux and UNIX:

   ```
   <serverDirectory>/tools/convert/ConvertRepository.bat
   ```
3. If DB2 Alphablox is using WebSphere or WebLogic, option 8 (“Configure Web Application Server Connection Pooling”) will be available. Select this option, then press Enter.
4. Select option 1 (“Turn On connection pooling”), then press Enter.
5. Enter the JNDI name of the data source defined in the application server.

Disabling

1. Run the Repository Conversion Utility.
2. If DB2 Alphablox is running on WebSphere or WebLogic, option 8 (“Configure Web Application Server Connection Pooling”) will be available. Select this option, then press Enter.
3. Select option 1 (“Turn Off connection pooling”).

Configuring Connection Pooling with BEA WebLogic

When using BEA WebLogic connection pooling with relational data sources, BEA WebLogic requires that you create a WebLogic user for each data source in order to access the defined connection pool.

To configure DB2 Alphablox correctly to work with WebLogic connection pooling, the DB2 Alphablox data source definition Default Username and Default Password must be a WebLogic user. If you are not using connection pooling, you do not need to create a WebLogic user to access relational data sources.

Chapter 18. Using Clustered Environments

This chapter describes how to configure and use DB2 Alphablox in clustered environments, allowing for scalable analytic applications. Two clustering environments are discussed: the WebSphere clustering environment and the WebLogic standalone clustering environment.

Overview of Clustered Environments

For increased scalability and availability, DB2 Alphablox can run on a cluster of two or more servers. The more servers in the cluster, the greater the number of users that the system can support. A cluster provide two key features:

- **Scalability:** The capacity of a cluster is not limited to a single machine. New servers can be added to the cluster dynamically to increase capacity
- **High Availability:** A cluster uses the redundancy of multiple servers to insulate clients from failures.

Note: If an DB2 Alphablox configuration is set up to run in a clustered environment, the repository must reside in a relational database. See "Overview of the DB2 Alphablox Repository" on page 109 to learn more about the DB2 Alphablox Repository and configuring it using a relational database.

WebSphere Clustering Environments

For information on configuring and installing DB2 Alphablox in a WebSphere clustering environment, see *Using DB2 Alphablox in WebSphere Clustered Environments* in the *Installation Guide*. For details on configuring and using WebSphere clusters, see the WebSphere Server documentation.

WebLogic Clustering Environments

A WebLogic cluster is a group of WebLogic servers that work together to provide scalability and reliability that is not available with a single server environment. While appearing to clients as a single server, a cluster is in fact a group of servers. These WebLogic clusters can be multiple instances on the same physical machine, or WebLogic servers installed on multiple physical machines.

Each DB2 Alphablox host communicates with all of the other nodes in the cluster to notify each other when anything in the repository has changed. Any changes to the repository (for example, a new user) are stored directly in the repository and each node can read and modify the repository.

Configuring and Installing DB2 Alphablox in WebLogic Clustering Environments

For information on configuring and installing DB2 Alphablox in a WebLogic clustering environment, see *Using DB2 Alphablox in WebLogic Clustered Environments* in the *Installation Guide*. For details on configuring and using WebLogic clusters, see the WebLogic Server documentation.

Creating New Applications in WebLogic Clustering Environments

For instructions on defining new DB2 Alphablox applications in a WebLogic clustering environment, see "Defining an Application When Using WebLogic Clusters" on page 34.

Using WebLogic Vertical Clusters

A vertical cluster configuration can be used to improve the scalability of DB2 Alphablox applications that work with Microsoft Analysis Services or to host multiple server instances on a single machine for efficiency and cost savings.

For Microsoft Analysis Services users, WebLogic vertical clusters allow DB2 Alphablox applications to bypass the Microsoft Analysis Services 2000 process limit of 2 GB on Windows operating systems by running multiple instances of DB2 Alphablox on a single machine. This can improve scalability of Microsoft Analysis Services.

Vertical clusters can also be used on single, powerful machines where you want to leverage your IT investments and reduce maintenance issues associated with the use of multiple machines.

For information on configuring and installing DB2 Alphablox in a WebLogic clustering environment, see *Using DB2 Alphablox in WebLogic Clustered Environments* in the *Installation Guide*. For details on configuring and using WebLogic clusters, see the WebLogic Server documentation.

Cluster Console Commands

In addition to the administrative user interface, you can also use the DB2 Alphablox console to enter the information on the **Cluster Options** page. To display the current cluster property settings, enter the following in an DB2 Alphablox console window:

```
get service cluster
```

The screen output to the console is similar to the following:

```
get service cluster
Service Cluster Manager Properties:

IsClustered ........ false
  (Is clustering enabled [true|false])
LeadHost ...........
  (Name or IP address of the lead AAS host in the cluster)
PortNum ............ 7855
  (The port number on which the lead host listens for cluster messages)
MaxHosts ........... 10
  (The maximum number of AAS hosts in the cluster)
StartupTime ........ 60
  (Time, in seconds, each normal node waits while connecting to
   the cluster)
```

```
LiveIsClustered .... false
  (Clustering is currently enabled)
LiveLeadHost .......
  (Current name or IP address of the lead AAS host.)
LivePortNum ........ 7855
  (Current port number the lead host is listening on.)
LiveMaxHosts ....... 10
  (Current maximum number of AAS hosts allowed in the cluster)
LiveStartupTime .... 60
  (Current time each normal node waits while connecting to the
   lead host)
```

The following table lists the cluster console commands and a description of what each one does:

Command Syntax	Description
`cluster shutdown`	Shuts down all of the instances in the DB2 Alphablox cluster. The `CLUSTER SHUTDOWN` command can be run from the console of any of the hosts in the cluster.
`get service cluster`	Lists the current settings for the cluster properties. The properties that begin with "Live" (for example, `LiveIsClustered`) return the actual values the cluster manager is using; they are not user configurable. The other properties can be set using a `SET SERVICE CLUSTER` command, as described below.

`set service cluster <property> <value>`	Changes the value of the cluster property to the value specified.
	
	The valid entries for *property* are:
	`IsClustered`
	`LeadHost`
	`PortNum`
	`MaxHosts`
	`StartupTime`
	
	The new settings take place when the cluster manager restarts (typically when DB2 Alphablox is restarted).

`set service cluster IsClustered <True|False>`	Sets the state of the cluster to *enabled* or *disabled*. When set to true, clustering is enabled. When set to false, clustering is disabled.
`set service cluster LeadHost <lead_host>`	Sets the lead host name or IP address (*lead_host*). The name or IP address must resolve over your network to the machine on which the lead host is installed.
`set service cluster PortNum <port_number>`	Sets the port number (*port_number*) in which the lead host listens for cluster communications. This port must be available on the lead host or the cluster startup will fail.
`set service cluster MaxHosts <maximum_hosts>`	Sets the maximum number of hosts (*maximum_hosts*), including the lead host, allowed in the cluster. Any normal host that attempts to join the cluster after *maximum_hosts* has been reached is denied entry to the cluster.

