# Appeon Migration Guide

Appeon® 3.1 for PowerBuilder®

FOR WINDOWS

DOCUMENT ID: DC37817-01-0310-01

LAST REVISED: September 13, 2005



Appeon Corporation, 1/F, Shell Industrial Building, 12 Lee Chung Street, Chai Wan District, Hong Kong.

# Contents

# 1 About This Book

## 1.1 Audience

This book is written for developers who want to develop new Web applications or deploy their existing Sybase® PowerBuilder® applications to the Web with Appeon® 3.1 for PowerBuilder®.

## 1.2 Chapter descriptions

There are nine chapters in this book:

Chapter 1: About This Book

A general introduction

Chapter 2: Web RAD with PowerBuilder and Appeon

Introduces rapid Web application development using PowerBuilder and Appeon for PowerBuilder

Chapter 3: Migration Process

Describes the methodology of Web conversion based on an existing PowerBuilder application

Chapter 4: Migration FAQ

Frequently asked questions and answers regarding the deployment of PowerBuilder applications to the Web with Appeon 3.1 for PowerBuilder

Chapter 5: Building and Migrating Distributed Applications

Specific instructions for the Web migration of distributed applications

Chapter 6: Migrating PFC applications

Specific instructions for the Web migration of PFC applications

Chapter 7: Database Connection Setup

Instructions on how to set up the EAServer connection cache for various database types

Chapter 8: Enhancing an application with Web or Appeon features

Describes the key enhancement features that you can add in the PowerBuilder application to make them effective in the deployed application

Chapter 9: Web Application Debugging

Introduces methodology for debugging Appeon Web applications

## 1.3 Related documents

Appeon provides the following user documents to assist you in understanding Appeon for PowerBuilder and its capabilities:

- *Appeon Demo Applications Tutorial*:

Introduces Appeon's demo applications, including the Appeon Sales Application Demo, Appeon Code Examples, and the Appeon ACF Demo, which show Appeon's capability in converting PowerBuilder applications to the Web.

- *Appeon Developer User Guide* (or *Working with Appeon Developer Toolbar*)

Provides instructions on how to use the Appeon Developer toolbar in Appeon 3.1.

*Working with Appeon Developer Toolbar* is an HTML version of the *Appeon Developer User Guide*.

- *Appeon Enterprise Manager User Guide*:

Introduces the Appeon Enterprise Manager, a Web application that maintains Appeon Web applications and Appeon Server over the Internet, an intranet, or an extranet.

- *Appeon Supported Features Guide for Appeon Xcelerator* (or *Appeon Features Help for Appeon Xcelerator*):

Provides a detailed list of what PowerBuilder features are supported and can be converted to the Web with Appeon 3.1, using the Appeon Xcelerator deployment option, and what features are unsupported.

*Appeon Features Help for Appeon Xcelerator* is an HTML version of the *Appeon Supported Features Guide for Appeon Xcelerator Deployment*.

- *Appeon Supported Features Guide for Pure-JavaScript* (or *Appeon Features Help for Pure-JavaScript*):

Provides a detailed list of what PowerBuilder features are supported and can be converted to the Web with Appeon 3.1, using the Pure-JavaScript deployment option, and what features are unsupported.

*Appeon Features Help for Pure-JavaScript* is an HTML version of the *Appeon Supported Features Guide for Pure-JavaScript Deployment*.

- *Appeon Installation Guide*:

Provides instructions on how to install *Appeon for PowerBuilder* successfully.

- *Appeon Migration Guide*:

A process-oriented guide that illustrates the complete diagram of the Appeon Web migration procedure, and includes various topics related to steps in the procedure.

- *Appeon Migration Tutorial*:

A tutorial that walks the user through the entire process of deploying a small PowerBuilder application to the Web.

- *Appeon Performance Tuning Guide*:

Provides instructions on how to modify a PowerBuilder application to achieve better performance with its *corresponding Web application*.

- *Appeon Troubleshooting Guide*:

Provides information about troubleshooting issues, covering topics such as product installation, Web deployment, AEM, Web application runtime, etc.

- *Introduction to Appeon*:

Guides you through all the documents included in Appeon 3.1 for PowerBuilder.

- *Using the PowerBuilder Foundation Class Library with Appeon* (or *Appeon-compliant Framework Reference*):

Provides a detailed list of what PowerBuilder PFC features are supported and can be converted to the Web with Appeon, and what features are not supported.

*Appeon-compliant Framework Reference* is an HTML version of the *Using the PowerBuilder Foundation Class Library with Appeon*.

- *What's New in Appeon*:

Introduces new features and changes in Appeon 3.1 for PowerBuilder.

## 1.4 If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support, or an Authorized Sybase Support Partner. If you have any questions about this product, or if you need assistance during the installation process, ask a designated person to contact Sybase Technical Support, or an Authorized Sybase Support Partner based on your support contract. You may access the Technical Support Web site at http://www.sybase.com/support.

# 2 Web RAD with PowerBuilder and Appeon

## 2.1 Overview

Developers can utilize PowerBuilder and Appeon for PowerBuilder to achieve rapid Web application development (Web RAD). Web RAD with PowerBuilder and Appeon greatly reduces the software development life cycle and I.T. development complexity, while preserving end-user productivity.

### 2.1.1 Traditional approach to Web development

In order to write even a small Web application with traditional methods, it takes a great deal of time, and the resulting Web application usually has an oversimplified user interface. Web application developers must master various new skills, such as ASP/JSP/PHP, JavaScript, XML, HTML, Java, and J2EE when using Microsoft or Java technologies.

### 2.1.2 The advantages of Appeon for Web RAD

Using PowerBuilder and Appeon to develop powerful Web applications has the following advantages:

**Provides the fastest path to the Web**

Appeon for PowerBuilder achieves unparalleled reductions in project cycle time and costs by eliminating the traditional Web application design, coding, and testing effort associated with rewriting applications for the Web. When your objective is to build a new application, Appeon provides the most productive development approach for building enterprise-class applications that adhere to the most stringent Web standards.

**Provides the best Web GUI possible**

Appeon Web applications precisely replicate the desktop application user interface with HTML running in standard Microsoft Web browsers. Appeon graphical capabilities make the end user far more productive than any possible rewrite and eliminates all user-retraining time and costs.

**Minimizes business risks**

Appeon for PowerBuilder greatly reduces business risks associated with building new software applications, because the time-tested PowerBuilder program code and DataWindows run at the core of the Web application.

**Leverages current organizational skills**

Appeon for PowerBuilder:

- Relies solely on PowerBuilder skills and IDE to develop and maintain the application.
- Reuses the existing application source code and database, preserves investments in designing, building and testing the application UI, application business logic, data-access logic, and the database.
- Operates upon a single set of native PowerBuilder source code for deployment to both desktop and Web environments.

## 2.2 Leveraged functionality with n-Tier NVO

Appeon's n-Tier NVO support enables you to access enhanced Web functionality. When developing new Web applications with PowerBuilder and Appeon, you can use the n-Tier NVO components in EAServer as a bridge to access many other functions on the Web and some Client Server application features that may even transcend the Web limitations. Refer to Section 5.2: *Moving unsupported features to Appeon Server as n-Tier NVOs* on how to leverage functionality with n-Tier NVOs.

## 2.3 Appeon Web RAD methodology

Web RAD (Rapid Application Development) with PowerBuilder and Appeon is for writing a new PowerBuilder application and converting it to the Web using Appeon for PowerBuilder.

There are three steps in an Appeon Web development project:

STEP 1 – Analyze the project requirements. According to these requirements, lay out the function specifications that should be met for the project.

STEP 2 – Develop a new PowerBuilder application and implement the functions in it with PowerBuilder programming that conforms to the Appeon PowerBuilder coding standards.

During the coding process, utilize the Code Insight tool in Appeon Developer to make sure the new code does not contain unsupported features.

While you develop a new PowerBuilder application, keep in mind one of the more recent and stringent standards, the Section 508 Amendment to the Rehabilitation Act. Refer to the whitepaper available at http://www.sybase.com/detail?id=1035234 for details of the standard.

STEP 3 – Migrate the PowerBuilder application to the Web with Appeon for PowerBuilder by following the migration process described in Chapter 3: *Migration Process*.

Most of the time and effort in Appeon Web RAD methodology is spent working in the PowerBuilder IDE to code the desktop PowerBuilder application.

### 2.3.1 Appeon-standard PowerBuilder coding

Appeon PowerBuilder coding standards are recommended ways to write PowerBuilder code so that it can be recognized and translated into corresponding Web languages (HTML, JavaScript and XML) by Appeon.

The Appeon PowerBuilder coding standards are laid out in *Appeon Help* (a compiled HTML help system). When coding the new PowerBuilder application, you should refer to these documents to ensure the Appeon PowerBuilder coding standards are followed.

After writing the new PowerBuilder application, use the Appeon Developer toolbar integrated into the PowerBuilder IDE to automatically convert the PowerBuilder application to the Web. Some further modifications to the PowerBuilder application may be needed in order to make the application fully functional. Finally, convert the PowerBuilder application and deploy the Web application to servers for production use. For detailed instructions on Web deployment, please refer to the *Appeon Developer User Guide*.

# 3 Migration Process

## 3.1 Introduction

Instead of providing detailed instructions on each step, this chapter emphasizes the overall process. Each section may reference other places within this document or other Appeon documents and support information.

If your goal is to write a completely new PowerBuilder application and deploy it to the Web, please first refer to Chapter 2: *Web RAD with PowerBuilder and Appeon*, and then refer to instructions in this chapter for migrating the new PowerBuilder application to the Web.

Figure 3-1 illustrates the general process of deploying an existing PowerBuilder application to the Web. The process is divided into two phases:

- Phase One: Planning – Evaluating strategies for your Web migration objective.

  STEP 1 – Install required software

  STEP 2 – Understand the general limitations of the Appeon solution

  STEP 3 – Choose the right deployment method

  STEP 4 – Define migration objective

- Phase Two: Implementation – Implementing your Web migration objective.

  STEP 5 – Upgrade the original PowerBuilder application

  STEP 6 – Prepare the target PowerBuilder application

  STEP 7 – Pre-configure for Web application

  STEP 8 – Remove or workaround unsupported features

  STEP 9 – Enhance the application with Web or Appeon features

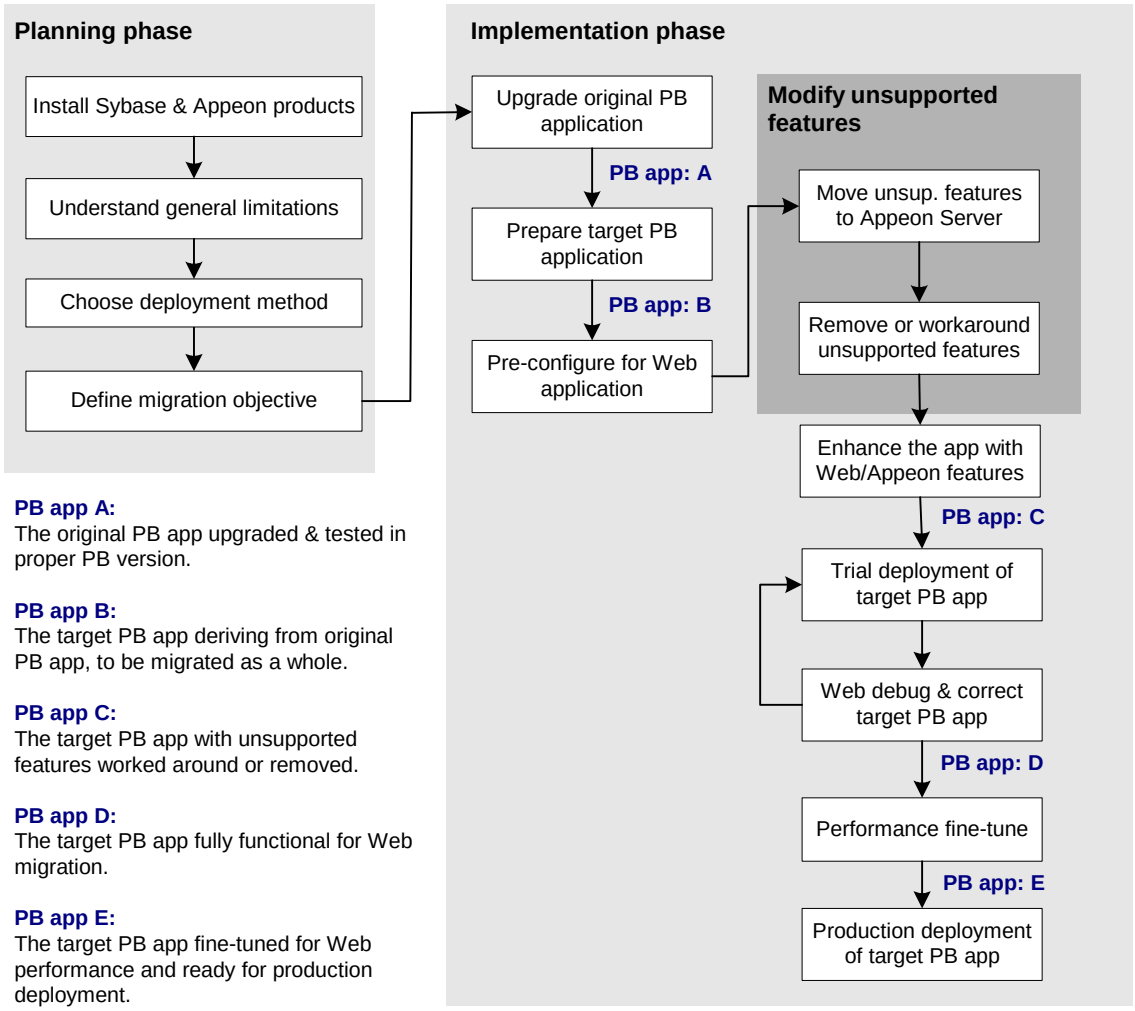
  STEP 10 – Trial deployments

  STEP 11 – Debug Web application

  STEP 12 – Performance fine-tune

  STEP 13 – Production deployment

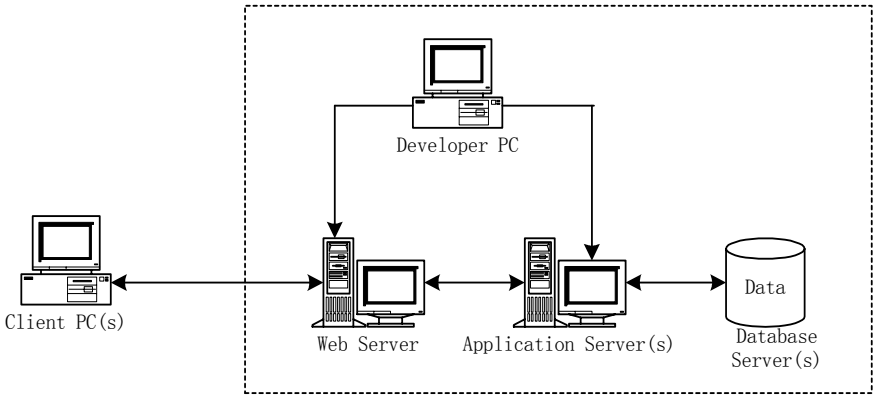
**Figure 3-1: Appeon Web Migration Process**




## 3.2 Installing required software

The first step in the Web migration process is to set up the software environment.

**Figure 3-2: Appeon for PowerBuilder works in a standard n-Tier Web architecture**




The following software should be installed to support the architecture as shown in Figure 3-2:

- Microsoft Windows 2000, Windows 2003 or Windows XP

- Microsoft Internet Explorer 6.0 or above
- Sybase PowerBuilder
- Database that your application uses
- Appeon for PowerBuilder, which includes:
    - Appeon Developer
    - Appeon Enterprise Manager
    - Appeon Server
    - PDFPrinter
    - Sybase EAServer

For detailed instructions on installing the required software, refer to the *Appeon Installation Guide*.

## 3.3 Understanding the general limitations of the Appeon solution

There are some general limitations in the migration capabilities of Appeon for PowerBuilder. These limitations affect how well a PowerBuilder application is suited for Web migration with Appeon. It is very important to be aware of these limitations in advance.

**Table 3-1: Limitations of the Appeon solution**

| Limitation | What it limits… | Try to work around the limit by… |
|---|---|---|
| Application size and complexity | Appeon recommends that the application be 100 MB or smaller for Pure-JavaScript deployment.<br>It covers the file sizes of all framework PBLs, includes the file sizes of all NVOs that are not run on the application server, but excludes the file sizes of all n-Tier NVOs used by the application. | Using Appeon Xcelerator deployment method. |
| External independency | Appeon Pure-JavaScript deployment provides very limited support for external functions and calls to DLLs or external EXE files. | Using Appeon Xcelerator deployment method.<br>Appeon Xcelerator deployment provides more powerful support for external functions and calls to DLLs or external EXE files. |
| Coding style | Appeon recommends not using certain coding styles such as generic code. | Avoiding the coding styles that do not work well with Appeon based on advices in Appeon Help. |
| Database | Appeon for PowerBuilder supports the following database types:<br>Sybase ASE 12.x<br>Sybase ASA 7.0.4, 8.0.2, and 9.0<br>Microsoft SQL Server 2000<br>Oracle 8i and 9i and 10g<br>IBM DB2 UDB 8.1 | Always using the supported database types. |

| | | |
|---|---|---|
| Unsupported features | Appeon Help provides a list of supported and unsupported features. | Working around unsupported features by following the Appeon Workarounds Guide at http://www.appeon.net/support/documents/workarounds.htm. |

For more information regarding the limitations, we recommend you read *Basic and Architectural Requirements* in *Appeon Help*.

## 3.4 Choosing the right deployment method

Appeon 3.1 provides two methods to deploy PowerBuilder applications: Pure-JavaScript deployment, and Appeon Xcelerator deployment. The Appeon Xcelerator deployment method is more powerful and supports more features for Web migration than the Pure-JavaScript deployment method.

Before you decide which deployment method to use, be aware of the following differences between the two methods:

- Pure-JavaScript deployment is excellent for new Web development and Web migration for simple applications. Appeon Xcelerator deployment can be used to migrate large and complex Client/Server-centric applications.
- Appeon Xcelerator deployment boosts the runtime performance of Appeon Web applications to levels approaching PowerBuilder Client/Server via a one-time download of a 1MB signed ActiveX control that is cached on the Client.
- The Xcelerator technology enables Appeon to support more Client/Server features that generally cannot be supported with JavaScript.

Applications with any of the following attributes will benefit the most from Appeon Xcelerator:

- PFC-based applications
- Windows that are heavy with DataWindows (containing 8 or more DataWindows)
- Heavy or complex logic at the Client-side
- Large number of features unique to Appeon Xcelerator deployment (Refer to *Appeon Help* for features difference between the two methods)
- Calling external resources, such as DLL, OLE

## 3.5 Defining migration objective

There are three typical migration objectives:

- Convert an existing PowerBuilder application to the Web.
- Extract a portion from an existing PowerBuilder application, such as a set of Windows and key functions, and migrate that portion to the Web.
- Migrate some DataWindows from the original PowerBuilder application to the Web.

### 3.5.1 Defining migration objective for non-PFC application

STEP 1 – Calculate the total size of the PBLs that make up the original PowerBuilder application.

STEP 2 – Identify the major unsupported features in the existing PowerBuilder application.

STEP 3 – Briefly assess the amount of work needed to modify the unsupported features in the existing PowerBuilder application. The assessment standards are as follows:

- The more complex the PowerBuilder application is, the more difficult it is to modify its unsupported features. The complexity can be characterized by advanced coding techniques including deep inheritance, etc.

  If there is a large amount of business logic on the Client-side (e.g. in Windows and behind controls) and the logic is complex (e.g. it fires 20 events before the business rule is completed), then your application will most likely benefit from moving the business logic into n-Tier NVOs. This takes additional work and needs to be factored in.

- The more unsupported features the PowerBuilder application has, the more effort it takes to modify them.

  Section 3.9: *Modifying unsupported features* provides methods on working around unsupported features.