Command Syntax	Description
`set service cluster StartupTime <startup_time>`	Sets the number of seconds (*startup_time*) in which a normal node attempts to join the cluster upon startup. After the *startup_time* has been exceeded, the normal node host automatically shuts down.
`show hosts`	Lists all the hosts currently connected to the cluster. If clustering is disabled, `SHOW HOSTS` returns nothing.

Chapter 19. DB2 Alphablox Console Commands

DB2 Alphablox can be administered through its Console or from web pages under the **Administration** tab of the DB2 Alphablox home page. Most administrative activities (for example, creating and editing users, groups, etc.) that are configured through the DB2 Alphablox home page user interface can also be configured through console commands. This chapter provides information about using the Console and lists the available Console commands.

Accessing the Console

There are two ways to access the DB2 Alphablox console:

- "HTML Console"
- "Telnet Console"

HTML Console

To access the Console from the DB2 Alphablox home page, click the **Administration** tab, **General** link, **Launch DB2 Alphablox Console** link. The Console HTML page opens.

Administrators can also open the Console in a telnet session, through the Console window that launches if you start DB2 Alphablox from the Windows **Start** menu, and through the window in which DB2 Alphablox started (on Linux and UNIX systems).

Telnet Console

To access the telnet console, start a telnet session to the machine in which DB2 Alphablox is running, specifying the telnet port configured on the **Telnet Console** administration page (23 by default). For example, if you are running DB2 Alphablox on a machine named *pastrami*, enter the following at a command prompt:

```
telnet pastrami 23
```

On Windows machine, the installation process puts a shortcut to the telnet console on the **Start Menu**, **All Programs** group, **DB2 Alphablox** folder, instance name folder (the default value is **AlphabloxAnalytics**), **Console**.

Enter the console user name and password to authenticate to the telnet console.

To end your telnet console session, enter the following command:

```
release
```

Command Syntax

DB2 Alphablox commands use the following syntax:

```
COMMAND object [value(s)] COMMAND object [value(s)] object [value(s)]
```

where:

- object is one of:
  - SERVER (the default)

  - CONSOLE *console ID* (where *console ID* is the ID of a specific active console, for example, C1)
  - the name of a server object. Allowable characters are A-Z, a-z, 0-9, underscore, or special characters (for example, accented characters) when running in a language other than English. The display of the name is case sensitive but the actual name that is authenticated is case insensitive. The names *Public*, *Private*, and *Properties* are reserved and cannot be used for an object name.
- value further qualifies the object

For example, in the following line, *get* is the command, *user* is the name of a server object, and *admin* is the name of a specific user object.

```
get user admin
```

If you enter a command followed by an object, a message is displayed showing the required value(s) for that command. For example, if you enter the following:

```
create data source
```

The following message is displayed:

```
Create data source takes more parameters: data_source_name adapter_name
SERVER server_name [property value]...
```

The property names are the names on the corresponding user interface pages. For example, if you are creating a DB2 OLAP Server or Hyperion Essbase data source, it has property names *application*, *database*, *username*, *password*, *maxrows*, *maxcols*, and *useaasuserauth*.

Command Abbreviations

To speed command entry, the console supports command abbreviations. For example, H is the abbreviation for the HELP command. If two or more commands begin with the same letters, the abbreviation must be long enough to be unique. For example, both the START and STATISTICS commands start with STA; adding the next letter creates a unique abbreviation (STAR for START and STAT for STATISTICS).

Tip: Note these tips on using commands:

- The console supports abbreviations for the entire command line syntax. For example, H C retrieves help for the CREATE command; G U L retrieves the properties for the user whose name begins with L.
- It is not necessary to enclose value strings in quotes, **except** for the name of a database adapter (for example, “ibm db2 olap server”).
- When using abbreviations for general properties, use the first letters of the property names, not their descriptions. For example, for the property name `DefaultMessageLevel` whose description is `Initial Console Message Level`, use an abbreviation such as DEFAULTM.

Console Command List

Most commands can be entered on both the Console screen (reached through the **Launch DB2 Alphablox Console** link on the **General** page under the **Administration** tab) and the DB2 Alphablox command screen. However, a few commands are supported only from one or the other. Additionally, not all commands are supported through telnet. For a list of console commands for the DB2 Alphablox Cube Server, see the *DB2 Alphablox Cube Server Administrator's Guide*. For a list of commands that are specific to clustering, see “Cluster Console

Commands" on page 124. For a list of commands specific to Extensible User Manager, see "Extensible User Manager Telnet Console Command" on page 93.

Note: The following console commands apply to DB2 Alphablox only. For example, if you delete an application from DB2 Alphablox installed on a WebSphere server, the application listing is removed from DB2 Alphablox only. To remove the application from the WebSphere server, you must remove it using the WebSphere Application Server's Administrative Console.

Command and Example	Purpose and Explanation
ADD object value object value add group admin user radams	Adds the second object to the first object Adds the user named radams to the group named admin
CREATE object value(s) create user radams haggis create data source TBC ibm db2 olap server adapter server db2erver1 application demo database basic username user1 password password1 maxrows 1000 maxcols 1000 useaasuserauth false	Creates an object such as a user or group. Creates a new user with values of radams for name and haggis for password Creates a data source named *TBC* that accesses the TBC (demo/basic) application on the IBM DB2 OLAP Server named *db2server1*.
DELETE object value delete user radams delete outlinecache...	Deletes an object from DB2 Alphablox and the DB2 Alphablox Repository. Deleting an application from DB2 Alphablox does not remove the corresponding application from WebSphere server. Deletes the user named radams from DB2 Alphablox. Deletes an entry from the DB2 OLAP Server or Essbase cache.
EXIT	Stops DB2 Alphablox and exits the Console screen. This can only be done from the DB2 Alphablox command screen; it cannot be done from the Console screen reached through the DB2 Alphablox home page.
ExtUserManager	For Extensible User Manager console commands, see "Extensible User Manager Telnet Console Command" on page 93.
GET object value get get user radams	Displays information about an object Displays information about the Server (because Server is the default object) Displays the properties for the user named radams
HELP object help help create	Displays help information for the named command Displays a list of all commands Displays a description and syntax for the create command
KILL object kill 1046976892932726235	Kills the current instance of the named object (usually a user session) Kills the session with the ID 1046976892932726235.