STEP 4 – Decide the migration objective:

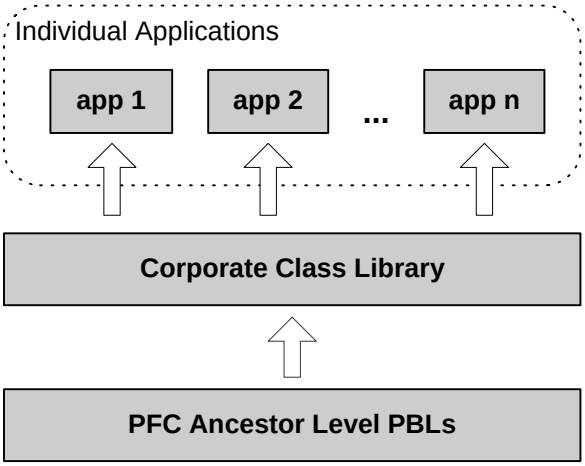
- If it is imperative that your company move business to the Web quickly, select the first migration objective and deploy the entire PowerBuilder application to the Web; otherwise select the second or third migration objective.
- Based on the assessments in the previous step, if substantial effort is required to modify the unsupported features in the original PowerBuilder application, you may want to consider converting part of the application to the Web (objective 2). Regardless of the approach taken, it will always take less work to deploy a PowerBuilder application to the Web with Appeon than to rewrite it with J2EE/.NET, as PowerBuilder is highly productive, especially for your PowerBuilder team.
- The third possible objective is to deploy only your application DataWindows to the Web (separately from the application). This involves building a much smaller new PowerBuilder application and reusing the existing DataWindows. This is a logical solution when some key reporting functionality or data needs to be available on the Web in a very short time and it would otherwise take significant effort to deploy the entire application.

### 3.5.2 Defining migration objective for PFC application

#### 3.5.2.a Corporate PFC architecture

The following diagram illustrates the typical architecture for PFC applications.

**Figure 3-3: Typical PFC applications architecture**




The Corporate Class Library extends the PFC Ancestor Level, and contains customized and enhanced functionality for reuse by various applications within the entire corporate class library. The PFC Extension Level PBLs are included in the Corporate Class Library, and utilized to extend the PFC Ancestor Level. For example, corporate analysts may add intermediate extension level(s) between the PFC Ancestor Level and the Extension Level, or they can use the existing PFC Extension Level.

Depending on the business need and corporate complexity, sometimes the PFC Ancestor Level is extended further into several layers containing corporate departmental standards and business rules. These several layers are all included in the Corporate Class Library.

### 3.5.2.b Defining migration objective progressively

If your PFC application's architecture has a large corporate layer, Appeon recommends that you first aim to migrate a small application to the Web or portion of your existing PFC application to get your framework working on the Web, then expand to deploy a larger portion of your application or the entire application. This progressive approach prevents you from being confronted by many problems at the same time; otherwise, you may be overwhelmed with unsupported features that exist in both the application and the Corporate Class Library.

STEP 1 – Deploy a small PFC application that consists of the following three parts:

- A small amount of application-specific logic separated from the Individual Application.
- Corporate Class Library
- PFC Ancestor Level

In this step, the application-specific logic should be kept small and simple so you can focus on migrating the PFC Ancestor Level and the Corporate Class Library onto the Web. The PFC Ancestor Level and Corporate Class Library are more important because they are the base classes for all corporate applications.

STEP 2 – Gradually increase the size and complexity of Individual Applications, and change your focus to fixing the Individual Applications for Web deployment.

## 3.6 Upgrading the original application

Appeon 3.1 supports PowerBuilder 8, 9 and 10. Any PowerBuilder source code that is intended for deployment to the Web MUST first be upgraded to be 100% compatible with the required PowerBuilder version. If your application is not upgraded, you will encounter runtime errors with the deployed Web application.

### 3.6.1 Upgrading obsolete PBLs

When loading PBLs from a previous version (e.g. PowerBuilder 5) into a later version, many error messages may appear in the PowerBuilder Output window. Since inheritance propagates are an issue from the ancestor object to all its descendent objects, some of the errors reported may actually come from a single source.

One common type of error found when upgrading PowerBuilder applications is string corruption. (for example, strings like “Welcome!” and “db_para.ini” coded in PowerBuilder 5 may become “? and “db_p?, missing the end quotation marks when upgraded to PowerBuilder 8). To correct these errors, edit the source code (by selecting *Edit Source* from the context menu on the error item displayed in the Output window). Errors will disappear when performing a Full Build to the application target.

When an application is upgraded from PowerBuilder 5 to 8, DataWindow columns may not be given proper column names, but the application may still run correctly in PowerBuilder. These columns do not display any data when deployed to the Web. Manually assign names to problematic columns.

### 3.6.2 Upgrading PFC applications

PFC applications intended for Appeon Web migration **must** be upgraded to be compliant with PFC 8.0. Some legacy PFC applications are based on the PFC 5.0 PBLs that have been migrated to the newer version, and you should upgrade those applications to use the new PFC 8.0 PBLs.

The PFC PFCOLD.PBL library contains obsolete objects that are not supported for Appeon Web migration. If the original PowerBuilder application is a PFC application that uses objects in the PFCOLD.PBL library, you should remove the objects when upgrading the application to PFC 8.0.

For more information on upgrading PFC applications, please refer to appropriate Sybase PowerBuilder PFC documentation.

## 3.7 Preparing the target application

In this step, you need to process the original PowerBuilder application to create the target application. This work focuses on transforming the original PowerBuilder application into a target application that complies with Appeon architectural guidelines set in *Appeon Help*.

### 3.7.1 Special processing required for PFC applications

Appeon provides the Appeon-compliant Framework (ACF) as the solution to migrate existing PowerBuilder PFC applications. Refer to Section 6.2.7: *Steps to migrate a PFC application* for instructions on replacing the PFC framework of the original application with ACF framework.

### 3.7.2 Processing application based on migration objectives

The following table shows what processing tasks you need to perform based on your migration objective.

**Table 3-2: Process application based on migration objectives**

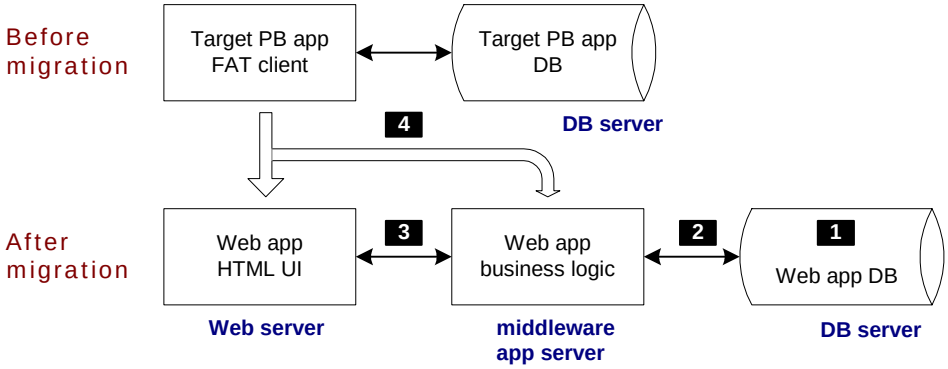
| If you plan to migrate… | Do the following… |
|---|---|
| Entire original application | 1. Test the original PowerBuilder application and correct any applications functionality or user interface problems.<br>Note: This step is for detecting and removing problems that may have existed in legacy PowerBuilder applications or problems caused by upgrading legacy PowerBuilder applications.<br>2. Perform a full build and optimize the original PowerBuilder application in PowerBuilder IDE. |
| A portion of the original application | 1. Extract the desired portion from the original PowerBuilder application into a new PowerBuilder application target.<br>2. Test the application to ensure there are no bugs and that it functions as expected.<br>3. Perform a full build and optimize this "extracted" portion of your application in the PowerBuilder IDE. |
| Some DataWindows in the original application | 1. Create a new PowerBuilder application target in the same workspace that holds the original PowerBuilder application.<br>2. Move the desired DataWindow objects from the original PowerBuilder application to the new PowerBuilder application target.<br>3. Add Windows, Menus, and a general UI for the application. Code simple business logic to the new PowerBuilder application in order to make it fully functional.<br>4. Test the new PowerBuilder application and correct any problems with its functionality and user interface.<br>5. Perform a full build and optimize the new PowerBuilder application in the PowerBuilder IDE. |

## 3.8 Pre-configuring for the Web applications

### 3.8.1 Why is pre-configuration necessary

The following illustration of the Appeon migration solution helps explain why these pre-configurations are necessary.

**Figure 3-4: Before and after Appeon Web migration**




Before the Web migration, the heavy Client of the target PowerBuilder application interacts with the Database Server. If the application is distributed, the Client also interacts with the application server that hosts and executes important business logic for the target application.

After the Web migration, the Web application has an n-Tier Web architecture involving Appeon Server as the middleware to hold business logic. This structure is called Browser Server, because the Web application is accessed via Web browsers such as Internet Explorer.

Making the prospective Web application function in an n-Tier Web environment requires more than just hosting the HTML user interface and the business logic on the servers.

### 3.8.2 Four pre-configuration tasks

As indicated by the numbers in Figure 3-4, four pre-configuration steps are necessary:

Task #1: Manually set up the Web application database server (non-Appeon task). This task is no different from setting up the PowerBuilder application database server.

Task #2: Manually set up communication between the back-end DB server and the middleware Appeon Server. Specifically, set up EAServer JDBC connection caches (non-Appeon task).

Refer to Chapter 7: *Database Connection Setup* for instructions on this task.

Task #3: Setup communication between the Web application UI (hosted on the Web Server) and the Web application business logic (hosted on Appeon Server). This is done via Database configuration for the application. (Appeon task)

Refer to Chapter 7: *Database Connection Setup* for instructions on this task.

Task #4: Configure Appeon Developer so that it has sufficient information to automate the task of converting and deploying the target PowerBuilder application. (Appeon task)

Refer to the Appeon Developer User Guide for instructions on this task.

## 3.9 Modifying unsupported features

Some PowerBuilder features cannot be supported because:

1. There are architectural differences between desktop applications and Web applications. The functionality in a Web application is often limited with regards to desktop applications.
2. The current version of Appeon for PowerBuilder does not support every PowerBuilder programming feature.

Unsupported features, if not modified, will be commented out in the generated Web files. The code that contains the unsupported features and other code that is dependent on those unsupported features will stop working.

This step involves modifying the unsupported features that have some functional impact on the running of the application. Some cosmetic features, such as "Border" property, can be ignored during the modification process if they will not affect the application.

### 3.9.1 How to identify unsupported features

Four primary sources of information can guide you in identify Appeon-unsupported features in the PowerBuilder applications:

- Unsupported Features Analysis (UFA) report – This tool is included in the Appeon Developer toolbar. It scans the PowerBuilder application for its unsupported features and assists you in making changes to unsupported features. For instructions on how to use the UFA, please refer to the *Appeon Developer User Guide*.
- Code Insight – This tool is also included in the Appeon Developer toolbar. It helps you directly identify unsupported code in the PowerBuilder painter. For instructions on how to use the Code Insight, please refer to the *Appeon Developer User Guide*.
- *Appeon Help for Pure-JavaScript Deployment* and *Appeon Help for Appeon Xcelerator Deployment* – These two Help files are searchable HTML files that can be launched by clicking the *Appeon Help* button in the Appeon Developer toolbar. They list the supported and unsupported features in detail and provide recommendations for writing convertible PowerBuilder code following the Appeon coding standards.

### 3.9.2 Feature modification methods

You can modify unsupported code by following the instructions in the *Appeon Workarounds Guide*, which is available at http://www.appeon.net/support/documents/workarounds.htm. The Guide provides examples of some common unsupported features and ways to work around them.

The following two modification methods are included in the *Appeon Workarounds Guide*, and worth highlighted here because of their importance:

- Encapsulating unsupported features into PowerBuilder non-visual user objects (NVOs) and deploying the NVOs to Appeon Server.

  This method can work around a vast number of unsupported features (both browser limitations and Appeon limitations). Refer to Section 5.2: *Moving unsupported features to Appeon Server as n-Tier NVOs* for instructions on this method.

- Using the Appeon workaround for distributed DataWindows

  Appeon Workaround PBL provides two objects *appeondatawindow* and *appeondatastore*, and four functions GetFullState, SetFullState, GetChanges and SetChanges, for supporting distributed DataWindows. Refer to Section 5.4: *Migrating distributed applications with distributed DataWindows* for instructions on this method.

## 3.10 Enhancing the application with Web or Appeon features

This is an optional step. Some enhancement features can be automatically included in the deployed application, but some features require you to make changes in the PowerBuilder application first. The following lists the key enhancement features that you should add in the PowerBuilder application to make them effective in the deployed application:

- Appeon Server open interfaces that can be called to manage all the applications run at Appeon Server
- Appeon client functions that can be called to get client information or enable Appeon DataWindow menu
- Integration with other applications via CommandLine argument, or HyperLink controls

Chapter 8: *Enhancing an application with Web or Appeon features* provides you instructions on how to implement a number of Web or Appeon features in the deployed application.

## 3.11 Trial deployments and debugging

Having completed the previous steps, the target PowerBuilder application is ready for the first trial deployment.

Trial deployments are the intermediate deployments before the final production deployment. For detailed instructions on deployment and running of Appeon Web applications, please refer to the *Appeon Developer User Guide*.

### 3.11.1 Special deployment steps for distributed applications

Special deployment steps are required for distributed applications with or without distributed DataWindows. Refer to Chapter 5: *Building and Migrating Distributed Applications* for instructions on deploying distributed applications.

### 3.11.2 Debugging deployed applications

There may be issues in the trial deployment. For example, a feature in the Web application may work differently from the target PowerBuilder application or not work at all. These issues are usually caused by unsupported features or known issues (documented in the *Appeon Release Bulletin*). Since UFA cannot detect every single different feature between Client/Server application and Browse/Server application, it is possible that your application contains an unsupported feature.

It is essential to debug the Web application and find the cause of the problem. Refer to Chapter 9: *Web Application Debugging* for instructions on debugging.

After modifying the target PowerBuilder application according to the debugging result, perform another trial deployment to see if the Web errors still exist in the Web application. The "Trial deployment → Debug Web application" process may need to be repeated many times before the Web application functions properly.

## 3.12 Fine-tuning the runtime performance

Given the architectural differences between Client/Server applications and Web applications, each of these has advantages and disadvantages. Typically, Web applications offer better

server scalability, while Client/Server applications provide a superior user experience (i.e. rich GUI) and better runtime performance.

Appeon Web applications offer the rich PowerBuilder GUI with the scalability of n-Tier Web architecture. The rich PowerBuilder GUI ensures that the superior user experience and high user productivity is preserved. The server scalability is comparable to J2EE applications, if not better, since Appeon moves more processing from the server to the Client. The *Appeon Performance Tuning Guide* has laid out the PowerBuilder programming features and coding styles that lead to poor Web performance, and it provides suggestions/workarounds. Please refer to the *Appeon Performance Tuning Guide* for more information.

## 3.13 Production deployment

The production deployment is the last step in the Web migration process. This step involves the deployment of the target PowerBuilder application to your production servers and makes the Web application available to your general users.

Make the following settings in the Appeon Developer Application Profile before starting the final deployment to your production servers:

- Set the deployment mode to Encrypted – This encrypts the JavaScript code in the Web application, thereby protecting your organizations business rules and intellectual property.
- Turn off all Report options – This will reduce the size of Web files and boost the runtime performance of the deployed Web application.

Refer to the *Appeon Developer User Guide* for detailed instructions on how to perform application deployment.

# 4 Migration FAQ

## 4.1 How do I rapidly build a new Web application with Appeon for PowerBuilder?

STEP 1 – Create a new application.

Quickly create a new PowerBuilder Client/Server application workspace in your PowerBuilder IDE.

STEP 2 – Code for the application.

Enable the Appeon Code Insight that is included in Appeon developer toolbar, and then write code. With the Code Insight, you can get instant information on whether the new code is supported by Appeon.

STEP 3 – Migrate the application automatically with Appeon Deployment Wizard.

After previous steps, your PowerBuilder application now contains features that are 100% supported by Appeon. You do not need to modify any unsupported features. You can easily migrate your Client/Server application to Browse/Server application with the deployment wizard in Appeon Developer.

## 4.2 Does Appeon support every PowerBuilder feature?

The current version of Appeon does not support all PowerBuilder syntax. Unsupported PowerBuilder syntax falls into three categories:

Category 1: PowerBuilder features that are Client/Server architecture specific, and cannot be implemented in n-Tier Web applications. Appeon Xcelerator can support some Client/Server features that are unsupported with Pure-JavaScript.

Unsupported features in this category must be reworked for n-Tier architecture.

Category 2: PowerBuilder features that cannot be implemented on the Web due to limitations in Internet Explorer or standard Web technologies. For example, Web applications cannot read files from the local machine without using Java Applets, ActiveX, or a plug-in.

Note: Most features in this category are unsupported in the Web application deployed in Pure-JavaScript, but supported in Appeon Xcelerator.

Category 3: PowerBuilder features that are currently unsupported by Appeon, but will possibly be supported in the future.

Section 3.9: *Modifying unsupported features* gives you high-level instructions on how to modify unsupported features. *Appeon Workarounds Guide* at http://www.appeon.net/support/documents/workarounds.htm provides examples on some common unsupported features and ways to work around them.

## 4.3 Do Appeon-deployed Web applications support external resources?

Appeon supports using external resources in Web applications. However, the support capabilities of Appeon Xcelerator and Appeon Pure-JavaScript deployments differ somewhat.

For Appeon Xcelerator deployment, most usages of external resources are supported, such as

INI and image files, OCX and OLE controls, etc.

For Appeon Pure-JavaScript deployment, the usage of external resources is very limited (for example, OCX and OLE control are unsupported). DLL controls are supported, but the usage of them is very limited.

Appeon Xcelerator deployment supports more features of external resource than Appeon Pure-JavaScript does. If you use many external resources in your PowerBuilder Client/Server application, Appeon recommends that you deploy your application with Appeon Xcelerator.

## 4.4 Why classify my application into types?

The PowerBuilder-to-Web process can be simple or complex depending on the type of application being migrated. Appeon provides guidelines for different types of applications. Once you have identified the type of application, follow the specific guidelines to get your application on the Web even faster.

## 4.5 What are the different application types?

PowerBuilder applications can be classified as one of three types.

Type 1 applications meet the following requirements and can be automatically deployed to the Web:

- The application does not contain any functionality that cannot be implemented on the Web.
- The application does not contain any unsupported features.
- These types of applications tend to be below 50 MB (application PBLs including Framework).

Type 2 applications contain some functionality that cannot be implemented on the Web, and/or some unsupported features. The size of the application may be large, and/or the user objects in the application may have complex interdependencies. Modify the objects or code in the application and make it compliant with *Appeon Help*.

Type 3 applications contain functions that cannot be implemented on the Web but are critical to the functionality of the application and have very complex frameworks that do not align well with Appeon supported features, such as PFC-based applications. Depending on the business requirements and project scope, the developer can leverage the RAD capabilities of PowerBuilder to get an application on the Web much faster than a typical J2EE or .NET rewrite while preserving the rich PowerBuilder GUI.

## 4.6 What are the recommendations for converting the different application types?

The following are recommendations for handling the different types of applications.

Type 1 applications: This type is ready to be migrated. There is no additional work to do. Appeon reads the PowerBuilder PBLs and generates the n-Tier Web application. The Web application will have DataWindows, business logic, and a UI identical to the original PowerBuilder application.

Type 2 applications: The developer needs to remove the unsupported features and Client Server specific functions from the application that cannot be implemented on the Web to ensure that the Web deployed application functions in the same manner as the PowerBuilder application. If the application is large and complex, Appeon suggests splitting it into smaller applications to simplify the debugging and deployment of the application. Remove as many minor and unnecessary features as possible. Simplify the code and object interdependencies to make the application more straightforward. The DataWindows and a good portion of the PowerBuilder source code can be utilized. The amount of PowerBuilder source code that can be reused depends on the actual application's complexity and the user's requirements.

Type 3 applications: These generally cannot be migrated to the Web as-is, but the DataWindows and other useful objects can be exported and used to rapidly build a more straightforward PowerBuilder application using standard PowerBuilder programming.

The ultimate aim of the guidelines is to help you modify the PowerBuilder application and make it well supported by Appeon. The existing DataWindows may be used. Depending on the application type, modify a fraction, a part, or all of the PowerBuilder source code.