Command and Example	Purpose and Explanation
LOAD object value	Loads an object into the server workspace
load user <userName>	Loads user <userName> into the server workspace
load theme load license	Loads all available themes into the server workspace. (Note that the command does not support loading only a named theme.) Loads (updates) license file (`license.xml`), located in DB2 Alphablox `bin` directory.
LOCK	Locks the local access to the server. To access DB2 Alphablox after entering this command, you need to enter your user name and password. This can only be done from the DB2 Alphablox Console screen and telnet.
MESSAGE object value(s)	Sends a system message to other consoles or log files
message review 20	Sends the last 20 messages to the current console
message review debug 20	Sends the last 20 DEBUG messages to the current console
message C2 Testing	Sends the message "Testing" to console C2
OBJECTS object	Displays information about manageable objects
objects	Displays a hierarchy of manageable objects
objects user	Displays information about users
RELEASE	Releases the remote server console from which this command is entered. This can only be done from the DB2 Alphablox Console screen.
REMOVE object object	Removes the source object from the target object.
REMOVE group admin user radams	Removes user radams from the group admin.
REPORT object report debug	Sets the message level for the current console to DEBUG, VERBOSE, INFO, SYSTEM, WARNING, ERROR, or FATAL. Sets the message level to debug.
RESUME object	Starts the named service from a suspended state.
resume user	Starts the User Manager from the previous suspended state.
RUN object	Runs the command file named by target.
run abc.console	Runs the abc.console command file.
run *createUsers.txt*	Runs the *createUsers.txt* file. (This file might be a batch file containing a series of create user commands for creating new users and establishing their initial passwords. See Note 2 for an example.)

Command and Example	Purpose and Explanation
SAVE object value	Saves the target's properties
save	Saves all General properties now (because General is the default object). Changes entered on the Console are for the current session only and are lost unless this command is issued, or DB2 Alphablox's Save on Exit property is set to `yes`.
save user radams	Saves the properties of user radams
SET object value object value	Sets a value on the object.
set user radams password castle	Sets the password for user radams to castle.
set smtpserver mail	Sets the SMTP server to the mail server named *mail*.
set resolveAliasesToBaseMembers	
SHOW object	Gets information for the target.
show topics	Lists all the topics for which information is available.
show server	Displays all server topics, including users.
show user radams	Displays information about the user named radams.
show theme	Displays theme name and descriptions.
show outlinecache	Displays all the entries in the DB2 OLAP Server or Essbase cache.
show hosts	Displays the name of each machine in the cluster if clustering is enabled.
show	Displays a list of objects. The SHOW command also displays, if it has not already been loaded, the version of the DB2 OLAP Server or Hyperion Essbase client library API loaded.
START object	Starts a service from a stopped state.
start user	Starts the User Manager from the previous stopped state.
STATISTICS object value	Displays available statistics about the target.
statistics user	Displays statistics for all users.
statistics user radams	Displays statistics for user radams.
STOP object	Stops a running or suspended service. Use START to take a service out of a stopped state. **Note:** Do not use the STOP command to stop the Cube Manager unless all of the running DB2 Alphablox cubes have been stopped.
stop user	Stops the User Manager.
SUSPEND object	Suspends a service. For example, suspending the User Manager prevents new user sessions from instantiating. Use RESUME to take a service out of a suspended state.
suspend user	Suspends the User Manager

Essbase-Specific Console Commands

DB2 Alphablox stores DB2 OLAP Server and Hyperion Essbase outlines cached in memory to improve performance. This section describes the syntax of the two commands used to manage the outline cache and the command to resolve alias names to base member names. The following commands are included:

RESOLVEALIASESTOBASEMEMBERS Commands

In a DB2 OLAP Server or Essbase database, you can set up aliases to member names. The `RESOLVEALIASESTOBASEMEMBERS` server property is designed to be used with bookmarks that save queries with aliases that might resolve to different members at different times. For example, if you bookmark a query that references an alias called `This Quarter`, the actual member in the DB2 OLAP Server or Essbase database to which the query refers is different today than it was six months ago. If you want your bookmarks to end up retrieving the current data for this quarter, you must set the `RESOLVEALIASESTOBASEMEMBERS` server property to `TRUE`.

The following is the syntax for the `RESOLVEALIASESTOBASEMEMBERS` console command:

```
set resolvealiasestobasemembers true | false
```

The following command returns the current state of the `RESOLVEALIASESTOBASEMEMBERS` server property:

```
get resolvealiasestobasemembers

ResolveAliasesToBaseMembers .... false
  (ResolveAliasesToBaseMembers)
```

To enable this feature, enter the following console command:

```
set resolvealiasestobasemembers true
```

Tip: You can abbreviate this command by typing in enough letters to ensure a unique name, as follows:

```
set resolve true
```

SHOW OUTLINECACHE Command

The following is the syntax for the SHOW OUTLINECACHE command for DB2 OLAP Server or Hyperion Essbase data sources:

```
show outlinecache [essbasecachemanager [entry [entryname]]]
```

Use this command to display information about the cache manager or the entries it manages. For example, use the following command to show all the entries being managed by the DB2 OLAP Server or Essbase cache manager:

```
show outlinecache essbasecachemanager entry
```

The system responds with the following:

```
Entry
  Entry MDB1.Financial.Current
  ....Total accesses: 7
  ....Current accessors: 1
```

The response shows that the DB2 OLAP Server or Essbase cache manager is currently managing only one entry, named MDB1.Financial.Current. This cache has been accessed a total of seven times (a cumulative number) and is currently being used by one connection.

DELETE OUTLINECACHE Command

The following is the syntax for the DELETE OUTLINECACHE command for DB2 OLAP Server or Hyperion Essbase data sources:

```
delete outlinecache [essbasecachemanager [entry [entryname]]]
```

Use this command to remove the named entry from the cache. DB2 Alphablox will remove the entry only if the current number of connections (accessors) is 0. When an entry is removed, all memory resources associated with the entry are released. For example, use the following command to delete the entry named MDB1.Financial.Current from the cache:

```
delete outlinecache essbasecachemanager entry MDB1.Financial.Current
```

The system responds with the following, showing that named entry has been deleted:

```
MDB1.Financial.Current
Cache entry deleted.
```

The following command and its response confirm that the entry is no longer in the cache:

```
show outlinecache essbasecachemanager

EssbaseCache Manager
  Total entries: 0
```

Notes About Console Commands

This section describes some miscellaneous behavior and provides some tips for using the DB2 Alphablox console. The following topics are included:

Viewing General Properties

Use the DB2 Alphablox Console to view general properties and objects. Most property changes take effect immediately; otherwise, a message indicates that DB2 Alphablox must be restarted before the change takes effect. For a complete list of current DB2 Alphablox properties, use the GET command.

Links at the bottom of the Console provide access to frequently-used information:

- **Help** presents a list of the DB2 Alphablox commands.
- **Status** provides status information on the DB2 Alphablox.
- **Messages** presents a scrollable view of the DB2 Alphablox log.
- **Sessions** presents status information on all current sessions on this instance of DB2 Alphablox.
- **Users** shows a list of users (regardless of whether they are logged on).
- **Services** shows the status of component of DB2 Alphablox. For a list of the services, see "DB2 Alphablox architecture" on page 7.

- **Settings** shows the current setting for each DB2 Alphablox property (both startup and extended).
- **Show** presents a hierarchical list of DB2 Alphablox topics.
- **History** presents a history of commands issued through this Console.

For information on accessing the Console, see "Accessing the Console" on page 127.

Message Levels

Messages generated by the DB2 Alphablox can be sent to all active consoles and the active log file. Message information includes:
- date and time
- message level
- message text
- message source (a service or user name)

The following is a typical message:

```
01/9/99 12:25:33 PM [VERBOSE] Request 4223: Processing request
'/servlet/AnalysisServer/console/consolestyle.css' [Session 41, User radams]
```

In order of severity (lowest to highest), the message levels are:
- DEBUG: debug information.
- VERBOSE: all system messages.
- INFO: minor system events for which no administrator action is necessary.
- SYSTEM: normal system events, such as creating a user.
- WARNING: a recoverable error, but suggests the administrator may want to investigate.
- ERROR: a non-recoverable error.
- FATAL: an error that causes the server to terminate.

Running a Text File Through the Console

Using the *run* command, the Console can execute a plain text file containing any number of DB2 Alphablox commands. For example, suppose you have a file named *d:\CreateUser.txt* containing the following lines:

```
create user radams blue
create user sadams green
create user lplanting purple
create user klawrence yellow
create user dmessink orange
```

You can run this file through the Console with the *run* command to create five new users and establish their initial passwords by executing the following command from the Console:

```
run d:/CreateUser.txt
```

Note: The filename must be a fully qualified name.

DB2 Alphablox Log Messages

Harmless messages like the following may appear in the DB2 Alphablox log file (by default, `Server.log`):

```
[VERBOSE] Request 36: File not found '...\resources\basic_en_US.class'
[VERBOSE] Request 37: File not found '...\resources\basic_en_US.properties'
[VERBOSE] Request 38: File not found '...\resources\basic_en.class'
[VERBOSE] Request 39: File not found '...\resources\basic_en.properties' ...
[VERBOSE] Request 45: File not found '...\resources\metal_en_US.class'
```

```
[VERBOSE] Request 46: File not found '...\resources\metal_en_US.properties'
[VERBOSE] Request 47: File not found '...\resources\metal_en.class'
[VERBOSE] Request 48: File not found '...\resources\metal_en.properties'
...
```

In this example, the JVM is attempting to find the most appropriate resource bundle for the language and location of the client's locale. It is not required for DB2 Alphablox to run properly; these messages can safely be ignored.

Chapter 20. Administering Alphablox FastForward Applications

This chapter describes how you to administer Alphablox FastForward applications, including configuring and basic administration tasks.

Overview

Alphablox FastForward is a sample application framework, preinstalled on DB2 Alphablox, for quickly developing, deploying, and sharing custom analytic views throughout business organizations. Out-of-the-box, the FastForward framework delivers common application services, including security, collaboration, customization, and personalization. Application administrators, typically OLAP administrators, can create new versions of an FastForward application, publish reports by selecting report templates and configuring report parameters, and then deploy the new application without ever looking at code. And, because of its flexibility and extensibility, JSP developers can modify or extend the application framework, and add new custom report templates for application administrators to configure and deploy.

Built into the FastForward application framework are features commonly found in reporting and analytic applications, including:

- exporting to Microsoft Excel
- generation of printable views
- easy saving and sharing of personal views of data
- e-mailing views to others
- easy navigation between different views

Roles of FastForward Users

The three major roles of Alphablox FastForward users include those of application administrators, template developers, and users. A good synergy between these three groups will help ensure the success of FastForward-based applications. More about these three roles are briefly described below.

Application Administrators

Application administrators, typically OLAP administrators, should be able to create new versions of FastForward applications by defining a few settings, create reports based on the available report templates, then quickly deploy solutions to users. If user requirements cannot be met using an existing report template, the application administrator works together with template developers to create new report templates. An application administrator should be able to accomplish their work using their OLAP database experience, the documentation available here (and in the online Administration Help available in the Admin Tasks mode of a FastForward application).

Template Developers

Template developers are typically JSP developers primarily responsible for creating custom report templates when existing ones cannot be used by an application administrator to configure requested reports. In consultation with application

administrators and users, template developers should be able to create new report templates by modifying existing report templates or creating new ones as necessary.

Using the Blox tag libraries, server-side Java API, and client-side JavaScript API, as well as your web programming experience, template developers should be able to create templates for almost every conceivable need. Besides being familiar with building DB2 Alphablox applications and views, developers should also be familiar with the FastForward User Help (available from the Help button in user mode), the Administrator Help (available from the Help button in Admin Tasks mode).

Users

Users, typically business analysts and other line of business users in your organization, should be able to log into a FastForward application and use published reports to analyze business issues. Depending on the interactivity available in a particular FastForward-based application, users can manipulate data, drill around data hierarchies, change chart types, add comments, and more. After modifying views to answer particular business questions, users can preserve their current views, creating saved reports under the Private tab for later use or by sharing them under the Groups tab to defined groups of application users.

For each report, users typically have a few other options available from the application toolbar, located above the reports. Besides saving reports for online analysis, the export to Excel option allows users to export views to Microsoft Excel spreadsheets for offline analysis at a later time. Users can also print a copy of a particular view using the Print Preview option. And, if desired, they can open an e-mail message containing a link to the current view, add comments, and send it to other application users.

If necessary reports are not available in their applications, users typically request new reports directly from application administrators.

System Requirements for FastForward Applications

Other than the requirements specified in the System Requirements section of the Installation Guide, note that Alphablox FastForward works with the DHTML client only.

Creating Alphablox FastForward Applications

New versions of Alphablox FastForward applications are created by the following steps, while logged in with administrator rights:

1. Open your web browser to the DB2 Alphablox Admin Pages. By default, the Applications page appears.
2. On the Application page, launch the Alphablox FastForward application.
3. On the application toolbar, click on the Admin Tasks button.
4. The Template Application dialog window appears, offering two options, Create or Continue. Click on the Create button.
5. In the New Application window that appears, fill out the following fields:

Entry Field	Description
Context Name	Enter an application context name that should be used as the directory name. This name should have no spaces in it.

Display Name	The display name that appears in the Applications page. This name can have spaces in the name.
Description	Optional. A brief description of the application.
Administrator Role	Required. Sets the AdministratorRole value in the application's config.xml file (located in the application's WEB-INF directory). Default setting is `AlphabloxAdministrator`.

Important: Do not configure or use the existing Alphablox FastForward application -- always create a new version. This sample application will be overwritten with each subsequent upgrade of the server, and your modifications will be lost.

Changing Administrator Roles

By default, an Alphablox FastForward application designates the `AdministratorRole` to `AlphabloxAdministrator`. To change the role that can administer a particular application, either set it in the New Application dialog above, or follow these steps:

1. Open the FastForward application's config.xml file, located here:

   ```
   <applicationDirectory>/WEB-INF/config.xml
   ```

2. In the `config.xml` file, change the assigned role in the following section, replacing `AlphabloxAdministrator` with the new role designated to administer this particular application:

   ```
   <param>      <param-name>AdministratorRole</param-name>
   <param-value><![CDATA[AlphabloxAdministrator]]></param-value> </param>
   ```

For information about role definitions and how to create new ones, see Chapter 10, "Role Definitions," on page 55

Administering FastForward Applications

The FastForward Administration application provides three general capabilities:

- managing published reports
- managing the basic look and feel of the FastForward application
- viewing and clearing the application log

Information about administering FastForward applications is also available by clicking Help button on the application toolbar when logged in to the Admin Tasks mode.