## 4.7 What are the basic requirements for rewriting complex applications?

*Appeon Help* outlines a set of requirements and recommendations that should be followed. See the first two sections of *Appeon Help* for more information. Essentially, you must re-code portions of the application to make the application more straightforward (by, for example, removing complex object interdependencies and reducing the number of DataWindows and DataStores in a given Window).

## 4.8 When would I need to modularize my application?

If the application is a Type 2 or Type 3 application, you may need to split the application into smaller applications and deploy them separately depending on the actual migration requirements.

## 4.9 What are the benefits of modularizing my application?

The following are advantages of splitting an application into smaller applications:

- Smaller applications have better performance when deployed to the Web.
- It is easier and more efficient to debug smaller applications and rework the unsupported features.
- It allows several developers to work on one project simultaneously and improves developer productivity because one or more developers can specialize on particular modules. The modules can each be deployed to the Web and linked together through a unified HTML interface.

## 4.10 What are the basic principles for modularizing an application?

The following are the basic principles for splitting an application into smaller applications (modularizing):

The functionality of each small application should be independent from the others. Each application to be deployed should be able to execute an independent and practical task. Following this principle, it is better to integrate closely related functional points in a small application and remove the loose object references. For example, to split a purchase order application into smaller parts, the order management logic (module) can be placed in a small application and the relevant supplier information management logic in another small application.

On the basis of the first principle, try to balance the complex objects evenly across all the small applications.

The applications should be as small as possible.

## 4.11 Can you give an example of the modularization process?

The following example highlights the process of modularization of an application:

A purchase order application needs to be deployed to the Web. It is a large and complex application that provides the following functionalities at a high-level:

- Order Input
- Order Management
- Supplier Management
- Resource Planning
- Reporting

According to business requirements, the most urgent application functions that must be brought on to the Web as soon as possible are Order Input, Order Management, and Reporting.

Based on the above business requirements, the first phase of the conversion process should involve extracting the Order Input, Order Management and Report functionalities from the original application and importing them back into PowerBuilder. This creates a new smaller application.

Using *Appeon Help* and Appeon Code Examples, you can work around or modify the unsupported source code in the smaller application to bring it into compliance and deploy it to the Web.

## 4.12 How can I transfer an n-Tier NVO from one EAServer to another?

STEP 1 – Zip or copy the following two files about the n-Tier NVO from the *%JAGUAR%\Repository\Component\PackageName* directory with the sub-directories.

- The component property files (*.props)
- The Component PBD files

STEP 2 – Unzip or paste the files obtained in STEP 1 to the *%JAGUAR%\Repository\Component\PackageName* directory in the new EAServer.

STEP 3 – Go to EAServer Manager and generate stubs and skeletons for the components.

## 4.13 How can a Web application load without an Internet Explorer menu?

When a Web application is opened in Internet Explorer, the user will see both the Internet Explorer menu and the Web application menu in most cases. It is possible to remove the Internet Explorer menu using one of the following two methods.

### 4.13.1 Method 1: Create a JavaScript file for loading the Web application

This method only works for applications that do not apply the 10X Web file compression feature.

Write an Open method in the JavaScript file. The following is an Open method script example:

```
function startApp() {
      g_newWindow = window.open("index.html", "_blank",
"location=no,titlebar=no,toolbar=no,menubar=no,status=no",false);
}
```

Note: The *sName* argument in the Open method must be "_blank", i.e. the URL displayed must use a new Internet Explorer browser. If the URL still opens in the existing Internet Explorer, the *sFeatures* settings will not be effective.

The JavaScript file is also stored in the Web Server.

The browser first loads the JavaScript file into Internet Explorer. When the Open method of the JavaScript file is triggered, a new Internet Explorer browser opens and loads the Web app. The display mode for the new Internet Explorer browser can be specified in the Open method script.

### 4.13.2 Method 2: Create an HTML file for loading the Web application

This method only works for applications that apply the 10X Web file compression feature.

STEP 1 – Create a new HTML file such as index1.htm, and code the following in the file (assuming the deployed application is *test*):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> Appeon </TITLE>
</HEAD>

<BODY>
<script>window.open("/gzip/test/index.html", "_blank",
"location=no,titlebar=no,toolbar=no,menubar=no,status=no",false);<
/script>
</BODY>
</HTML>
```

STEP 2 – Copy the generated index1.htm to UNIX system and zip it to be index1.htm.gz.

STEP 3 – Copy the index1.htm.gz file back to the %JAGUAR\html\gzip\test directory of Appeon Server ("test" stands for the name of the application), and modify the file name to index1.htm.

Now the application can be loaded without an Internet Explorer menu with the URL http://localhost:9988/gzip/test/index1.htm (not http://localhost:9988/gzip/test/index.htm).

### 4.13.3 Method 2: Create a C++ program that utilizes COM API at the Client side

This method works for applications regardless of whether they apply the 10X Web file compression feature.

The sample code in the C++ program is as follows:

```
// Start a new Internet Explorer as a separate process
   IWebBrowser2* pIE = NULL;
   HRESULT hr;
   hr = CoCreateInstance(CLSID_InternetExplorer, NULL,
CLSCTX_SERVER,
         IID_IWebBrowser2, (LPVOID*)&pIE); // if open IE OK
    if (SUCCEEDED(hr))      {
      pIE->put_Visible(TRUE);
      pIE->put_AddressBar(FALSE);
      pIE->put_MenuBar(FALSE);
      pIE->put_StatusBar(TRUE);
      pIE->put_ToolBar(FALSE);
      pIE->put_FullScreen(FALSE);
      COleVariant vtEmpty;
      CString strURL = "http://appeonserver:81";    //The URL to
be opened by the program.
      BSTR bstrURL = strURL.AllocSysString();
      pIE->Navigate(bstrURL, &vtEmpty, &vtEmpty, &vtEmpty,
&vtEmpty);
      ::SysFreeString(bstrURL);      }
```

If the end user runs the C++ program in the Client machine, the Web application will be opened in an Internet Explorer browser and the display mode of the Internet Explorer browser is specified in the C++ program.

# 5 Building and Migrating Distributed Applications

## 5.1 Overview

Appeon supports migrating distributed applications regardless of whether they contain server DataStores. If your non-distributed application is large and contains complex logic or many unsupported features, consider building a distributed application out of it by moving a portion of business logic to the server.

The migration process for distributed applications is essentially the same as outlined in Chapter 3: *Migration Process*. However, you should carefully read the instructions in this chapter, and perform special preparation or deployment process to ensure successful migration. Refer to:

- Section 5.2: *Moving unsupported features to Appeon Server as n-Tier NVOs* for details on the advantages, restrictions, and guidance of n-Tier NVO usage in distributed applications.
- Section 5.3: *Migrating distributed applications without distributed DataWindows* for special preparation tasks required for migration of distributed applications without distributed DataWindows.
- Section 5.4: *Migrating distributed applications with distributed DataWindows* for special workaround techniques for supporting the GetFullState, SetFullState, GetChanges and SetChanges functions.

## 5.2 Moving unsupported features to Appeon Server as n-Tier NVOs

### 5.2.1 Strategy

When an application contains unsupported features, you can encapsulate some of them into PowerBuilder non-visual user objects (NVOs) and deploy the NVOs to Appeon Server. These NVOs can be called by Appeon Web applications as well as any other Web application or PowerBuilder application.

This n-Tier NVO support allows the user to program powerful PowerBuilder features into the application. If an existing PowerBuilder application is deployed to the Web, the n-Tier NVO technique can work around a vast number of unsupported features (both browser limitations and Appeon limitations).

### 5.2.2 Advantages of n-Tier NVO usage

With n-Tier NVOs, you can work around:

- A large amount of unsupported DataWindow (DW) syntax (functions, events, properties, dot notation and expressions) in conjunction with the Appeon distributed DataWindow technique (support of DW Get/SetFullState, Get/SetChanges).
- Some unsupported non-visual PowerBuilder system objects
- Some unsupported system functions
- Dynamic SQL statement Format 4

N-Tier NVOs can also:

- Remove Web browser limitations by running PowerBuilder NVO code inside Appeon Server
- Connect to DLLs
- Connect to other PowerBuilder NVOs & Java/EJBs in Appeon Server/EAServer.
- Connect to EJBs in BEA WebLogic, IBM WebSphere, and other J2EE-compliant application servers through PowerBuilder 9.0 PBNI/EJB support
- Connect to remote Web Services or .NET components
- Create and expose Web Services from PowerBuilder NVOs using the Appeon Server/EAServer Web Services Toolkit
- Connect to C++ Classes/DLLs through PBNI and vice-versa
- Connect to Messaging Systems through Message Queues (JMS, MQSeries, Tibco)
- Create XML result sets for sending to other companies/departments using PowerBuilder 9.0 XML DataWindow or PBDOM functionality
- Consume XML result sets from other companies/departments using PowerBuilder 9.0 XML DataWindow or PBDOM functionality
- Create PDF DataWindow files and/or manipulate text files using the PowerBuilder 9.0 DataWindow SaveAs (PDF) functionality
- Move logic and processing from the Client to the server. The more you move to the server, the faster the Client will run and the lighter and less complex the Web files will be. For example, there was an application from a large Japanese conglomerate that took more than ten seconds to execute a DataWindow Update. The complex validation rules in the Update were consolidated into n-Tier NVOs, and the DataWindow Update now only takes one second.

### 5.2.3 Restrictions in n-Tier NVOs usage

Typically, what is encapsulated into n-Tier NVOs is the application business logic that is not related to the visual aspects of the application. Not all of PowerBuilder's unsupported features can be encapsulated into NVO components that run in Appeon Server.

The following are the restrictions for n-Tier NVOs by EAServer and Appeon. Some restrictions are marked as "Appeon Pure-JavaScript limitation" because they no longer exist with Appeon Xcelerator deployment. You cannot:

- Use the PowerScript MessageBox function. (EAServer limitation)
- Use the GetEnvironment function (Appeon Pure-JavaScript limitation)
- Use application global variables. (EAServer limitation)
- Use BLOB data type. (Web browser and Appeon Pure-JavaScript limitation)
- Pass arrays as parameters. (Appeon limitation)
- Use visual controls or objects. (EAServer limitation)
- Use Any or visual control/object data type as the parameters for NVO functions and/or events. (EAServer limitation)

- Share database transactions across several physical machines (EAServer limitation)
- Use dot notations on the client to refer to the instance variables of an n-Tier NVO. Appeon suggests that you add functions in an n-Tier NVO to get/set its instance variable values. (Appeon limitation)
- Trigger or post the events of an n-Tier NVO. (Appeon limitation)

If an environment-related function is packaged into an n-Tier NVO, the return value of the function may be different from that in PowerBuilder. For example, the GetEnvironment() function in an n-Tier NVO returns the system information for the Client machine when executed in the PowerBuilder Client, but returns the server information when executed in Appeon Server.

For detailed information on using n-Tier NVOs, please refer to *Appeon Help for Pure-JavaScript Deployment* or *Appeon Help for Appeon Xcelerator Deployment*.

### 5.2.4 Steps for moving unsupported features or business logic to Appeon Server

STEP 1 – Create new PowerBuilder NVO objects; add user functions, events and/or properties to them as necessary. Encapsulate unsupported code or business logic into these functions/events.

STEP 2 – Deploy the NVOs holding unsupported features to the Appeon Server that is used for Web deployment of the target PowerBuilder application. To deploy an EAServer component from a PowerBuilder NVO, use the EAServer Component Project wizard in PowerBuilder.

STEP 3 – Generate stubs and skeletons of the server NVO components that you have deployed in the previous step. During the generation process, select the following options, and leave the other options at default:

Generate Stubs

Generate Java Stubs

Generate Java Files

Compile Java Stubs

Refer to Section 5.3.2: *Generating Stub/Skeleton in EAServer* for step-by-step instructions.

STEP 4 - Make sure the Bind Thread property of each server component is enabled in the Properties | Instances tab.

STEP 5 – Build the EAServer Proxy objects in the target application Client. The Proxy objects act as agents for Server NVO Components. To define an EAServer Proxy object, use the EAServer proxy object generator in the PowerBuilder Project painter.

STEP 6 – At the target application Client, modify the Connection object properties and connect to the deployment Appeon Server.

STEP 7 – Instantiate the Server NVO Components and call their methods to replace the original unsupported code in the target PowerBuilder application.

STEP 8 – Run and debug the target application in PowerBuilder to ensure the application Client works correctly with Server NVO Components, then perform a full build and optimize the application target in the PowerBuilder IDE.

For more information on deploying and using EAServer components from PowerBuilder NVOs, please refer to *PowerBuilder Help* and Sybase EAServer documentation.

## 5.3 Migrating distributed applications without distributed DataWindows

### 5.3.1 Requirements

You must generate stubs and skeletons for n-Tier NVOs in distributed applications as required below:

- All n-Tier NVOs used in a distributed application must be located in the package that has the same name with the PowerBuilder application name.
- During the stub and skeleton generation process, select the following options, and leave the other options at default:

  Generate Stubs

  Generate Java Stubs

  Generate Java Files

  Compile Java Stubs

  Refer to Section 5.3.2: *Generating Stub/Skeleton in EAServer* for step-by-step instructions. You can avoid generating stubs and skeletons for the NVOs one by one by generating stub and skeleton for the package that contain the NVOs.

### 5.3.2 Generating Stub/Skeleton in EAServer

#### 5.3.2.a Stubs and skeletons

Stubs and skeleton files allow communication between a Client and a component in EAServer, regardless of the Client and the server component types. For example, a Java or C++ Client can be used to interact with a PowerBuilder NVO component in EAServer.

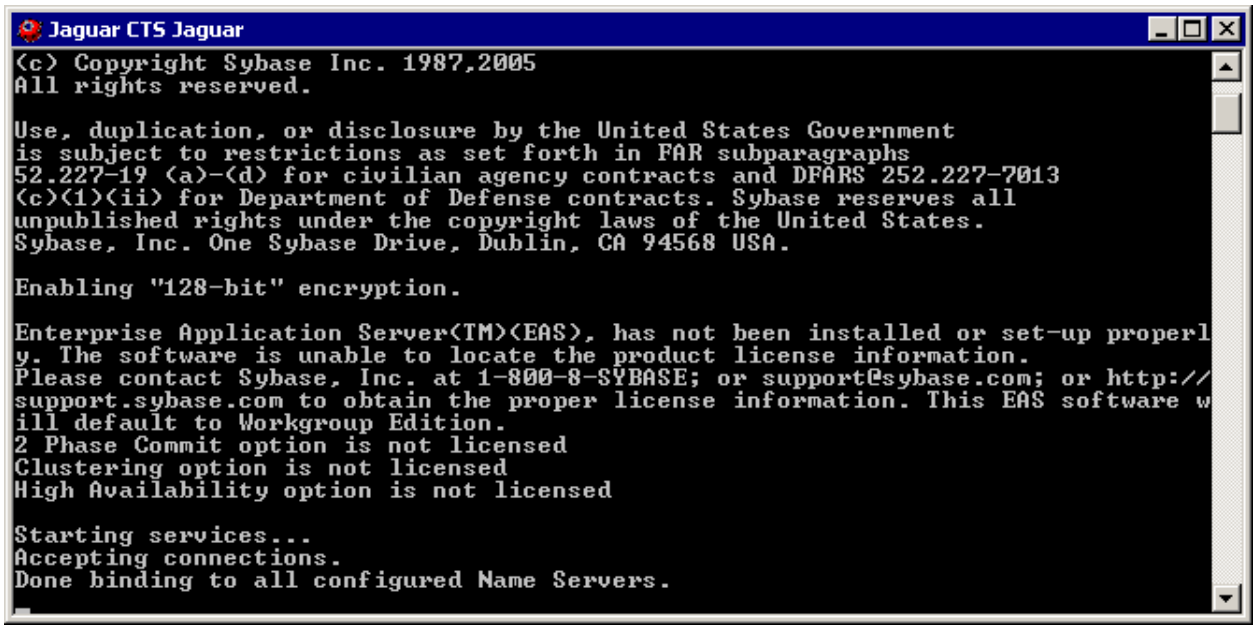
Stubs and skeletons for an Appeon Web application are the counterparts to EAServer Proxy objects and EAServer Component objects in an n-Tier PowerBuilder application. Both the n-Tier PowerBuilder application and the Appeon Web application call NVO components in an EAServer. The Proxy object allows the PowerBuilder Client to access the NVO components on EAServer. Stubs allow the Appeon Web application to establish communication with the EAServer and access its NVO components on Appeon Server.

#### 5.3.2.b Steps to generate Stub/Skeleton in EAServer

Perform the following steps to generate EAServer Stub/Skeleton files for the prospective Web application. Steps 4 and 5 are the most important.

STEP 1 – Start the Appeon Server that will be used for the Web deployment of the target distributed application, as shown in Figure 5-1.

**Figure 5-1: Appeon Server**



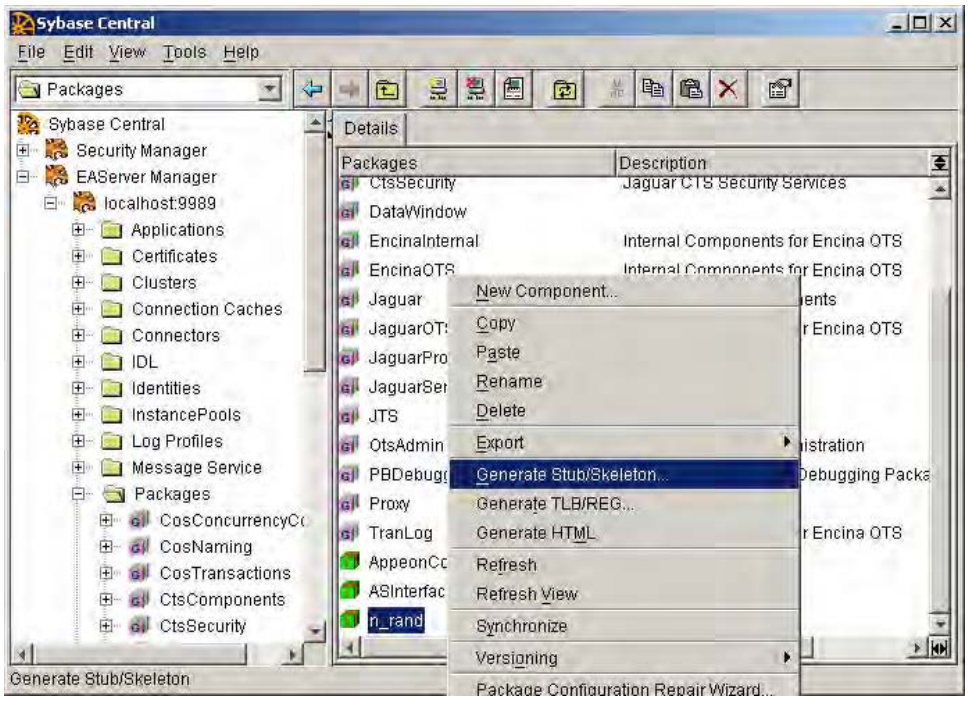

STEP 2 – Use EAServer Manager to login the Appeon Server, as shown in Figure 5-2.

**Figure 5-2: Login to Appeon Server *localhost***


STEP 3 – Select the package under Installed Packages in the tree of files, so components in the selected package are displayed in the right area. Right-click on a component and select *Generate Stub/Skeleton* from the context menu.

Stubs/Skeletons can be generated for each component in a package one by one, or for all the components in a package, by right clicking on the package name and choosing *Generate Stub/Skeleton*, as shown in Figure 5-3.

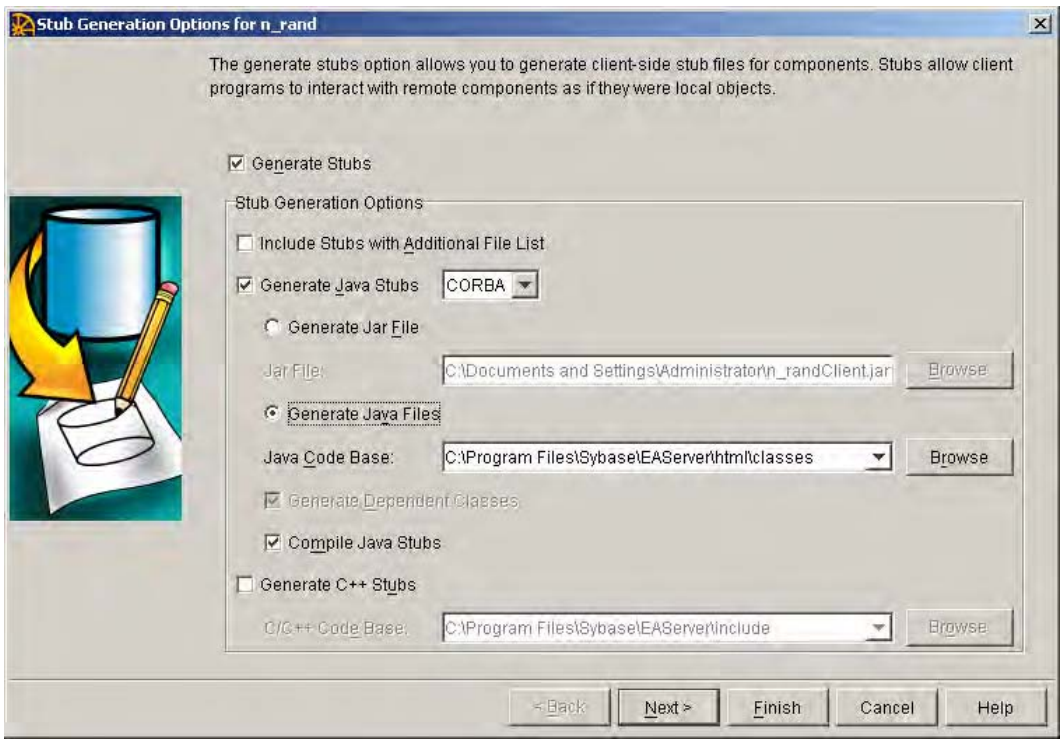
**Figure 5-3: Generate stub/skeleton**




STEP 4 – Select the following options, as shown in Figure 5-4 and leave the other options at default:

- Generate Stubs
- Generate Java Stubs
- Generate Java Files
- Compile Java Stubs

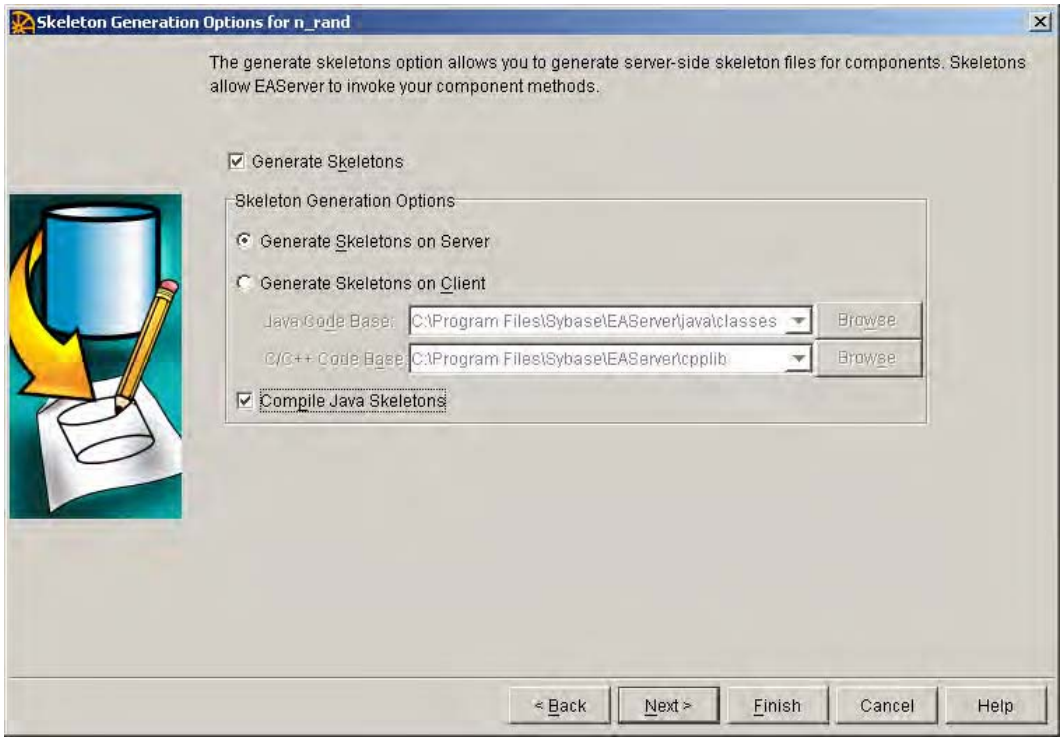
**Figure 5-4: Generate stubs & skeletons**



For details on the options displayed in the Generate Stubs & Skeletons dialog box, please refer to *EAServer Manager Online Help*. Online help can be accessed from "*Help | EAServer Manager Help*" in EAServer Manager.

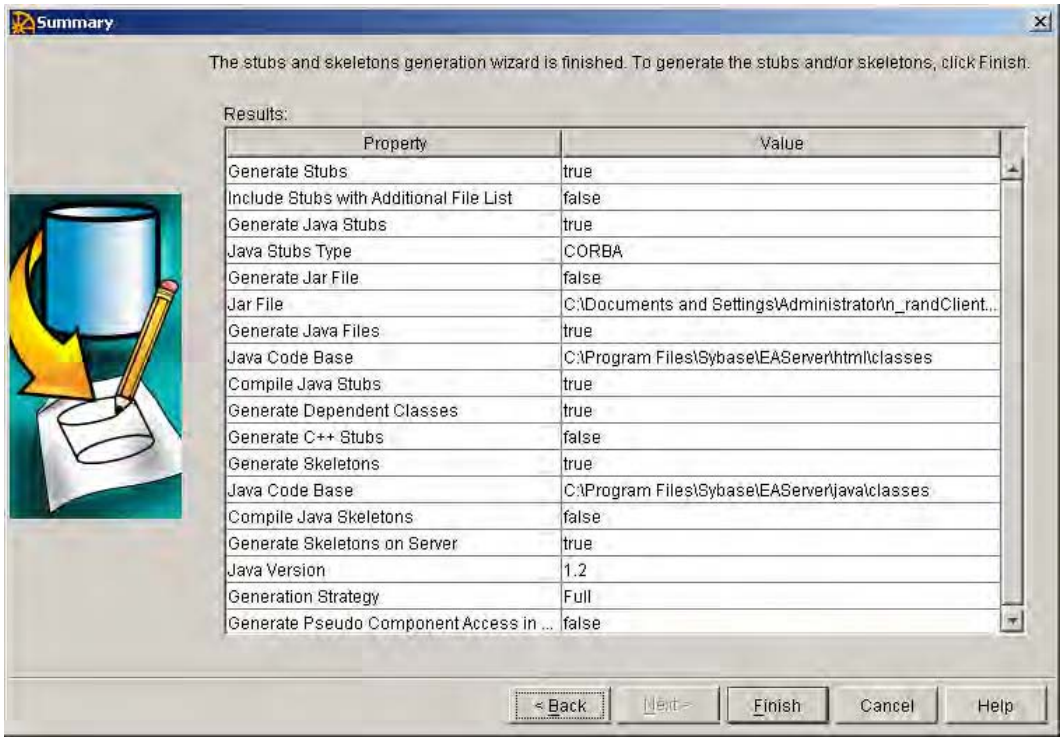
STEP 5 – Select the *Generate Skeletons* and *Compile Java Skeletons* options and leave the others at their default values, as shown in Figure 5-5. Click *Next*.

**Figure 5-5: Skeleton generation options**



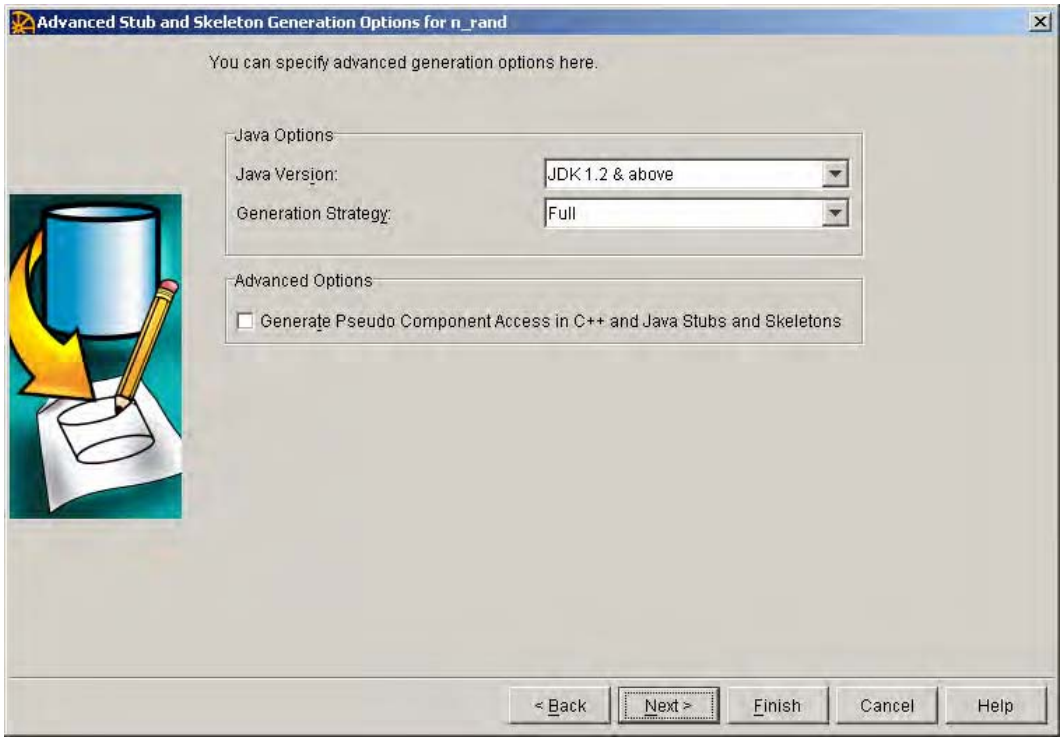

STEP 6 – The wizard now comes to the last screen, as shown in Figure 5-6. Click *Finish*.

**Figure 5-6: Summary**



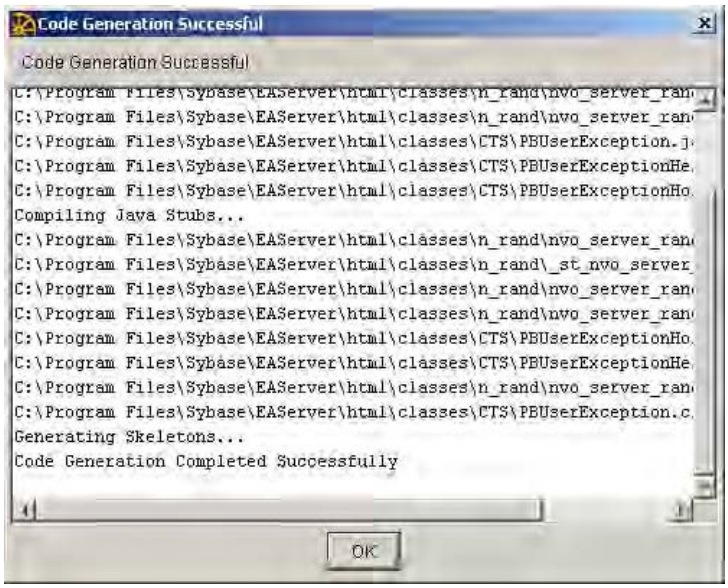
STEP 7 – Leave all options at default and click *Next*, as shown in Figure 5-7.

**Figure 5-7: Advanced stub and skeleton generation options**




STEP 8 – The Stub/Skeleton file generation progress is displayed. When it is complete, click *OK* to close the Code Generation status window, as shown in Figure 5-8.

**Figure 5-8: Code generation successful**




STEP 9 – Repeat the above operations to generate stub and skeleton files for all the PowerBuilder NVO components.

### 5.3.3 Deployment of the application

After the stubs and skeletons are generated as required for its n-Tier NVOs, the migration of a distributed application that does not contain distributed DataWindows is similar to that of a normal Client/Server application. You only need to create an application profile for the client

application of the distributed application, and deploy the application profile using the Appeon Developer deployment wizard.

## 5.4 Migrating distributed applications with distributed DataWindows

### 5.4.1 Benefits in using distributed DataWindows

"Distributed DataWindows" refers to the use of DataWindow/DataStore objects in a distributed environment. In a distributed PowerBuilder application, a DataWindow control at the Client can associate with a DataStore object in EAServer. The Client DataWindow control is responsible for the visual representation of data and deals with user operations, while the DataStore object in EAServer is responsible for transactions. The state of the Client DataWindow control is synchronized with the state of the DataStore object in EAServer and vice versa, using relevant DataWindow functions.

There are two benefits to using distributed DataWindow technology with Appeon:

- Provides more scalability by separating user interface and business logic.
- Works around Appeon-unsupported DataWindow functions by moving the functions in server DataStore objects.

### 5.4.2 Workaround required if you use distributed DataWindows

If you use distributed DataWindows in the application and plan to migrate the DataWindows to the Web, you must use the workaround provided by Appeon for the distributed DataWindows via:

1. Deriving distributed DataWindows and DataStore objects from *appeondatawindow* and *appeondatastore*
2. Deriving PowerBuilder GetFullState, SetFullState, GetChanges and SetChanges functions from corresponding Appeon functions.

#### 5.4.2.a Why the workaround is required

PowerBuilder GetFullState, SetFullState, GetChanges and SetChanges functions use BLOB (Binary Large Object) parameters for passing DataWindow or DataStore object specifications. Appeon Pure-JavaScript deployment does not support BLOB. Appeon Xcelerator deployment supports BLOB, but it cannot directly interpret the BLOB DataWindow or DataStore object specifications.

For Pure-JavaScript deployment, the workaround is required for overriding the BLOB parameters to Appeon-supported String parameters, and interpreting the String DataWindow or DataStore object specifications.

For Appeon Xcelerator deployment, the workaround is required for interpreting the BLOB DataWindow or DataStore object specifications.

#### 5.4.2.b Main workaround steps

This section introduces the main workarounds steps you should take for using distributed DataWindows in your application. For detailed step-by-step instructions and samples, refer to Appeon Workarounds Guide available at http://www.appeon.net/support/documents/workarounds.htm.

STEP 1 – Add the workaround PBLs and DLLs provided by Appeon to your application.

Table 5-1 shows you how to add the files to the application.

**Table 5-1: Appeon workaround PBLs and DLLs**

| Deployment Option | File Name | Location | Copy the File to the location of the PowerBuilder application that contains… |
|---|---|---|---|
| Appeon Xcelerator deployment | *appeon_workarounds_ax.pbl* | \Appeon\Developer\appeon_workarounds\AX | Server DataStores |
| | *EonEmfPic.dll* and *EonAXNVO.dll* | \Appeon\Developer\appeon_workarounds\AX | DataWindows on the client |
| Pure-JavaScript deployment | *appeon_workarounds_js.pbl* | \Appeon\Developer\appeon_workarounds\JS | Server DataStores |
| | *EonEmfPic.dll* and *EonJSNVO.dll* | \Appeon\Developer\appeon_workarounds\JS | DataWindows on the client |

STEP 2 – Derive all distributed DataWindows and DataStores from AppeonDataWindow and AppeonDataStore.

AppeonDataStore and AppeonDataWindow are built in *appeon_workarounds_js.pbl* or *appeon_workarounds_ax.pbl*.

STEP 3 – (For Pure-JavaScript deployment only) Change the parameter type from BLOB to String.

For Appeon Xcelerator deployment, you do not need to take this step because the deployment method supports BLOB.

STEP 4 – Migrate n-Tier DataWindows. In PowerBuilder, deploy the server DataWindows and DataStores that are inherited from AppeonDataStore to Appeon Server.

You also need to deploy AppeonDataWindow objects and AppeonDataStore objects to the Appeon Server.

Note: After the deployment, you need to generate stubs and skeletons for the server DataWindows and DataStores as well as n-Tier NVOs in the application by following the instructions in Section 5.3: *Migrating distributed applications without distributed DataWindows*.

### 5.4.2.c Workaround limitations

When using the *appeondatawindow* and *appeondatastore* objects to work around the distributed DataWindow technique, there are some limitations regarding the use of Appeon GetFullState, SetFullState, GetChanges, and SetChanges functions.

**Table 5-2: Limitations of Appeon workaround**

| Limitation/Difference in… | Limitation/Difference Description |
|---|---|
| DataWindow styles | The workaround only works with DataWindows or DataStores of Grid, Group, Freeform, and Tabular style. |
| Return value of the functions | The return value of Appeon SetFullState may have different meaning from that of PowerBuilder system SetFullState function. |
| | The Appeon GetChanges function always returns -1 if it fails. In PowerBuilder, the function can return more error numbers (-1, -2 and -3). |
| | The Appeon SetChanges function can return -1 and -3, but cannot return 2 and -2. |
| Manipulation of DataWindow properties | (For Appeon Pure-JavaScript deployment only) You can use the Appeon workaround distributed DataWindow technique to manipulate DataWindow data and data status, but cannot manipulate DataWindow properties, such as modifying the sort or filter criteria. |
| Dynamically creating of DataWindows | (For Appeon Pure-JavaScript deployment only) Dynamically creating DataWindows is unsupported. |
| DataWindow ImportString function | If using the DW ImportString function in a distributed DataWindow environment, keep the date display format the same at the Client and Appeon Server machines. In addition, the date/time format configuration in AEM should be kept the same as the system date/time configuration on Appeon Server. |
| State information initialization of a DataWindow/DataStore | In PowerBuilder, the state information of a DataWindow/DataStore is initialized whenever you set its DataObject property. However, if using *appeondatawindow* and *appeondatastore*, the state information is initialized only when you change the DataObject property to a different DataWindow object. |
| Truncation of characters in certain cases | When applying Appeon SetChanges to a target DataWindow/DataStore, if a column of Char type in the source DataWindow/DataStore has defined more characters than its corresponding column in the target DataWindow/DataStore, characters from the source column that exceed the length limit of the target column are truncated, but in PowerBuilder the extra characters are preserved. |
| Un-modified or modified data | When calling PowerBuilder GetFullState and GetChanges, changed (but not accepted) data in a DataWindow control is treated as un-modified data, but if using the Appeon *appeondatawindow* and *appeondatastore*, changed (but not accepted) data is treated as modified data. |

### 5.4.3 (For Pure-JavaScript deployment only) Special steps for deploying an application that contains distributed DataWindows

For Appeon Xcelerator deployment method, the steps for deploying an application that contains distributed DataWindows are the same as the steps for deploying common distributed applications.

For Pure-JavaScript deployment, the steps for deploying an application that contains distributed DataWindows are different from the steps for deploying common distributed applications. The difference can be summarized in the following four steps.

STEP 1 – Put all server DataStores into one PBL.

Assuming that the application to be deployed is *application_distribute*, you need to build a new server application (for example, *application_datastore*) in PowerBuilder to keep server DataStores.

STEP 2 – Create a component profile for *application_datastore* in Appeon by using *Add component* function provided in the Appeon Developer Configuration window.

STEP 3 – Deploy *application_datastore* to Appeon Server with the Appeon Developer deployment wizard.

STEP 4 – Deploy *application_client* to Appeon Server and Appeon Server Web Server.

# 6 Migrating PFC applications

## 6.1 Overview

Appeon 3.1 supports PowerBuilder applications that have been built with the PowerBuilder Foundation Class Library or PFC framework.

The PFC framework contains windows, user objects, DataWindows, and structures that you can use to help build PowerBuilder applications. By using PFC, you can quickly add powerful capabilities to your applications.