Report Access Categories and Security

FastForward applications have three report access categories for users: Published, Private and Group.

Published Reports

Published reports are managed from the administration screen as described below. Published reports, if not otherwise managed, are visible to any and all users who can access the FastForward application. However, the administrator can control access to published reports by setting security on report folders. If a user has

access to a given folder he or she can access any report within that folder. The administrator assigns access to folders by associating a folder with one or more DB2 Alphablox groups.

Private and Group Reports

Private and Group access reports are created by users from previously published reports. Private access reports are only available to the user who created the report. Group access reports are available to users who are members of DB2 Alphablox user groups, and any user who is a member of a given group can create reports accessible to that group.

Layout and Controls

All applications based on Alphablox FastForward have a similar layout structure, including the navigation menu on the left and the application toolbar on the top right. Available reports are viewed in the report window, located below the application toolbar.

Navigation Menu

On the navigation menu, the following selections are available:

Button	Description
Create Folder	Creates a new folder named ″New Folder″. The Edit Folder screen is loaded in order to facilitate changes to the folder.
Create Report	Loads the ″Edit Report″ screen so that a new report can be edited. The report must be successfully saved before it will appear in the left hand report tree.
Delete Report	Pops up a dialog asking you to confirm the deletion of a report.
Copy	Copies the report with ″Copy of″ prepended to the name of the report. You can then rename the file.

Managing Reports

Alphablox FastForward provides basic report administration using the application's user interface while in the Admin Tasks mode.

Creating Reports

To create a report you click on the ″Create Report″ button in the Toolbar. The ″Edit Report screen loads and you can configure the following settings:

Field	Description
Name	The name of the report as displayed in the tree.
Description	This short description is displayed when a user mouses over a report in the tree and when the report is loaded.
Template	The template drop-down list provides the name of every template available to the application. Report templates are stored in the application's `templates` directory. Depending on the specific report template, other parameters will need to be selected.

Save	Save the report. Errors may appear if the report parameters can not be validated.
Preview	Opens the report in a new browser window. Error may occur if parameters cannot be validated.

Modifying Reports

To edit a report, left-click on an existing report in the tree. This loads the ″Edit Report″ screen.

Deleting Reports

To delete a report, left-click on the report to delete, then click the delete report button in the toolbar.

Moving Reports

To move a report:

1. Place the mouse pointer over the report icon next to the report to be moved, press and hold the left mouse button.
2. While holding down the mouse button move the pointer to the desired destination in the tree, release the mouse button

Managing Folders

Folder administration is basic and can be accomplished using the application's user interface.

Creating Folders

To create a folder, click on the ″Create Folder″ button in the Toolbar. The ″Edit Folder″ screen load with the following editable fields:

Field	Description
Name	The name of the report as displayed in the tree
Description	The value displayed when a user mouses over a folder in the tree.
Access Groups	The list of available DB2 Alphablox user groups. Multiple user groups can be selected. If no groups are selected, then all users can access the report. (and a warning is displayed).
Save	Saves the folder.

Modifying Folders

To edit a folder, left-click on an existing report in the tree. This loads the ″Edit Folder″ screen, where changes can be made.

Deleting Folders

To delete a report, left-click on the report to delete, then click the delete report button in the toolbar.

Moving Folders

To move a folder:

1. Place the mouse pointer over the folder icon next to the folder you want to move, press and hold the left mouse button.
2. While holding the mouse button down, drag the folder to the desired destination in the tree, then release the mouse button.

Managing Application Properties

To manage application properties, click on the Global Settings button in the upper right hand corner of the page. The following settings can be modified:

Field	Description
Title	The title of the application. This appears at the top of the user's application as well as at the top of the administration page.
Logo	Displays the name of the file displayed as the logo in the user's application.
Upload Logo Image	Uploads a new logo image from any locally available file (GIF or JPEG format). Note that the upload does not actually occur until you click Save.
Theme	A drop-down list for selecting an installed theme. Note that the sample application is designed only for the `fastforward` theme.
Footer	Changes the text display at the bottom of a user's application page.
Admin Email	Sets the e-mail address used for application feedback. Not setting this address leaves the feedback button disabled in the user's application page.
Save	This saves any changes in the edit fields above.

Using the Application Log

The FastForward Application framework has its own log file to report application-level information and anomalies. This log file can be viewed and cleared by clicking on the View Log File button.

Appendix. OLAP Terms and Concepts

This section illustrates terms and concepts critical to understanding multidimensional analysis (MDA) and OLAP (online analytical processing). Beginning with terms used in two-dimensional analysis, we define their expanded counterparts in MDA. Then we illustrate the related terms used in OLAP databases.

For more OLAP definitions, see also the DB2 Alphablox Glossary and the OLAP Council Glossary.

Two-Dimensional Analysis

Users who perform two-dimensional data analysis with such tools as spreadsheets and reports are familiar with the following terms, which are illustrated in the table that follows:

- A **row** contains a set of related data. The table contains two data rows.
- **Row labels** appear to the left of the data values. The row labels are navy.
- A **column** also contains a set of related data. The table contains four data columns.
- **Column labels** appear above the data values. The column labels are maroon.
- A **data point** (also called a **cell**) is the intersection of a row and a column. Data points have a gray background.
- A **data value** is the element that resides at a specific data point. The numbers 10 through 30 are data values.

A Two-dimensional Sales Table

	Qtr 1 Units	Qtr 2 Units	Qtr 3 Units	Qtr 4 Units
Diet Cola	10	15	20	25
Cola	12	18	24	30

Multidimensional Analysis

In **multidimensional analysis**, these terms and concepts take on additional complexities. MDA involves more than rows, columns, and their intersections. Although the medium for displaying multidimensional data is often a two-dimensional grid, MDA capabilities involve dimensions, hierarchies, members, titles, values, and instances, as well as rows, columns, and data points.

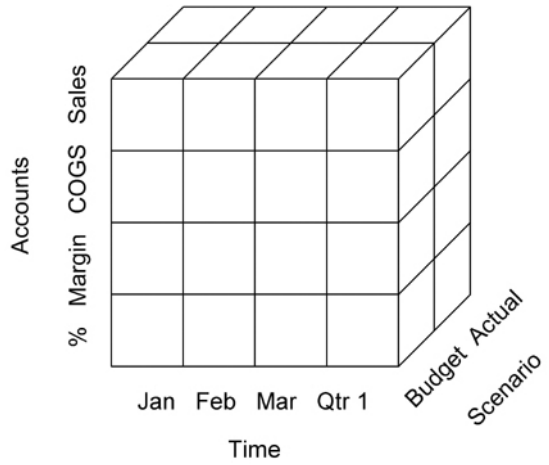
The following definitions apply to multidimensional data and are illustrated in the multidimensional matrix that follows:

- A **dimension** is a structural attribute of a data cube, and is composed of related, hierarchical members. For example, a Time dimension might have the following members: years, quarters, months, and weeks. A Geography dimension might have the following members: regions, countries, and cities.

- A **dimension member** is an element in a dimension. For example, Quarter and Month are members of the Time dimension.
- A **dimension hierarchy** organizes dimension members into parent/child relationships. For example, Month is a child of (belongs to) a Quarter; a Quarter is a child of a Year, and so on.
- A **dimension title** is the name by which the dimension is known, such as Time or Geography.
- A **dimension member title** is the name by which the dimension member is known, such as Month or Region.
- A **dimension member value** is an instance of a dimension member; 1998 is a value for the dimension member Year.
- A **data point** is the intersection of multiple dimensions. Data points in the multidimensional sales matrix below have a gray background.
- A **data value** resides at a data point. For example, in the multidimensional sales matrix, each of the numbers in the grayed area is a data value.

A Data Cube

The illustration shows data formatted into a cube with three dimensions, each with several members:



Dimension	Member
Accounts	Sales, Cost of Good Sold (COGS), Margin, Margin Percent
Time	Quarter 1, composed of: January, February, and March
Scenario	Budget, Actual

A Multidimensional Sales Matrix

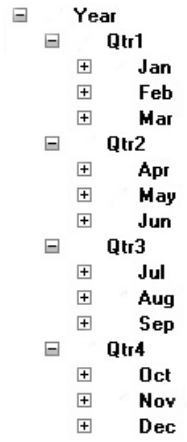
The illustration below shows a hierarchical, multidimensional view of sales data, with two members of the Time dimension (Year and Quarter) appearing on the column axis, and two members of the Inventory dimension (Category and Product) appearing on the row axis.

	Column Axis		
	Qtr 1		
	Jan	Feb	Mar

	Accounts	Budget	Actual	Budget	Actual	Budget	Actual
Row Axis	Sales	300	350	300	325	325	325
	COGS	175	185	175	175	185	190
	Margin	125	165	125	150	140	135
	Percent	42	47	42	46	43	42

OLAP Database Terms

The following terms help to identify, organize, and retrieve data in an OLAP database. For example, when a user drills down from Qtr1, the OLAP action is to retrieve the "children" of Qtr1 (Jan, Feb, and Mar).

Sample Dimension Hierarchy	Term and Definition
	**Root:** The topmost position in the hierarchy. In the example, the root is Year. **Descendant:** A member that is at least one generation below the Root. All the other dimensions in the example are descendants of Root. Jan is also a descendant of Qtr1 and Year. **Child:** A member that is exactly one generation below another. Jan is a child of Qtr1. **Parent:** A member that is exactly one generation above another. Qtr1 is the parent of Jan. **Sibling:** A member that is on the same generation as another. Jan, Feb, and Mar are siblings. **Ancestor:** A member that is one or more generations above another. Qtr1 and Year are both ancestors of Jan.

Note: The capital letter "I" used as a prefix to one of these terms (IPARENT, ICHILD, and so forth) designates "inclusion." For example, on a Parent drill up, the result would show only the parent's data. On an IPARENT drill up, the result would include the data for both the parent and the child.

Glossary

The terms provided in this glossary are relevant to DB2 Alphablox. In addition, refer to the discussion of "OLAP Terms and Concepts," on page 143. More extensive glossaries are available on the Internet, at such sites as A Web of Online Dictionaries and WhatIs.

2D. Refers to a two-dimensional display of information, using rows and columns. Examples include simple lists, reports, and two-dimensional spreadsheets.

3D. Refers to a multidimensional display of information, using page, row, and column dimension. GridBlox presents such a display.

A

ACL. Access Control List. A list of users allowed access to a particular service.

DB2 Alphablox application. A combination of (1) JSP files containing Blox, (2) the access information for data used by the application, and (3) a collection of properties that define how the pages appear to the user. The application resides on a web server and appears in a browser window in response to a user request.

ancestor. A dimension member that is one or more generations above another. See ancestor in OLAP Terms and Concepts for an example.

API. Application Program Interface, the interface used by programmers when writing source code. The API consists of the names of the library calls, and the number and types of arguments they take. In DB2 Alphablox, the API consists of the Blox methods and properties exposed to Java and JavaScript.

applet. See Java applet.

application database. A database that is resident on a server machine and exists to support specific applications. An application database may be extracted from a production database to provide applications with fast access to relevant data. DB2 Alphablox uses multidimensional and relational databases as the source for application data.

Application page. A single HTML page that is part of an DB2 Alphabloxapplication.

authentication. The process of verifying the identity of a person or process. Typically, authentication requires that a user enter a valid ID/password pair.

axis. A coordinate used to organize and display data. An axis can display one or more dimensions. For multidimensional data, an axis is also defined as one edge of a cube.

- A line or bar chart displays data along horizontal (X) and vertical (Y) axes.
- A grid presents multidimensional data using row, column, and page axes. In addition, DB2 Alphablox can display an axis called "Other" that shows dimensions not currently in use. Placing a dimension on the Other axis removes it from the current view, but keeps it available for use.

B

Blox. Reusable software components that facilitate the rapid assembly of applications for intranet/Internet-based, enterprise-wide analysis solutions. For information on how to assemble Blox into a complete application, see the *Developer's Guide*. For detailed information on each Blox, see the *Developer's Reference*.

C

Cascading Style Sheets (CSS). A technology used by web page developers to define the appearance of HTML page elements. Similar to a style sheet used by word processing programs. For more information, see Cascading Style Sheets: Designing for the Web by Hakon Wium Lie and Bert Bos (Addison Wesley Longman 1997), or http://www.w3.org/Style/.

catalog. An organizational level in the structure of a data source. In DB2 OLAP Server or Essbase, this is called an "application." In general, cubes of data are grouped into schemas which are grouped into catalogs.

CGI. Common Gateway Interface. A programming interface used for executing programs on web (HTTP) servers. CGI provides the structure for passing data from the server to the server's gateway program (which does the processing), and for returning the results from the gateway program to the web server back to the requesting client. Contrast with ISAPI or NSAPI.

child. A dimension member that is exactly one generation below another. See child in OLAP Terms and Concepts for an example.

collapse. To consolidate detailed information into a single, summary value. Contrast with expand.

column axis. An orientation for viewing multidimensional data. The column axis is one of three viewable axes (the others being page and row). A column displays the data values for dimension

members. Placing a dimension on the column axis enables users to view data values for members of that dimension. For example, placing the Time dimension on the column axis enables users to view member identifiers (such as Year and Quarter) and associated data values (such as sales volumes).

cookie. An object containing information about a user's session, such as session ID, that is stored on the client machine.

CSS class. A CSS technology used to define an extension or override to an element's normal style.

CSV. Comma-separated values. A file format containing rows delimited by a newline character, and data values within the rows separated by commas.