Appeon provides the Appeon-compliant Framework (ACF) as the solution for migrating existing PowerBuilder PFC applications. This is essentially the PFC 8 framework with all unsupported features removed or reworked. ACF supports approximately 85% of all PFC 8 features. Because of the low percentage of unsupported features, PFC applications usually require only minimal work to deploy to the Web.

Note: Appeon-compliant Framework is designed on the supported status of Pure-JavaScript deployment. Because Appeon Xcelerator supports more features than Pure-JavaScript deployment, if you plan to deploy a PFC application with Appeon Xcelerator deployment method, you can either deploy the application without applying ACF and comment out unsupported features, or deploy the application with ACF and uncomment those features that are supported in Appeon Xcelerator but reserved in ACF.

For more information on ACF objects, refer to *Appeon-Compliant Framework Reference* in Appeon Help (accessed from the Appeon Help button in the Appeon Developer toolbar).

## 6.2 Appeon-compliant Framework

The ACF PowerBuilder Library (PBL) files are located in the *acf_frameworkForAX* and *acf_frameworkForJS* directories in the Appeon Developer installation path. (For example, *C:\Program Files\Appeon\Developer\Appeondemo\acf_frameworkForAX*)

### 6.2.1 What is ACF

Appeon-compliant Framework (ACF) is essentially the PFC 8 framework with all unsupported features removed or reworked. ACF supports approximately 85% of all PFC 8 features.

### 6.2.2 Why ACF is introduced

ACF is the Appeon solution to migrating existing PowerBuilder PFC applications onto the Web. Earlier versions of Appeon for PowerBuilder could not migrate PFC applications because the PFC framework contained too many unsupported features. To solve this problem, Appeon has modified the PFC framework – removing or working around its unsupported features – to get the ACF.

ACF is compatible with the PFC framework. This means the PFC framework can be replaced with ACF in your PFC applications and the PFC application can be migrated into any standard PowerBuilder application.

ACF is also excellent for building new PowerBuilder applications even if you do not plan to deploy them to the Web. Building new PowerBuilder projects with ACF gives you the option of deploying the application to the Web at the click of a button.

### 6.2.3 ACF architecture

By preserving the architecture and interface of PFC, the PFC files in a PFC application can essentially be swapped with ACF files and the application will work as before, except for unsupported features.

#### 6.2.3.a Inheritance hierarchy

The inheritance hierarchy of ACF is the same as PFC.

***Ancestor Level & Extension Level***

Like PFC, ACF implements an ancestor level and an extension level. Objects in the ancestor-level libraries contain all instance variables, events, and functions; objects in the extension-level libraries are unmodified descendants of corresponding objects in the ancestor library.

***Object inheritance***

Objects in ACF are inherited in the same way as in PFC. For example, *pfc_n_base* is the ancestor object for all ACF custom class user objects, and *pfc_u_base* is the base object for all ACF custom visual user objects.

#### 6.2.3.b Objects

Most of the PFC objects have been extracted into ACF. ACF uses the same object naming conventions as PFC. For example, *pfc_w_master* is the master Window and is in the ACF ancestor level; *n_cst_dwsrv* is the custom class user object for DataWindow services and is in the ACF extension level.

#### 6.2.3.c Object Functions and Events

For any ACF object, its function/event names and function/event parameters are kept the same as the corresponding PFC object. This is to ensure optimal compatibility when the PFC framework is replaced or swapped with ACF. However, there will be dummy parameters in some functions/events since Appeon has simplified their script.

#### 6.2.3.d Object Instance Variables

Most of the instance variables of supported PFC objects are also supported in ACF.

### 6.2.4 ACF objects classification

As the revised version of PFC, Appeon-compliant Framework (ACF) is made up of a set of objects that are extracted from PFC. Extracted objects (ACF objects) can be classified into the following three categories.

- **ACF Supported Objects** - These are extracted from the corresponding PFC objects, with some or little modification to the PFC objects' original source code. These objects are both functional in PowerBuilder desktop applications and after the PowerBuilder application is deployed to the Web.
- **ACF Preserved Objects** - These are extracted from the corresponding PFC objects, with the original source code of the PFC objects being entirely commented. Preserved objects have only empty functions and events that are not functional in Client Server or after being deployed to the Web.

  Preserved objects are “shell” objects that help to keep ACF compliant with PFC. In addition, while un-extracted objects may never be supported, the preserved objects are

likely to be supported in subsequent versions of ACF. They are preserved for an ACF future release: the commented script for a preserved object will be restored as soon as the object can be supported.

- **ACF Debugging Objects** - These are extracted from the corresponding PFC objects (mostly in pfcutil.pbl), with no modification of the PFC object's original source code. Debugging objects are only used to debug the PowerBuilder desktop application and not the deployed Web application. These debugging objects are essentially ignored when the application is deployed to the Web.

  In the ACF library, the application running environment is checked before any debugging object is called; if an application is running on the desktop, the relevant debugging objects will be called; otherwise, they will not be called. Debugging objects are functional in Client Server but not functional on the Web.

  For detailed information on ACF Supported Objects, Preserved Objects, and Debugging Objects, please refer to *Appeon-compliant Framework Reference* in *Appeon Help*.

### 6.2.5 Un-extracted PFC objects

Un-extracted objects are the PFC objects that have not been extracted into ACF. If your PFC applications use any of these un-extracted objects, remove them from your application before deploying to the Web. These objects are not extracted and do not exist in ACF because they relate to functionality beyond the capabilities of the Web.

For detailed information on the un-extracted objects, please refer to *Appeon-compliant Framework Reference* in *Appeon Help*.

### 6.2.6 ACF files location

ACF contains 12 PowerBuilder Library files (PBLs). They are located in the *acf_frameworkForAX* and *acf_frameworkForJS* directories in the Appeon Developer installation path.

(For example, *C:\Program Files\Appeon\Developer\Appeondemo\acf_frameworkForAX*)

**Table 6-1: ACF files**

| Contents | Ancestor level | Extension level |
|---|---|---|
| Appeon-specific objects for new features or workarounds | PFC_APPEON.PBL | PFE_APPEON.PBL |
| Application and global services | PFCAPSRV.PBL | PFEAPSRV.PBL |
| DataWindow services | PFCDWSRV.PBL | PFEDWSRV.PBL |
| Visual and standard class user objects | PFCMAIN.PBL | PFEMAIN.PBL |
| Utility services | PFCUTIL.PBL | PFEUTIL.PBL |
| Window services | PFCWNSRV.PBL | PFEWNSRV.PBL |

If its PFC extension level does not contain any user extended code, use the ACF extension level to replace the PFC extension level. Otherwise, you should manually delete the un-extracted objects in the PFC extension level.

### 6.2.7 Steps to migrate a PFC application

Perform the following steps to migrate a PFC application:

STEP 1 – Swap the PFC ancestor Level PBLs with the ACF ancestor level PBLs in the target PFC application to be migrated.

STEP 2 – If the PFC extension level of the target PFC application does not contain any user extended code, use the ACF extension level to replace the PFC extension level; or, manually delete the un-extracted objects in the PFC extension level.

STEP 3 – If the Corporate Class Library contains additional layers for business rules, delete within these layers any of the objects that have an inheritance relationship with the un-extracted objects.

STEP 4 – Perform a Full Build to the target PFC application.

Since in Step 1 and Step 2, PFC ancestor level and ACF ancestor level have been swapped, some objects in the PFC extension level may have been deleted. There may be a number of error messages in the Output window when performing the Full build.

STEP 5 – Correct the errors displayed in the PowerBuilder Output window.

Correct the errors until no errors display when performing a Full build.

STEP 6 – Run the target PFC application in PowerBuilder, and test the application:

Functionality related to the ACF Preserved Objects will be completely lost. Pay special attention to such functionality and, if necessary, modify the source code related to these unsupported features.

The ACF supported objects will have some limitations, so any functionality related to these objects in the target PFC application will also have some limitations. Pay attention to such limitations and, if necessary, modify the source code related to these limitations.

The functionality related to the ACF Debugging objects will only work in a desktop environment. They will be ignored in the Web migration.

STEP 7 – Modify the target PFC application to remove any bugs or issues you find. Perform a Full Build and optimize application PBLs in the PowerBuilder IDE.

STEP 8 – Configure the application as required in Section 3.8: *Pre-configuring for the Web applications*, and deploy the application with the Appeon Developer deployment wizard.

# 7 Database Connection Setup

## 7.1 Overview

The steps for configuring the database for an Appeon-deployed application are the same as the steps for configuring the database for a PowerBuilder application. However, the way the database server is accessed is different: a PowerBuilder application directly accesses the database server via transaction object(s), while an Appeon-deployed application accesses the database server via Appeon Server connection caches.

This chapter describes how to enable a deployed application to access its database. Two key tasks are involved:

- Setting up communication between the database server and Appeon Server. This refers to setting up Appeon Server connection caches.
- Setting up communication between the deployed application and Appeon Server. This refers to setting up the mapping between the application transaction objects and Appeon Server connection caches.

There are some advanced configurations that are also related with database connection setup (for example, database auditing). This chapter outlines common techniques for handling such configurations in the Appeon environment.

## 7.2 Setting up Appeon Server connection caches

The connection cache for a Web application is the counterpart to the transaction object in the target PowerBuilder application. The transaction properties in the target PowerBuilder application contain database connection parameters which should be correspondingly configured in connection caches.

Appeon Web applications rely on Appeon Server JDBC connection caches to interact with the database servers. When creating a JDBC connection cache, you can use different JDBC drivers. However, Appeon has some recommendations on which JDBC driver is to be used for certain types of databases.

Setting up an Appeon Server connection cache is the same as setting up an EAServer connection cache, but Appeon has provided some useful instructions in this section. If there are problems creating JDBC connection caches, you should refer to the documentation from the database/JDBC driver vendor and Sybase EAServer.

### 7.2.1 Why a JDBC connection cache?

In Appeon Web applications, data-related operations are managed by Appeon Server. The data related operations are built with J2EE technology and they require the JDBC interface. Regardless of the interface (ODBC, JDBC, or native driver) the target PowerBuilder application uses for its database connection, the Web application will always need to use JDBC.

One issue with the JDBC interface is that most PowerBuilder applications use a native database driver, and there may be differences between the behavior of the native/ODBC database interface and the JDBC interface. Before you configure a JDBC connection cache, you should test your PowerBuilder applications with the JDBC driver to make sure it does not cause any issues. For Sybase ASA and ASE installations, the iAnywhere™ JDBC driver

from Sybase resolves many of the differences between the ODBC and JDBC interfaces. iAnywhere JDBC driver is the recommended and certified driver for Sybase ASA and ASE.

### 7.2.2 JDBC driver type

The JDBC connection caches can use any of the four types of JDBC drivers:

Type 1: JDBC-ODBC Bridge

Type 2: Native-API/partly Java driver

Type 3: Net-protocol/all-Java driver

Type 4: Native-protocol/all-Java driver

Each type has advantages and disadvantages.You should run tests to decide which type of JDBC driver works the best for the specific application and database. Generally, Type 3 and Type 4 drivers show better performance than Type 2 drivers, so it is recommended that you evaluate Type 3 or Type 4 for both intranet and Internet Web deployments. Because of performance considerations, Type 2 drivers should only be used in an intranet environment where response times are generally faster.

Because of some known problems (see the *Appeon Release Bulletin*), avoid using Sybase jConnect™ for JDBC™ driver with Sybase Adaptive Server® Enterprise database. Instead, choose a connection cache that uses the iAnywhere JDBC driver.

The iAnywhere JDBC driver has a thin Java layer that communicates directly with ODBC. It is four times faster than the Sun JDBC-ODBC Bridge and twice as fast as jConnect for cursor support; it also works with other ODBC drivers and should be able to connect with other databases as well. Especially for Sybase ASA database installations, the iAnywhere JDBC driver is a viable long-term solution since Sybase has rewritten the driver (in ASA 8.0.2 and above only) to offer production quality performance and stability.

### 7.2.3 JDBC driver preparation

#### 7.2.3.a Preparing PowerBuilder component support files

Whichever database and JDBC driver you use for the deployed application, you should add the following PowerBuilder component support files to the *%JAGUAR%\java\lib* directory in the Appeon Server computer. *%JAGUAR%* indicates the installation path of the EAServer that hosts Appeon Server:

- Pbjdbc12.jar for PowerBuilder 8
- Pbjdbc1290.jar for PowerBuilder 9 and 10

You should be able to get the file from the *%Sybase%\Shared\PowerBuilder* folder. Make sure the version of the file is the same as the version of PowerBuilder that Appeon supports.

#### 7.2.3.b Checklist for JDBC driver preparation

Before you configure a JDBC connection cache for your application database, the JDBC driver file(s) must be copied to the *%JAGUAR%\java\lib* directory in the Appeon Server computer. *%JAGUAR%* indicates the installation path of the EAServer that hosts Appeon Server. Table 7-1 is the checklist of the JDBC driver file(s) that should be copied to the *%JAGUAR%\java\lib* directory.

**Table 7-1: Checklist for JDBC driver preparation**

| Database | Driver Type | Driver Files | Availability of the Driver Files |
|---|---|---|---|
| ASA 7, 8 or 9 | iAnywhere JDBC driver | dbjodbc8.dll (or dbjodbc9.dll)<br>jodbc.jar | Available in Sybase ASA 8.0.2 Build 4361 or above. For earlier versions, you can obtain the files from http://sybase.com/downloads. |
| | Sun JDBC-ODBC driver | - | Bundled with the Java 2 SDK, Standard Edition, so there is no need to download it separately. |
| | jConnect JDBC driver | jconn2.jar | Available at *%Sybase%\Shared\jConnect-5_5\classes*.<br>Note: Install *sql_asa.sql* provided at *%Sybase%\Shared\jConnect-5_5\sp* for jConnect to function properly. |
| ASE 12.0 or 12.5 | iAnywhere JDBC driver | dbjodbc8.dll (or dbjodbc9.dll)<br>jodbc.jar | Available in Sybase ASA 8.0.2 Build 4361 or above. For earlier versions, you can obtain the files from http://sybase.com/downloads. |
| | Sun JDBC-ODBC driver | - | Bundled with the Java 2 SDK, Standard Edition, so there is no need to download it separately. |
| | jConnect JDBC driver | jconn2.jar | Available at *%Sybase%\Shared\jConnect-5_5\classes*.<br>Note: Install *sql_server12.sql* for ASE 12.0 or *sql_server12.5.sql* for ASE 12.5 for jConnect to function properly. |
| Oracle 8i | Oracle JDBC driver | classes12.zip<br>Oracle8i 8.1.7.1 Patch<br>nls_charset12.zip | Available at the Oracle Web site (http://otn.oracle.com/software/tech/java/sqlj_jdbc/index.html). |
| Oracle 9i | Oracle JDBC driver | For use with JDK 1.3:<br>classes12.zip<br>nls_charset12.zip<br>For use with JDK 1.4:<br>Ojdbc14.jar | Available at the Oracle Web site (http://otn.oracle.com/software/tech/java/sqlj_jdbc/index.html).<br>Note: *Classes12.zip* and *ojdbc14.jar* cannot be placed in the same location and used at the same time. |
| Oracle 10g | Oracle JDBC driver | For use with JDK 1.3:<br>classes12.zip<br>For use with JDK 1.4:<br>Ojdbc14.jar<br>orai18n.jar | Available at the Oracle Web site (http://otn.oracle.com/software/tech/java/sqlj_jdbc/index.html).<br>Note: *Classes12.zip* and *ojdbc14.jar* cannot be placed in the same location and used at the same time. |
| SQL Server 2000 | Microsoft SQL Server JDBC driver | msbase.jar<br>mssqlserver.jar<br>msutil.jar | Available at the Microsoft Web site (http://www.microsoft.com/sql/downloads).<br>Note: The files have different versions. Make sure the file sizes are equal or close to the following.<br>msbase.jar: 281KB<br>mssqlserver.jar: 66KB<br>msutil.jar: 58KB |

| | | | |
|---|---|---|---|
| DB2 7.2 | IBM JDBC driver | db2java.zip | Available in the *java* folder of the DB2 Server installation directory |
| DB2 8.1 | IBM JDBC driver | db2java.zip<br>db2jcc.jar | Available in the *java* folder of the DB2 Server installation directory |

### 7.2.4 Setting up connection cache for ASA or ASE

Appeon recommends using the Sybase iAnywhere JDBC driver for connecting ASA or ASE databases. The iAnywhere driver is an ODBC-JDBC bridge that can use various types of databases.

The iAnywhere JDBC driver offers production-quality performance and stability, and displays superior performance over the Sun JDBC-ODBC bridge driver and the Sybase jConnect for JDBC. Using the iAnywhere JDBC driver is also recommended as certain problems have been found in using the Sun JDBC-ODBC bridge driver which are resolved in the Sybase version.

In performance testing, Sybase jConnect was four to six times faster than Sun's JDBC-ODBC bridge, while the Sybase iAnywhere JDBC driver is proved to be as fast as jConnect, and in some cases twice as fast. Another advantage is that this bridge driver does not have to adhere to any of the TDS limitations. In future versions of ASA, all of the Sybase ASA tools will use the Sybase iAnywhere JDBC driver as the default communication link, and provide jConnect support as a secondary link.

#### 7.2.4.a Connection cache with iAnywhere JDBC driver

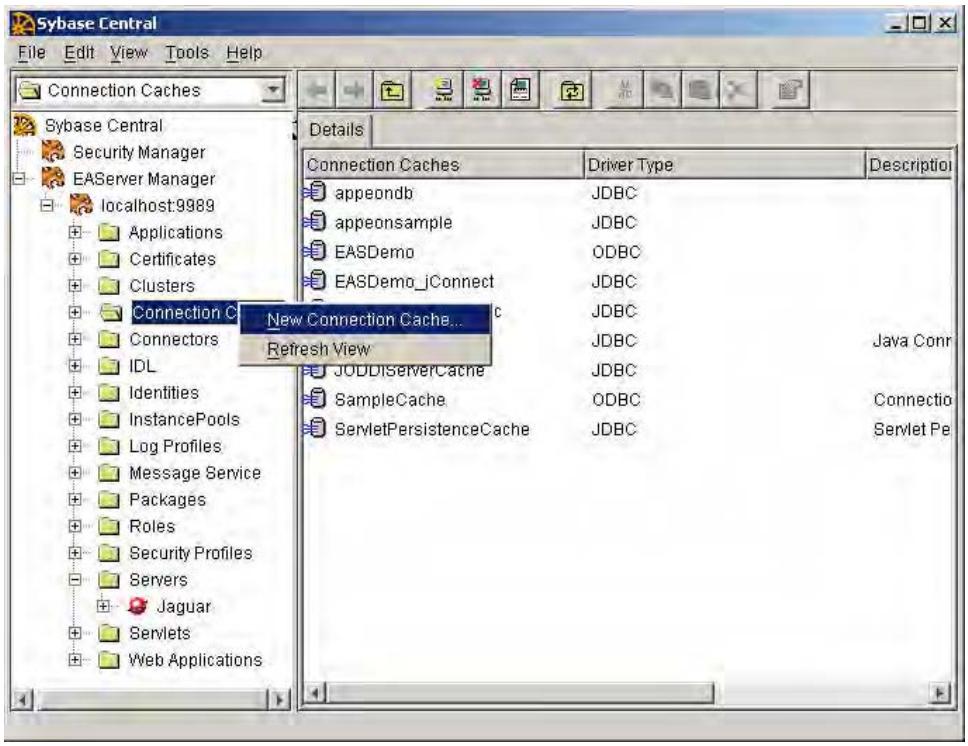
To set up an EAServer connection cache that connects to ASA/ASE with iAnywhere:

STEP 1 – Start EAServer and log in with Jaguar Manager.

For instructions on how to access EAServer using Jaguar Manager, please refer to the *EAServer System Administration Guide*.

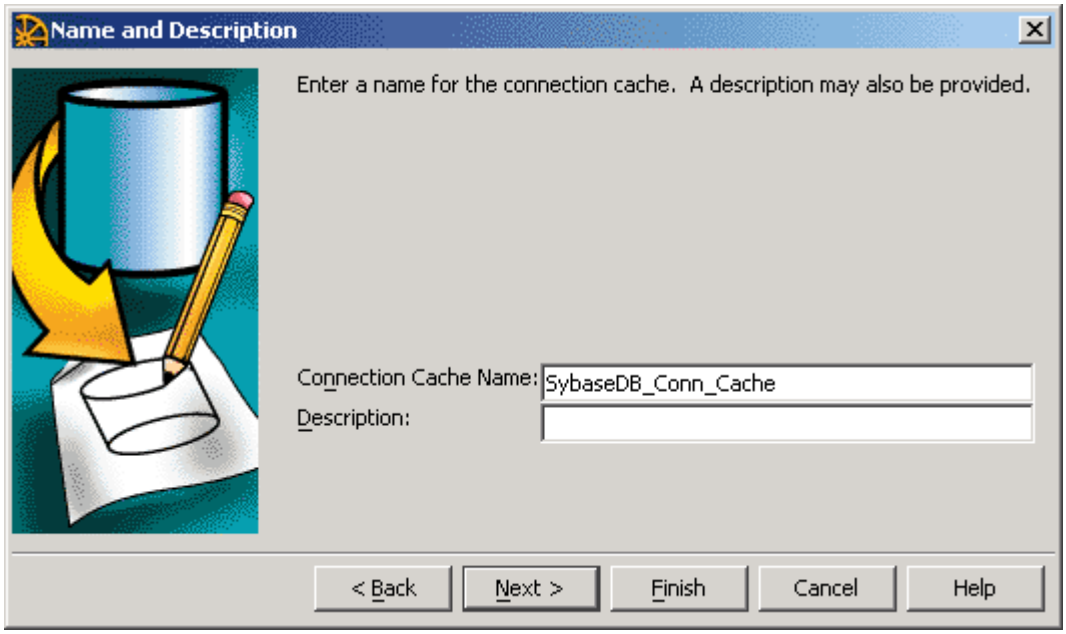
STEP 2 – Right click on *Connection Caches* in the tree view. Choose *New Connection Cache* from the context menu, as shown in Figure 7-1.

**Figure 7-1: New Connection Cache**




STEP 3 – Type the name of the connection cache, and click *Finish*, as shown in Figure 7-2. In this example, the connection cache's name is *SybaseDB_Conn_Cache*.

**Figure 7-2: Create Connection Cache**

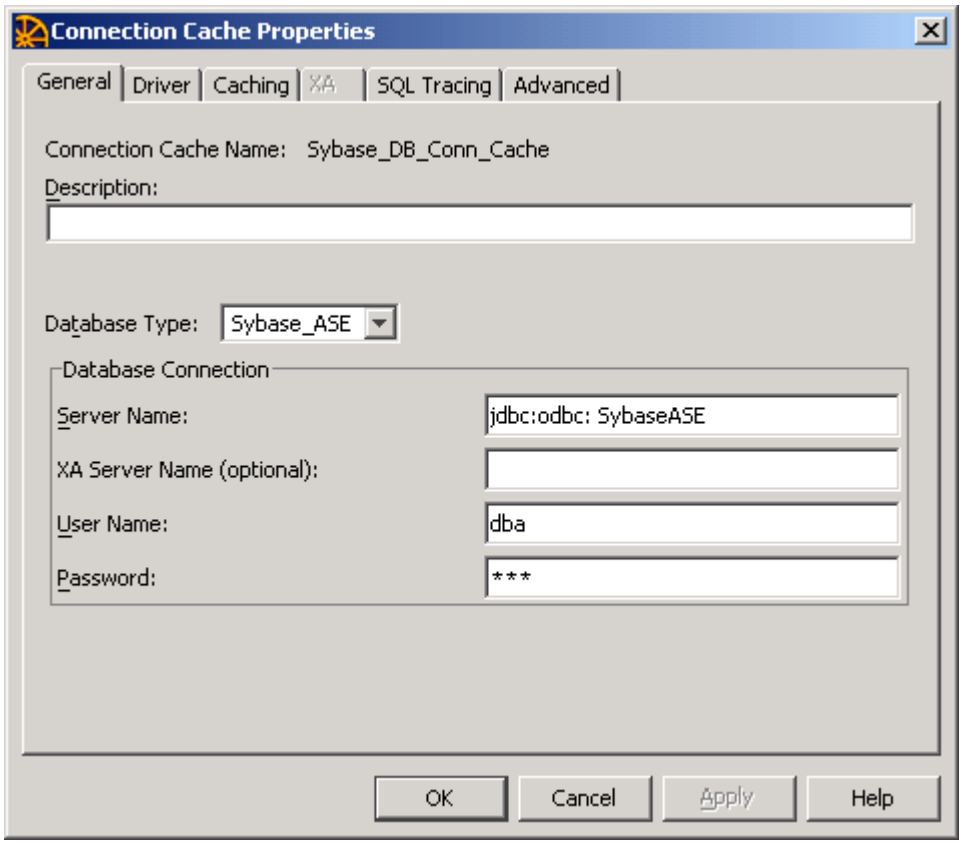



STEP 4 – On the *Connection Cache Properties* dialog box that appears, enter the settings under the General tab, as shown in Table 7-2. The parameters you type are case sensitive.

**Table 7-2: Connection Cache Properties**

| | |
|---|---|
| Database Type | Select “Unknown” for the specific database type. |
| Server Name | Syntax: jdbc:odbc:*DSNname*<br>DSNname refers to the name of the ODBC DSN that is created for the database.<br>e.g. jdbc:odbc:SybaseASE |
| User Name | Type the database login username. The username is set on the Database Server. |
| Password | Type the database login password. The password is set on the database server. |

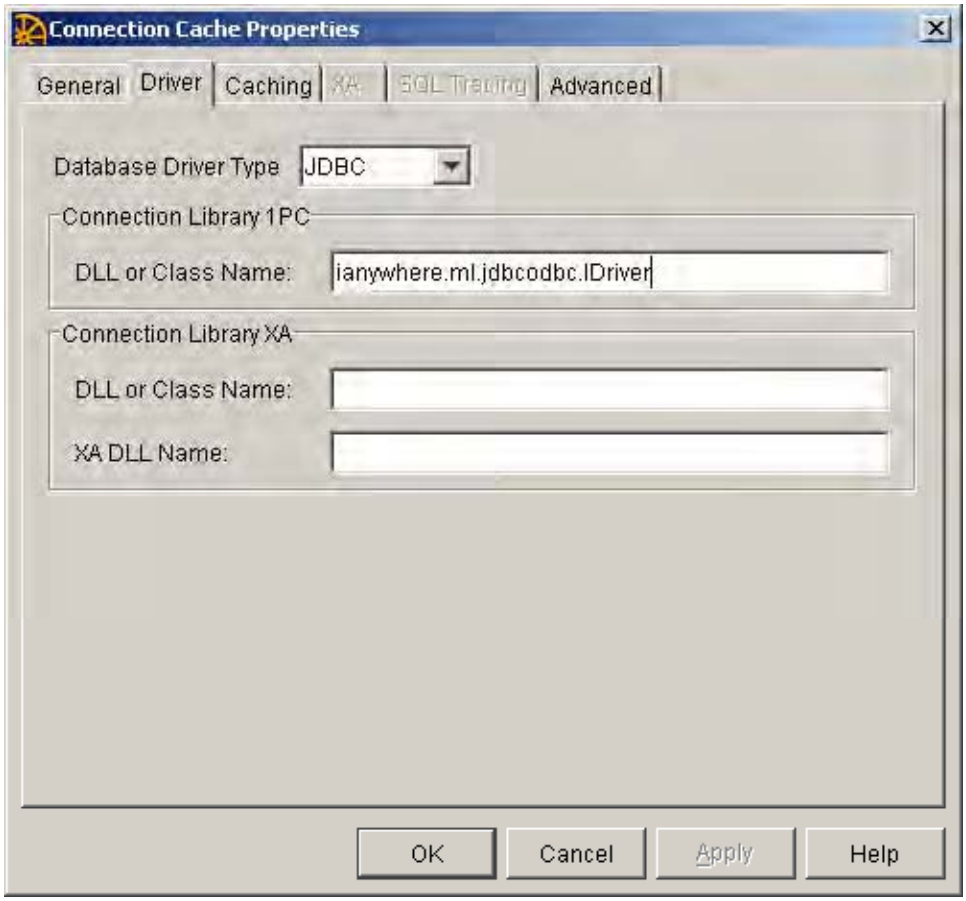
**Figure 7-3: Connection Cache Properties**



STEP 5 – Click the *Driver* tab. Check the JDBC radio button and type the driver string into the DLL or Class Name text box.

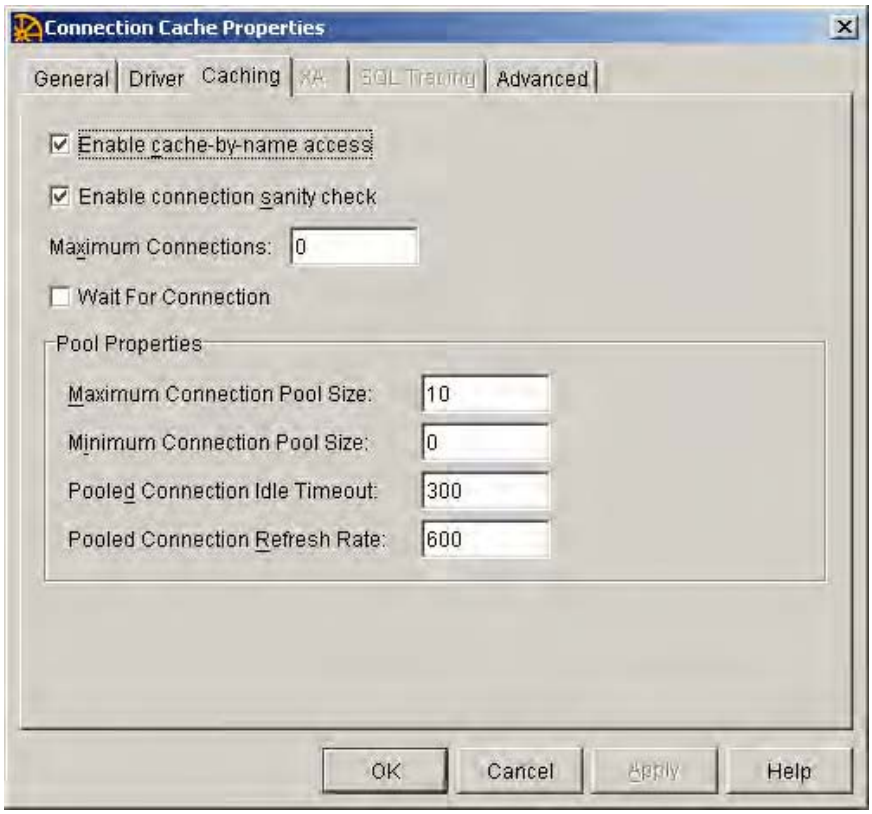
The DLL or Class Name should be “ianywhere.ml.jdbcodbc.IDriver” (It is case-sensitive).

**Figure 7-4: Connection Cache Properties**



STEP 6 – Click the *Caching* tab. Make sure to check both the *Enable cache-by-name access* option and the *Enable connection sanity check* option, as shown in Figure 7-5.

**Figure 7-5: Connection Cache Properties**




STEP 7 – Return to the General tab. Click the Refresh button, and then click Ping. Make sure the ping succeeds before you continue, as shown in Figure 7-6.

**Figure 7-6: Ping Results**


STEP 8 – Close the *Connection Cache Properties* dialog box.