D

DAA. See Dynamic Application Assembly.

data item. In multidimensional analysis, the value located at the intersection of two or more dimensions.

database connector. A programming interface that provides a common language for accessing a database across a network. Examples include JDBC (Java Database Connectivity) for Java implementations and ODBC (Open Database Connectivity) for Microsoft and Apple Macintosh applications.

data mart. A repository of data gathered from operations and other sources to meet the analysis and presentation needs of knowledge workers. This data may be derived from an enterprise-wide database or from a data warehouse, and is generally presented in terms familiar to a specific group of knowledge workers. A data mart emphasizes ease of access and usability for a particular purpose, such as sales analysis.

data warehouse. A central repository for data collected by an enterprise's business systems. A data warehouse is typically housed on an enterprise server. Data from various online transaction processing (OLTP) applications is extracted and organized for use by analytical applications and user queries.

deprecated. Used with parameters, classes, and methods to indicate they may cease to exist in the future. The need for deprecation occurs as a class evolves and its API changes: parameters and methods are renamed for consistency, and new ones are added. The old API remains until assemblers make the transition to the new one, but once an API is deprecated, assemblers shouldn't use it. DB2 Alphablox writes messages to the client console whenever a deprecated item is encountered.

descendant. A dimension member that is at least one generation below the root (the topmost position in a dimension hierarchy). See descendant in OLAP Terms and Concepts for an example.

dimension. A structural attribute of a cube that helps to define its hierarchy. A dimension lists members, all of which are perceived by the user to be similar types of data. For example, a Time dimension might include members for years, quarters, months, and weeks. A Market dimension might include members for countries, regions, states, and cities. A dimension helps to organize data in a multidimensional data source, enabling multidimensional analysis.

DLL. Dynamic Link Library. An application file that runs as part of the application's process and address space. The DLL files are loaded when an application is started and remain as long as they are needed, therefore eliminating the repeated application retrieval time required for a CGI application.

drill down. To move downward through the hierarchy of dimension members. For example, a user might view financial data for North America, then drill down through US, Western states, California, Northern California, and finally Silicon Valley.

drill up. To move upward through the hierarchy of dimension members. For example, a user might view financial data for Silicon Valley, then drill up through Northern California, California, Western States, and finally US.

DSS. Decision Support System. An information and planning system that supports ad hoc query, analysis, and forecasting using a variety of information sources.

Dynamic Application Assembly. An approach in which applications are quickly assembled from a set of components with well-defined behavior.

E

expand. To show the supporting detail for a summarized value. Contrast with collapse.

extranet. A corporate web site that is accessible to the outside (Internet) world, but only via privilege mechanisms such as registered passwords. Extranets facilitate interaction with partners, customers, and others with a predefined relationship.

F

fat client. A client that performs significant processing using client-resident application software that must be installed on the client. Contrast with thin client.

filter. A way to limit data returned from a query. For example, a filter might specify that the query return only data with a certain value for Year.

frameset. A set of two or more independently controllable sections (frames) in a browser window, each section displaying a separate HTML file. The file displayed in one frame can contain links to files that will appear in another (or the same) frame. Often, one frame contains user tools for navigation and selection, while other frames display the results of user actions.

G

GIF image. Graphics Interchange Format. One of the graphics formats supported by HTML.

grid. A multidimensional display of information that uses row, column, and page axes. A grid lets users drill down (and drill up), slice through, and pivot the data display. GridBlox displays data in a grid.

group. The suppression of repetitive headings for multiple row or column items. Additionally, the automatic generation of total values for data items in the group. Programmers can define a group in relational databases with the SQL GROUP BY clause. Programmers define a group in multidimensional databases through the definition of the database hierarchy.

grouping. When displaying numeric data, the formatting of that data into digit groups (such as thousands), usually marked by a comma or dot.

H

hashtable. A Java data structure that maps keys to values. The following examples are taken from the JavaSoft Java API. The first creates a hashtable of numbers and uses the letters of the alphabet as keys (so that A points to 1, B to 2, and so on). The second retrieves a specific value (2) using its key (B).

```
Hashtable letters = new Hashtable();
numbers.put("A", new Integer(1));
numbers.put("B", new Integer(2));
numbers.put("C", new Integer(3));
```

```
Integer n = (Integer)numbers.get("B");
if (n != null) {
    System.out.println("B = " + n);
}
```

hierarchy. The organization of a dimension's members into parent-child relationships, typically where a parent member represents the consolidation of its children. For example, a hierarchy in a Time dimension might be year, quarter, and month. Each month is a child of a particular quarter, and each quarter is a child of a particular year.

HSB. A color model that describes colors in terms of hue, saturation, and brightness.

HTML. HyperText Markup Language. A text markup language that specifies how the text will appear in a browser window. HTML connects screen text to other text, images, and other objects, such as Java applets, that may reside elsewhere on the Internet.

HTML page. A text document that can contain standard HTML markup, text, graphics, multimedia objects, and Blox. HTML pages are displayed in web browsers.

HTML tag. Language code embedded in an HTML document that tells the browser how to display the document content. HTML tags include a name and optional attributes.

HTTP. Hypertext Transfer Protocol. A set of rules for exchanging files (text, images, sound, video, and multimedia) across the World Wide Web, or across an intranet.

I

instantiate. To create an object of a specific class. For example, when a user invokes an DB2 Alphablox application, DB2 Alphablox instantiates (creates an instance of) that application.

Internet. A worldwide network of computers that use a common set of networking and software protocols. The Internet allows users to share digital information, posted to web pages, across multiple platforms. See also extranet and intranet.

intranet. A corporate network for internal use that provides services similar to those provided by the Internet. See also extranet.

ISAPI. Internet Server Application Program Interface. A set of Microsoft Windows program calls, specific to Microsoft Internet Information Server, that lets you write a web server application that runs faster than a CGI (Common Gateway Interface) application.

J

Java applet. A program written in Java, a language developed by Sun Microsystems. Java applets are embedded in web pages and execute when the browser displays the page.

JavaScript. An object-based scripting language used to extend the capabilities and enhance the user interface of web applications. As with Java applets, JavaScript can be embedded in a web page.

JDBC. Java Database Connectivity. A primary API for relational database access in Java. JDBC is based on the Structured Query Language (SQL) Call-Level Interface. It allows programs to use SQL requests to access databases without using the proprietary interfaces to those databases. A separate module or driver for each

database handles the SQL request and converts it into a request that the individual database system understands.

JPEG image. Joint Photographic Experts Group. One of the graphics formats supported by HTML. To conform to the DOS requirement for a three-character file extension, JPEG files typically have an extension of .eps or .jpg.

JRE. The JavaSoft Java Runtime Environment, which consists of the Java Virtual Machine, Java core classes, and supporting files. It is the runtime component of the J2SE Software Development Kit (JDK). The JRE is the smallest set of executable files and other files that comprise the standard Java Platform.

JVM. Java Virtual Machine. The entity responsible for interpreting and executing Java code.

L

legacy database. A database, typically resident on a mainframe and often using proprietary technology. It is usually a significant information vault for the enterprise.

M

metadata. Data that describes other data. Database schema and data dictionaries contain metadata.

MDA. Multidimensional analysis. The ability to analyze data by navigating through and displaying a subset of a database by querying, drilling down, slicing through, pivoting, and defining calculations to be performed on the data.

middleware. Software that manages the interaction between applications across heterogeneous networks. For example, JDBC drivers are middleware that allow programs to access databases without using the proprietary interfaces to those databases.

MIME. Multipurpose Internet Mail Extensions. Allows computer systems to exchange multimedia information in audio, graphics, video, and text formats using Internet mail standards.

multidimensional database. See OLAP database.

multi-SQL. Multiple SQL statements chained together to form a single, complex query. Related terms include correlated query, subquery, and nested query.

multi-tier access. See N-tier access.

N

N-tier access. The process of accessing data across multiple computing tiers. Simple client-server architectures use two-tier access, with the client requesting data stored on the server, and the server passing results to the client. Intranet architectures often involve a three-tier access, with a client browser requesting data via a web server, which in turn passes the request to a database server. Also called *multi-tier access*.

native SQL. The relational database programming language provided with a given database. Native SQL often includes functions that are optimized for that database. See SQL. Contrast with ODBC SQL.

NSAPI. Netscape Application Program Interface. Enables you to write a web server application that runs faster than a CGI (Common Gateway Interface) application.

O

ODBC. Open Database Connectivity. A database programming interface that makes it possible to access any data from any application regardless of which Database Management System (DBMS) is handling the data. ODBC inserts a middle layer, known as a database driver, between the application and the DBMS. This middle layer translates the application's data queries into commands that the DBMS understands.

ODBC SQL. An open, common programming language for accessing relational databases on a network. To provide nearly universal database access, ODBC SQL may not include some of the functions found in a database's native SQL. See SQL. Contrast with native SQL.

OLAP database. On-Line Analytical Processing database. A database using special indexing techniques to provide fast query processing of summarized data and multidimensional views of that data. DB2 OLAP Server is an example of an OLAP database.

P

page axis. An orientation for viewing multidimensional data. A page defines a slice through a multidimensional database. Placing a dimension on the page axis enables users to select a member from that dimension and view data only for that member. For example, placing the Market dimension on the page axis enables users to select New York from that dimension and view data only for New York.

parent. A dimension member that is exactly one generation above another. See parent in OLAP Terms and Concepts for an example.

persistence. A property of a programming language where created objects and variables continue to exist and retain their values between runs of the program.

pivot. To move a dimension from one axis to another. Assemblers can define a pivot through an Essbase Report Specification or through JavaScript. Users can perform a pivot through the user interface.

proxy server. A server that sits between a client and a server. The proxy server intercepts all requests to the server to determine if it should fulfill the requests itself. If not, it forwards the request to the server. In an Internet/intranet environment, proxy servers can dramatically improve web server performance by intercepting, saving, and serving pages that are frequently requested.

R

report specification. In DB2 OLAP Server or Hyperion Essbase, the definition of the result set to make available to an application. The report specification contains data extraction commands that define the data to retrieve from the database, and formatting commands that define the dimensions to appear on the page, column, and row axes in a grid.

repository. An information database that identifies objects and makes them available for reuse. The DB2 Alphablox Repository stores user and application objects, access control information, and configuration parameters.

RGB. A color model that describes colors in terms of red, green, and blue, the three colors of light which can be mixed to produce any other color.

root. The topmost position in a dimension hierarchy. See root in OLAP Terms and Concepts for an example.

row axis. An orientation for viewing multidimensional data. The row axis is one of three viewable axes (the others being page and column). A row displays the data values for dimension members. Placing a dimension on the row axis enables users to view data values for members of that dimension. For example, placing the Product dimension on the row axis enables users to view member identifiers (such as Product ID or its alias) and associated data values (such as sales volumes).

S

schema. An organizational level in the structure of a data source. In DB2 OLAP Server or Essbase, this is called a "database." In general, cubes of data are grouped into schemas which are grouped into catalogs.

sibling. A dimension member that is at the same generation as another. See sibling in OLAP Terms and Concepts for an example.

slice. To filter a set of data by placing one or more dimensions on the page axis. For example, a cube of sales data may have product on the row axis, month on the column axis, and year on the page axis. Choosing a year slices the data, presenting the data just for the selected year.

SQL. Structured Query Language, which provides a user interface for accessing and retrieving data from a Relational Database Management Systems (RDBMS). See also ODBC SQL and native SQL.

T

TCP/IP. Transmission Control Protocol/Internet Protocol. The basic communication language (protocol) used to connect host machines on the Internet. Also called Transport Control Protocol/Internet Protocol.

tier. A level in a computing hierarchy. Internet/intranet environments often implement a three-tier hierarchy: Tier 1 is a browser running on a client machine; Tier 2 is a web (HTTP) server running on a separate machine; Tier 3 is a database server running on yet another machine.

theme. A collection of design elements, such as fonts and images, that can be applied to an HTML page. Using the same theme across several pages is an easy way to ensure a consistent appearance.

thin client. A client that performs significant application processing, yet has no resident application software. Instead, the software is downloaded at the moment it is needed. A Java-enabled web browser is a thin client. The browser can display web pages and execute embedded applets. Contrast with fat client.

thread. The part of a program that can execute independently of other parts. Operating systems that support multithreading allow developers to design programs whose threaded parts can execute concurrently.

Threadsafe. Code that can have multiple, simultaneous, interleaved or nested invocations that do not interfere with each other, or code that is protected from multiple simultaneous executions by some form of mutual exclusion.

tuple. In DB2 Alphablox, a set of members (one from each of several dimensions) that defines a subset of a cube. In the grid below, tuples on the column axis include South, East, and West; tuples on the row axis include VCR QTR1, TV QTR1, and TV QTR2.

		South	East	West
VCR	QTR 1	10	20	30
TV	QTR 1	20	30	40
TV	QTR 2	30	40	50

U

URL. Uniform Resource Locator. A draft standard for specifying an object, such as a file or newsgroup, on the Internet. URL addresses are used in HTML documents to specify the target of a hyperlink.

V

Vector. An expansible array of objects that are called elements. The number of elements in a Vector can increase or decrease dynamically as program logic dictates. The number of elements is defined by the Vector's size, the amount of storage by its capacity.

W

web server. A server process that sends out HTML pages in response to requests from remote browsers. The term also refers to the machine on which the process is running.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation, Licensing, 2-31 Roppongi 3-chome, Minato-ku, Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation, J46A/G4, 555 Bailey Avenue, San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

DB2

DB2 OLAP Server

DB2 Universal Database™

IBM

WebSphere

Alphablox and Blox are trademarks or registered trademarks of Alphablox Corporation.

Intel® and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, JavaBeans, JavaScript, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W

Program Number: 5724–L14

Printed in USA

SC18-9432-03

Spine information:

IBM DB2 Alphablox Administrator's Guide Version 8.4