The connection cache has been added.

Note: When using the iAnywhere driver, Merant, or other ODBC drivers to connect to Enterprise databases like Sybase ASE or Oracle, the following message or similar message may pop up on Appeon Server each time a Web DataWindow retrieve is executed. The Web browser will remain blank until you have clicked the *OK* button in the message on Appeon Server, as shown in Figure 7-7. This is because the database driver has not been licensed for use with your database. To resolve this issue and continue using the iAnywhere driver, contact Sybase to get the appropriate license. For all other database drivers, please contact the driver vendor.

**Figure 7-7: License Warning**


### 7.2.4.b Connection cache with jConnect JDBC driver

Steps 1-3 for setting up a connection with jConnect JDBC driver are the same as those in Section 7.2.4.a: *Connection cache with iAnywhere JDBC driver*.

STEP 1 – Start EAServer and log in with Jaguar Manager.

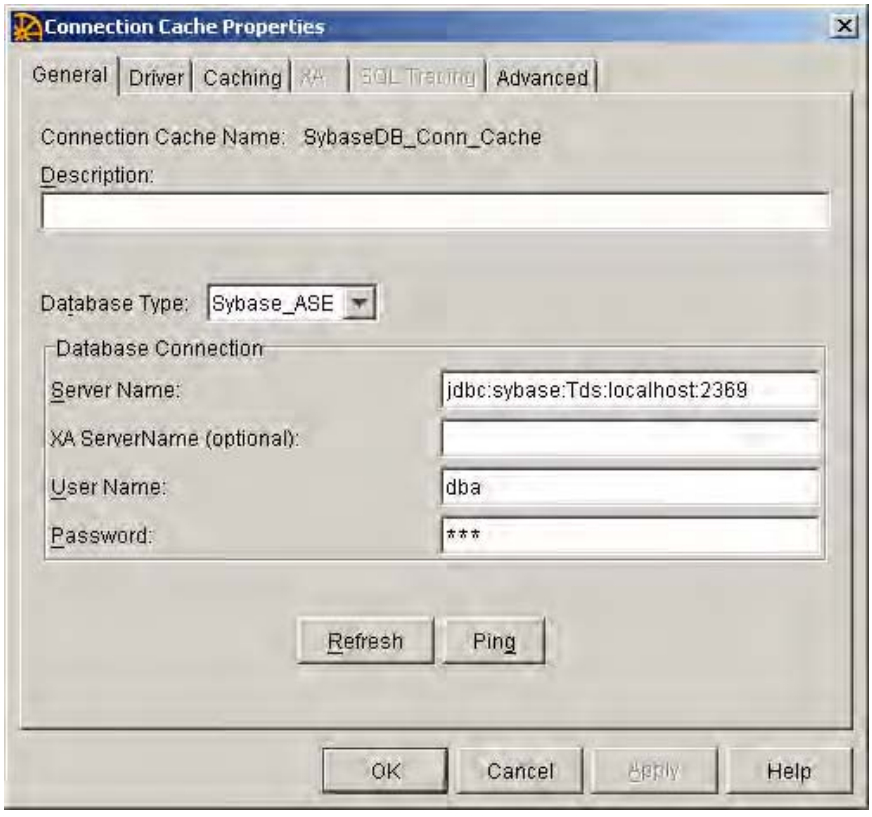
STEP 2 – Right mouse click on *Connection Caches* in the tree view.

STEP 3 – Type the name of the connection cache, and click *Finish.*

STEP 4 – On the *Connection Cache Properties* dialog box that appears, enter the settings under the General tab. The parameters you type are case sensitive.

Syntax for Server Name: jdbc:sybase:Tds:*hostname*:*port/dbname.*

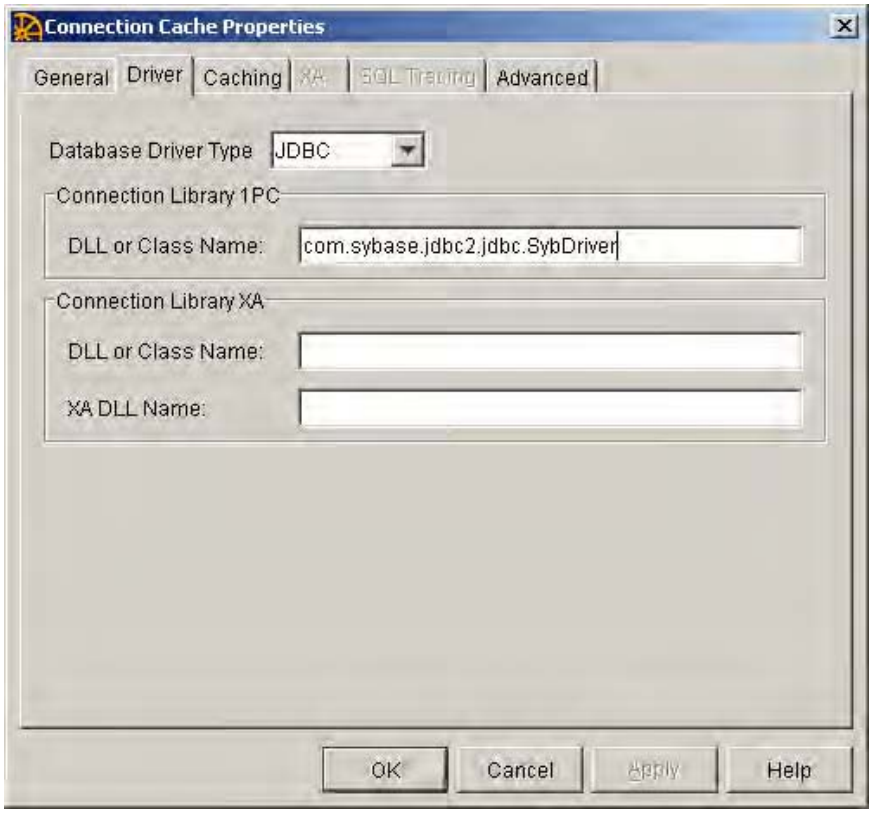
**Figure 7-8: Connection Cache Properties**




STEP 5 – Click the *Driver* tab. Check the JDBC radio button and type the driver string into the DLL or Class Name text box.

The driver string should be *com.sybase.jdbc2.jdbc.SybDriver* as shown in Figure 7-9.

**Figure 7-9: Connection Cache Properties**




STEP 6 – Click the *Caching* tab. Make sure to check both the *Enable cache-by-name access* and *Enable connection sanity check* options.

STEP 7 – Return to the General tab. Click the Refresh button, and then click Ping. Make sure the ping succeeds before you continue.

STEP 8 – Close the *Connection Cache Properties* dialog box.

The connection cache has been added.

### 7.2.4.c Connection cache with Sun JDBC driver

Steps 1-3 for setting up a connection with Sun JDBC driver are the same as those in Section 7.2.4.a: *Connection cache with iAnywhere JDBC driver*.

STEP 1 – Start EAServer and log in with Jaguar Manager.

STEP 2 – Right click on *Connection Caches* in the tree view.

STEP 3 – Type the name of the connection cache, and click *Finish.*

STEP 4 – On the *Connection Cache Properties* dialog box that appears, enter the settings under the General tab. The parameters you type are case sensitive.

Syntax for Server Name: jdbc:odbc:DSNname.

STEP 5 – Click the *Driver* tab. Check the JDBC radio button and type the driver string into the DLL or Class Name text box.

The driver string should be: sun.jdbc.odbc.JdbcOdbcDriver.

STEP 6 – Click the *Caching* tab. Make sure to check both the *Enable cache-by-name access* option and the *Enable connection sanity check* option.

STEP 7 – Return to the General tab. Click the Refresh button, and then click Ping. Make sure the ping succeeds before you continue.

STEP 8 – Close the *Connection Cache Properties* dialog box.

The connection cache has been added.

### 7.2.5 Setting up connection cache for Oracle

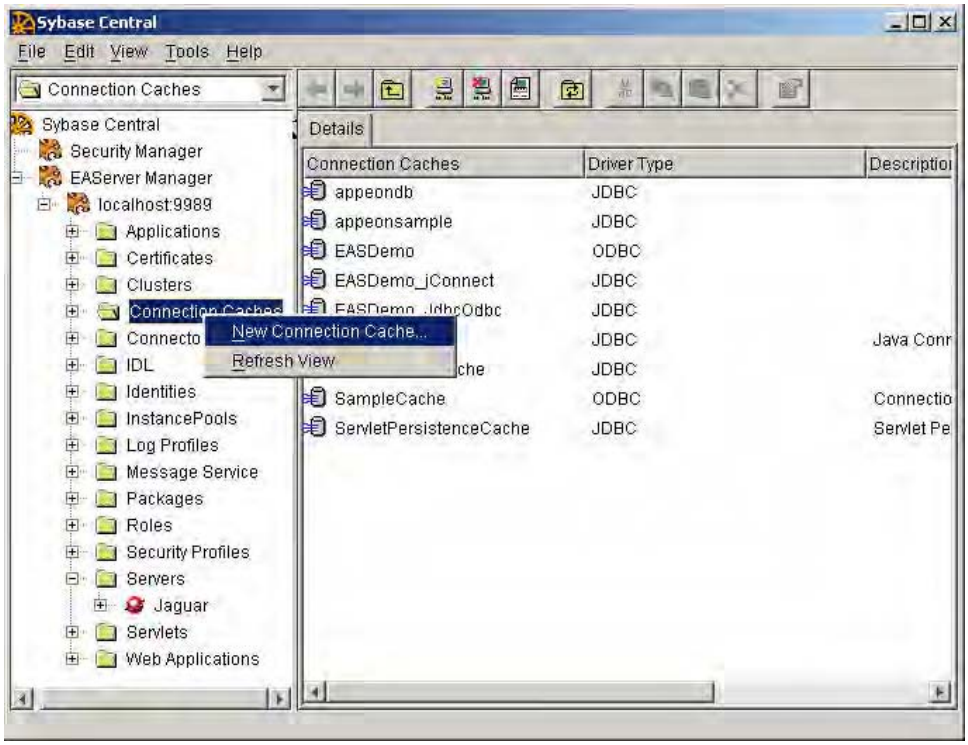
Appeon recommends using the Oracle JDBC driver rather than the JDBC-ODBC bridge driver for Oracle databases. The following instructions deal with setting up connection caches that use the Oracle JDBC driver.

STEP 1 – Start EAServer and log in with EAServer Manager.

For instructions on how to access EAServer using EAServer Manager, please refer to the *EAServer System Administration Guide*.

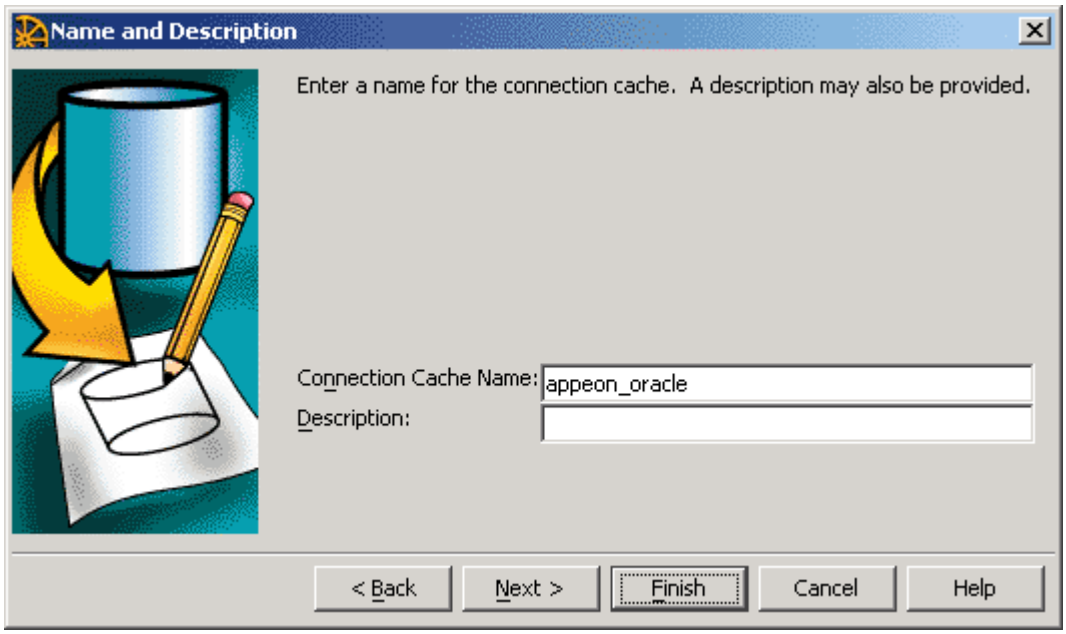
STEP 2 – Right-click on *Connection Caches* in the left tree. Choose *New Connection Cache* from the context menu, as shown in Figure 7-10.

**Figure 7-10: New Connection Cache**



STEP 3 – Type the name of the connection cache and click *Finish*, as shown in Figure 7-11. In this example, the connection cache name is *appeon_oracle*.

**Figure 7-11: Create Connection Cache**

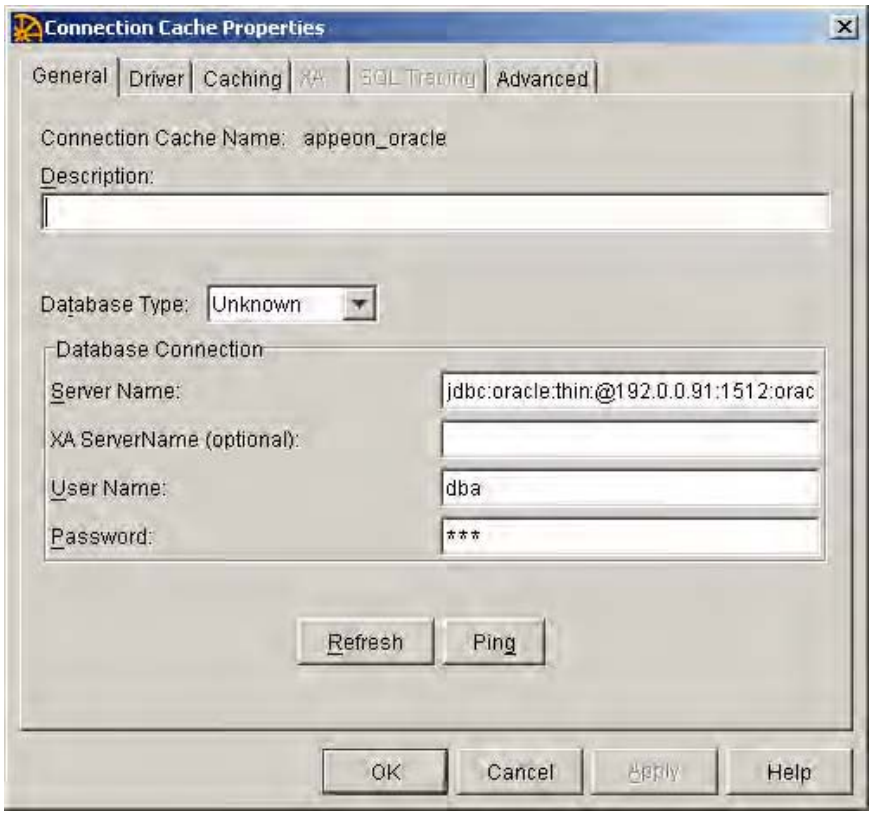



STEP 4 – On the *Connection Cache Properties* dialog box that appears, enter the following settings under the General tab. The parameters you type are case sensitive.

**Table 7-3: Connection Cache Properties**

| | |
|---|---|
| Database Type | Select “Unknown” or the specific database type (Oracle 8i or 9i) |
| Server Name | Establish a connection to the database by specifying the DB URL.<br>Syntax: jdbc:oracle:thin:*@hostname*:*port*:*DBName*<br>e.g. jdbc:oracle:thin:@192.0.0.91:1512:oracletest01 |
| User Name | Type the database login username. The username is set at the database server. |
| Password | Type the database login password. The password is set at the database server. |

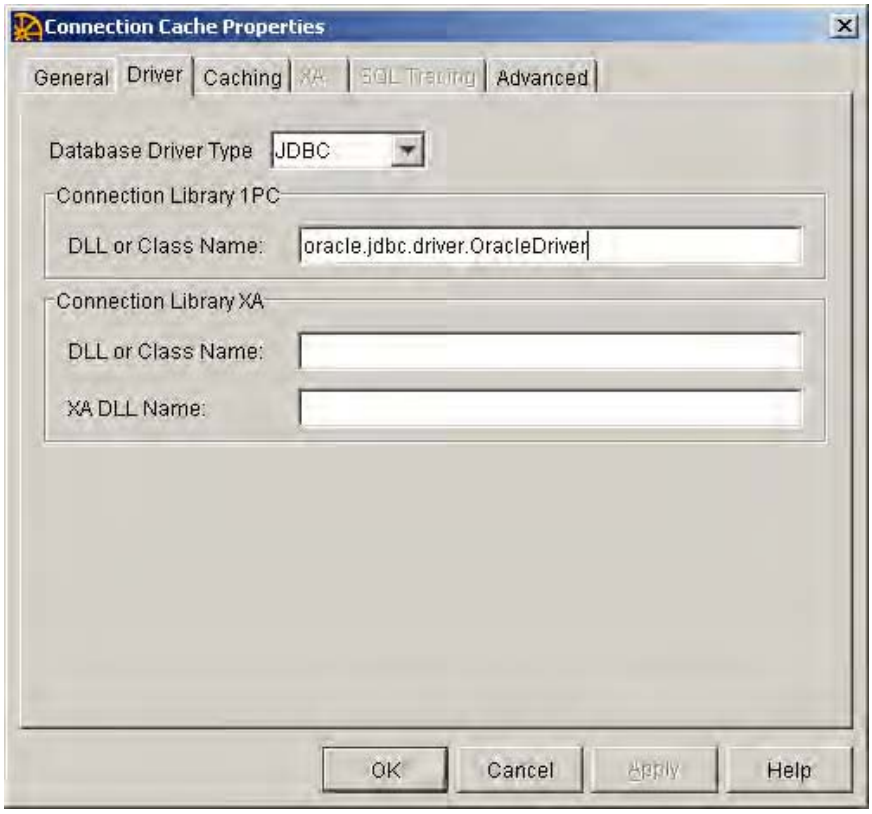
**Figure 7-12: Connection Cache Properties**




STEP 5 – Click the *Driver* tab. Choose *JDBC* from the DropDownListBox and type the driver string into the DLL or Class Name text box, as shown in Figure 7-13.

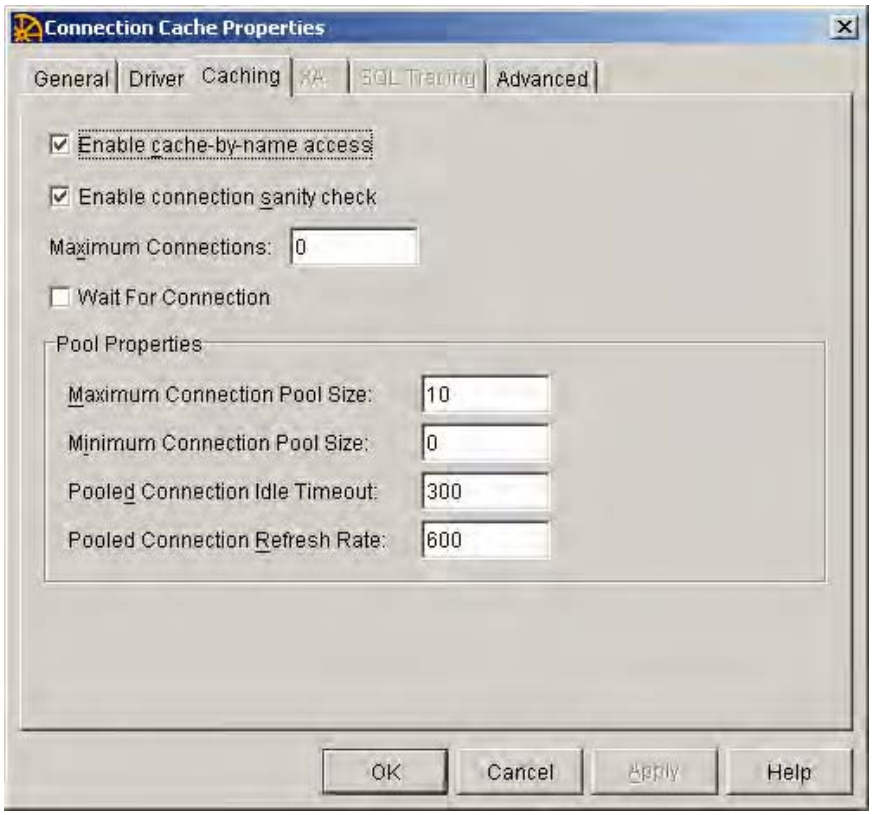
The string in the DLL or Class Name text box is for loading the JDBC driver. Since the connection cache will use the Oracle JDBC driver, the string entered should be exactly as shown here: "*oracle.jdbc.driver.OracleDriver*".

**Figure 7-13: Connection Cache Properties**



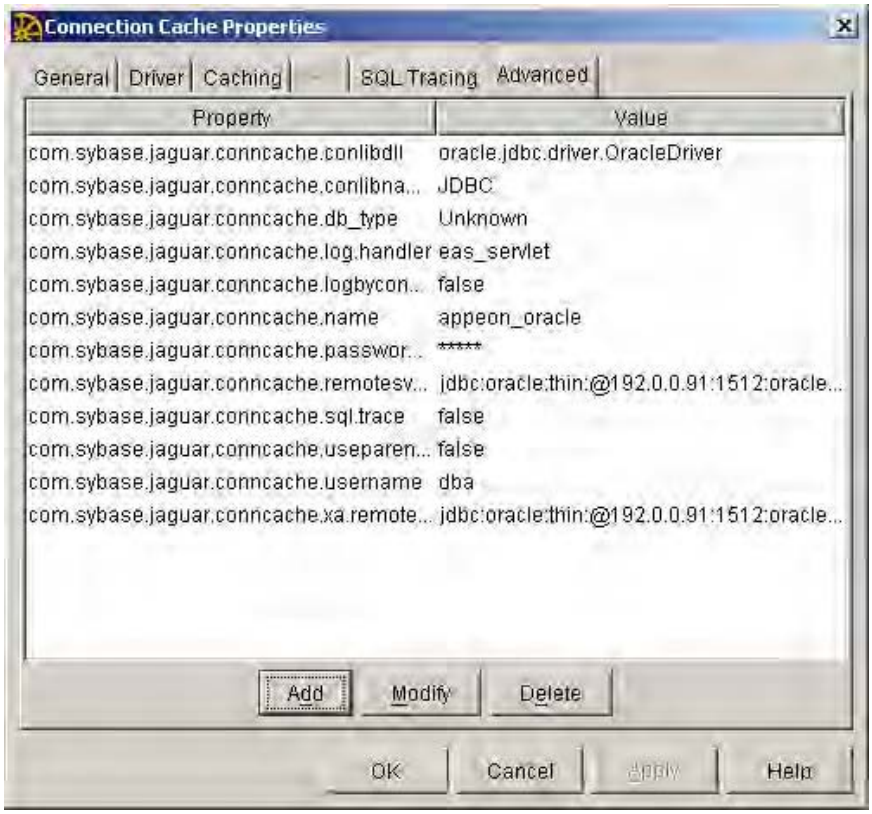

STEP 6 – Click the *Caching* tab. Make sure you check both the *Enable cache-by-name access* and *Enable connection sanity check* options, as shown in Figure 7-14.

**Figure 7-14: Connection Cache Properties**



STEP 7 – Click Add under the *Advanced* tab, as shown in Figure 7-15.

**Figure 7-15: Connection Cache Properties**




STEP 8 – Return to the *General* tab. Click the Refresh button, and then click Ping. Make sure the ping succeeds before you continue.

STEP 9 – Close the *Connection Cache Properties* dialog box.

The connection cache has been added.

### 7.2.6 Setting up connection cache for IBM DB2

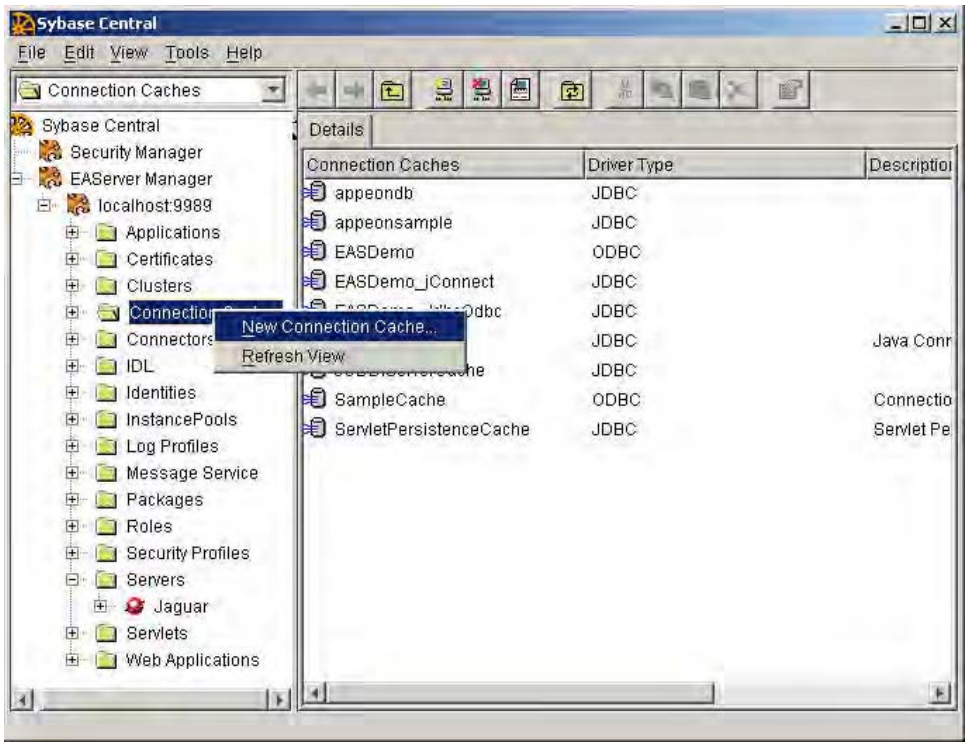
Appeon recommends using the IBM JDBC driver rather than the JDBC-ODBC bridge driver for IBM DB2 databases. The following instructions deal with setting up connection caches that use the IBM JDBC driver.

STEP 1 – Start EAServer and log in with EAServer Manager.

For instructions on how to access EAServer using EAServer Manager, please refer to the *EAServer System Administration Guide*.

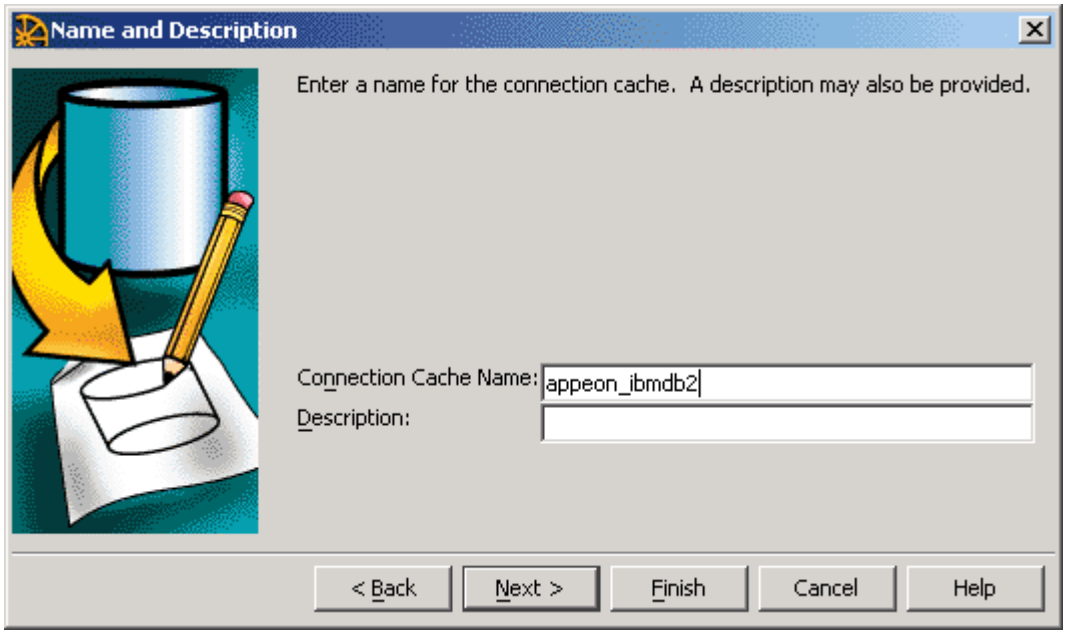
STEP 2 – Right-click on *Connection Caches* in the tree view. Choose *New Connection Cache* from the context menu, as shown in Figure 7-16.

**Figure 7-16: New Connection Cache**




STEP 3 – Type the name of the connection cache and click *Finish*, as shown in Figure 7-17. In this example, the connection cache name is *appeon_ibmdb2*.

**Figure 7-17: Create Connection Cache**

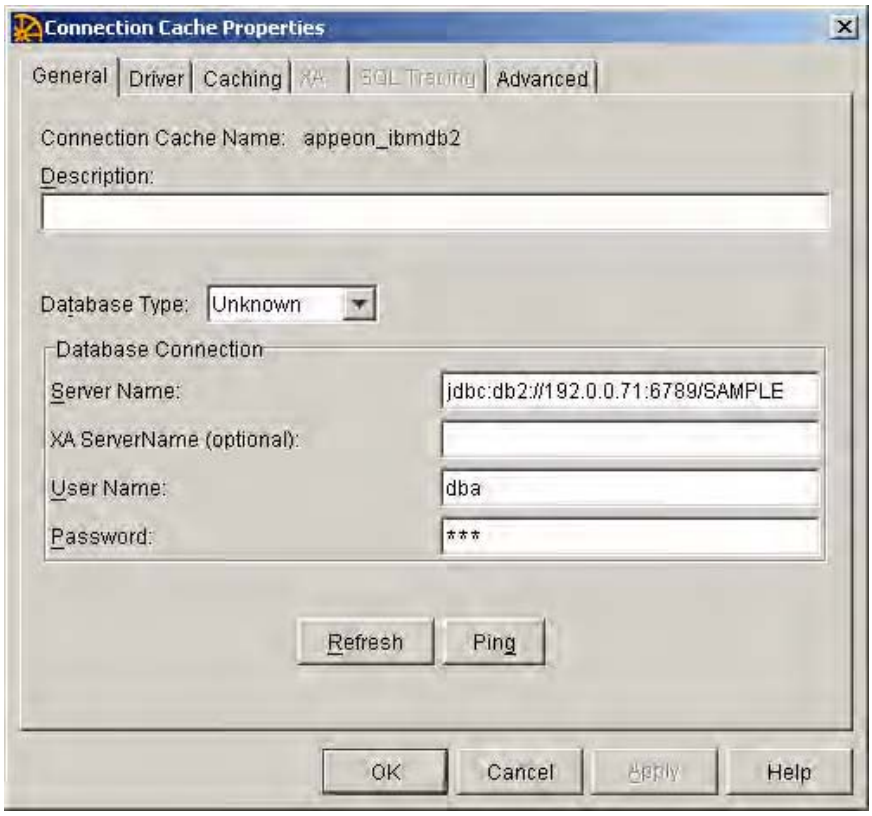


STEP 4 – On the *Connection Cache Properties* dialog box that appears, as shown in Figure 7-18, enter the settings under the *General* tab, as shown in Table 7-4. The parameters you type are case sensitive.

**Table 7-4: Connection Cache Properties**

| | |
|---|---|
| Database Type | Select "Unknown" for the Database type |
| Server Name | Establish a connection to the database by specifying the DB URL.<br>Syntax: jdbc:db2://*hostname*:*port*/*DBName*<br>e.g. jdbc:db2://192.0.0.71:6789/SAMPLE<br>Note: The port used by JDBC will be different than the one used by db2 client. The default JDBC port is 6789 while the port used by db2 client is 50000. |
| User Name | Type the database login username. The username is set at the Database Server. |
| Password | Type the database login password. The password is set at the Database Server. |

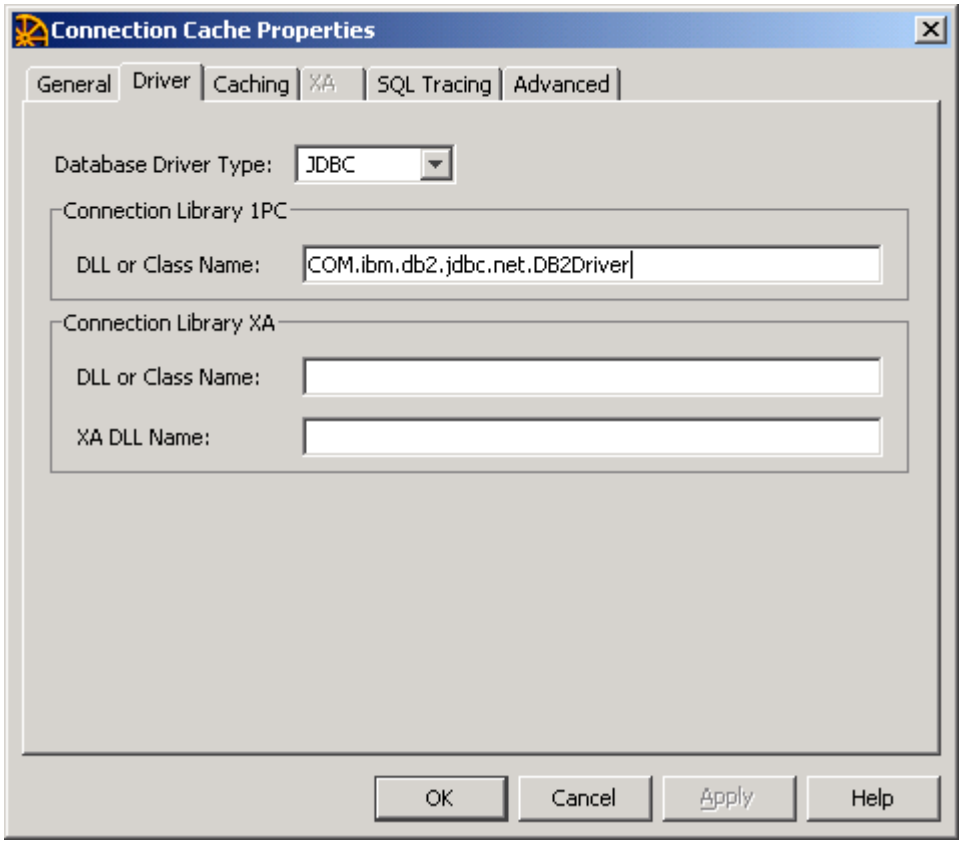
**Figure 7-18: Connection Cache Properties**



STEP 5 – Click the *Driver* tab. Choose *JDBC* from the DropDownListBox and type the driver string in the *DLL or Class Name* text box, as shown in Figure 7-19.

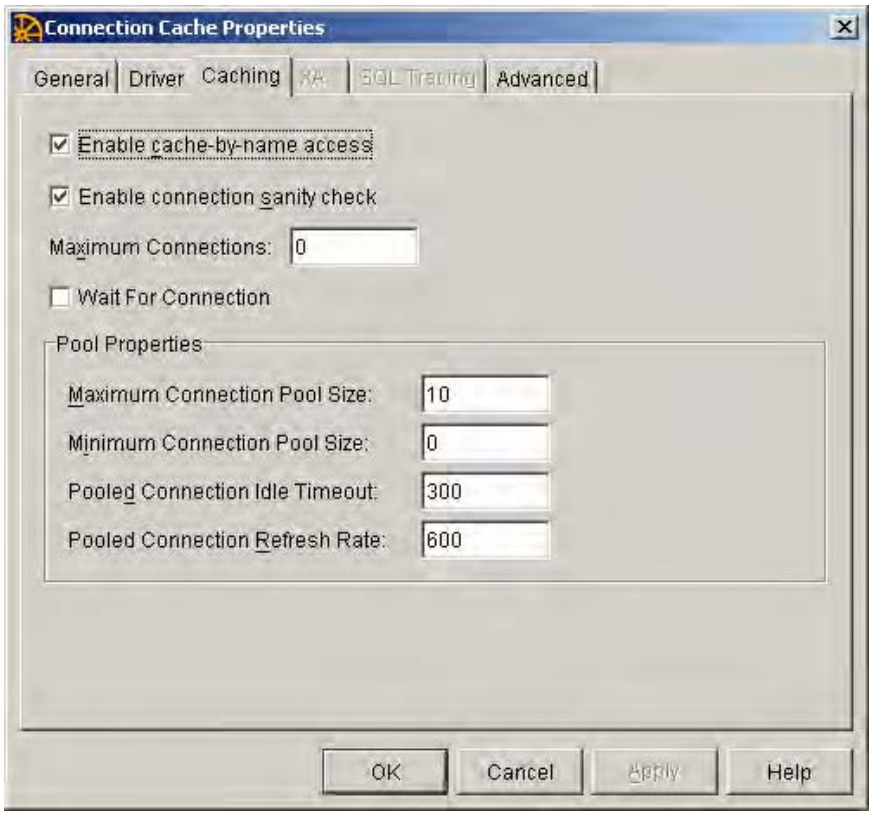
The string in the *DLL or Class Name* text box is "COM.ibm.db2.jdbc.net.DB2Driver" for loading the JDBC driver. The connection cache will use the IBM JDBC driver.

**Figure 7-19: Connection Cache Properties**



STEP 6 – Click the *Caching* tab. Make sure you check both the *Enable cache-by-name access* and *Enable connection sanity check* options, as shown in Figure 7-20.

**Figure 7-20: Connection Cache Properties**




STEP 7 – Return to the General tab. Click the Refresh button, and then click Ping. Make sure the ping succeeds before you continue.

STEP 8 – Close the *Connection Cache Properties* dialog box.

The connection cache has been added.

### 7.2.7 Setting up connection cache for Microsoft SQL Server

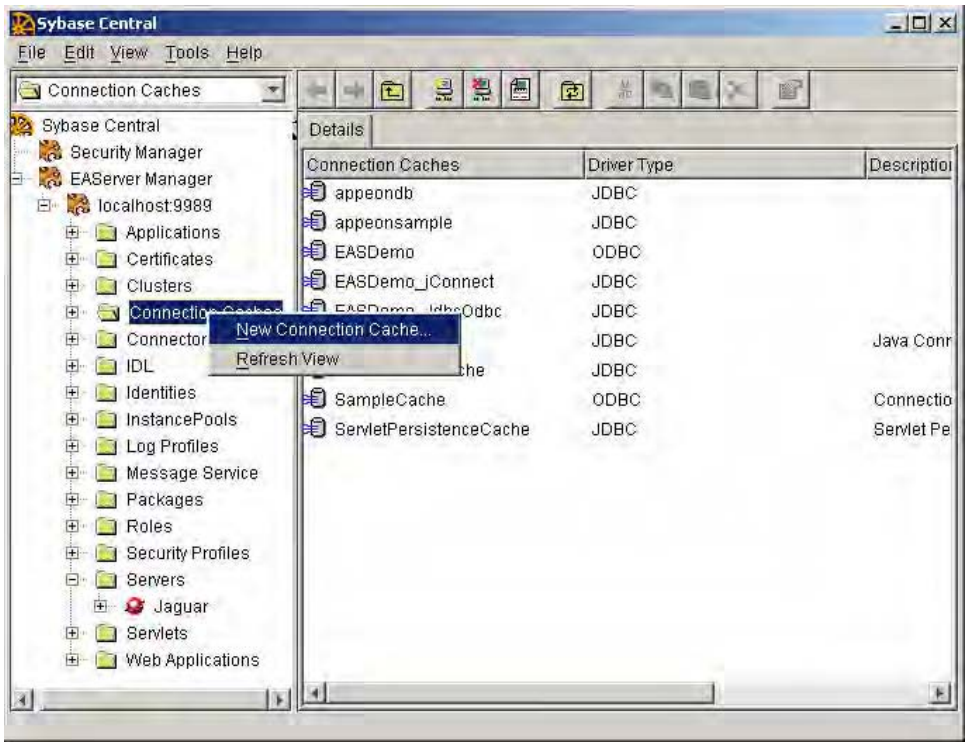
Appeon recommends using the Microsoft SQL Server JDBC driver rather than the JDBC-ODBC bridge driver for SQL Server databases. The following instructions deal with setting up connection caches that use the SQL Server JDBC driver.

STEP 1 – Start EAServer and log in with EAServer Manager.

For instructions on how to access EAServer using EAServer Manager, please refer to the *EAServer System Administration Guide*.

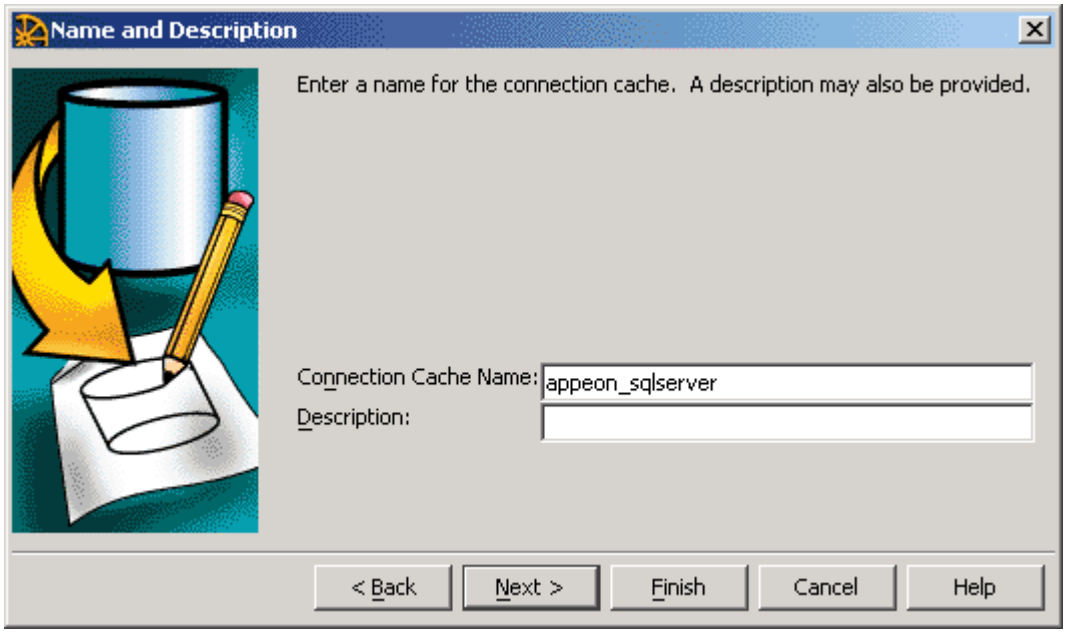
STEP 2 – Right-click on *Connection Caches* in the left tree. Choose *New Connection Cache* from the context menu, as shown in Figure 7-21.

**Figure 7-21: New Connection Cache**




STEP 3 – Type the name of the connection cache and click *Finish*, as shown in Figure 7-22. In this example, the connection cache name is *appeon_sqlserver*.

**Figure 7-22: Create Connection Cache**

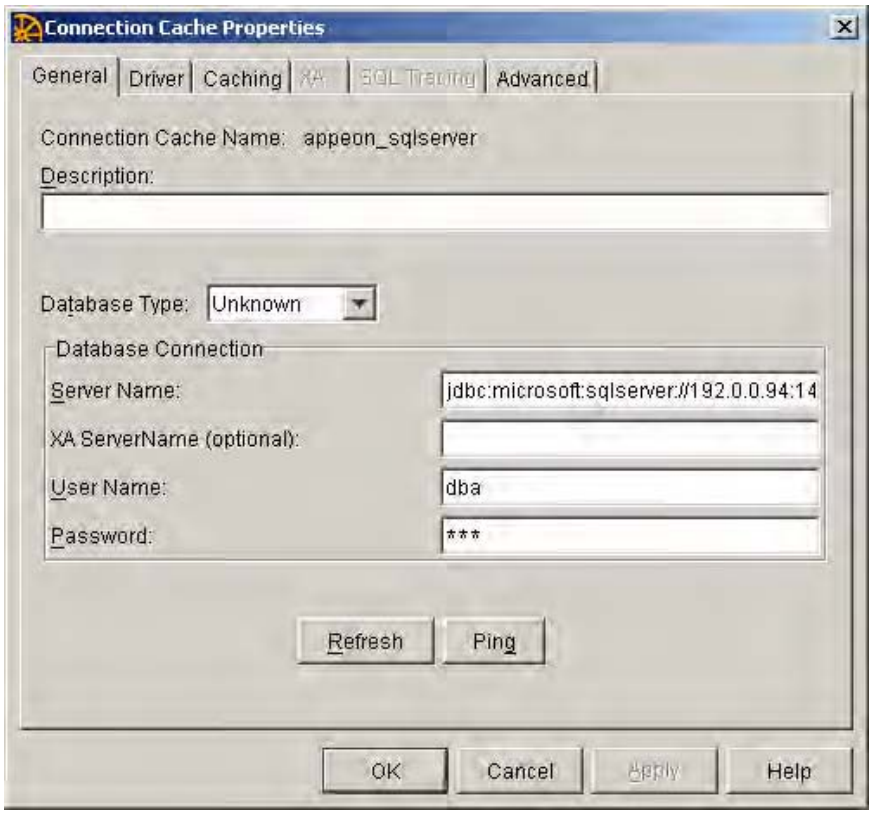


STEP 4 – On the Connection Cache Properties dialog box that appears, as shown in Figure 7-23, enter the settings under the General tab, as shown in Table 7-5. The parameters you type are case sensitive.

**Table 7-5: Connection Cache Properties**

| | |
|---|---|
| Database Type | Select "Unknown" or the specific database type (SQL Server) |
| Server Name | Syntax:<br><br>jdbc:microsoft:sqlserver://*hostname*:*port*;DatabaseName=*DBName*;SelectMethod=cursor<br><br>e.g. jdbc:microsoft:sqlserver://192.0.0.94:1433;DatabaseName=sqltest;SelectMethod=cursor |
| User Name | Type the database login username. The username is set at the database server. |
| Password | Type the database login password. The password is set at the database server. |

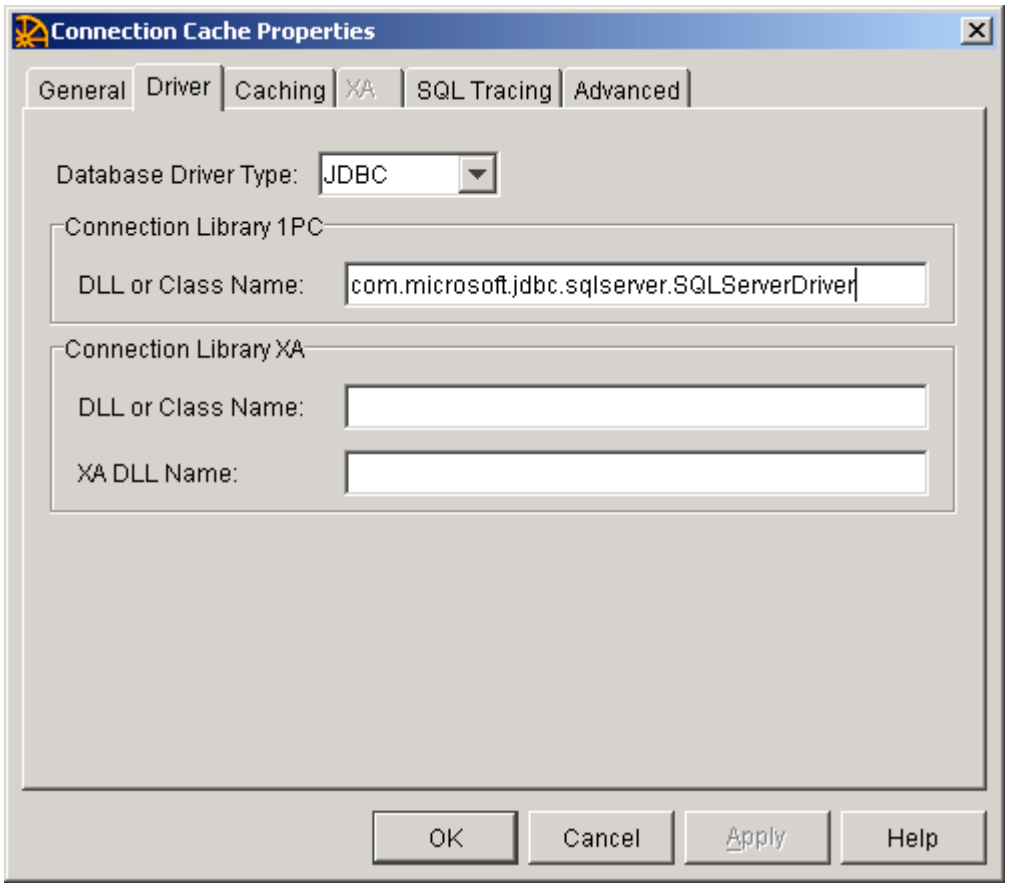
**Figure 7-23: Connection Cache Properties**



STEP 5 – Click the *Driver* tab. Choose *JDBC* from the DropDownListBox and type the driver string "*com.microsoft.jdbc.sqlserver.SQLServerDriver*" in the *DLL or Class Name* text box, as shown in Figure 7-24.

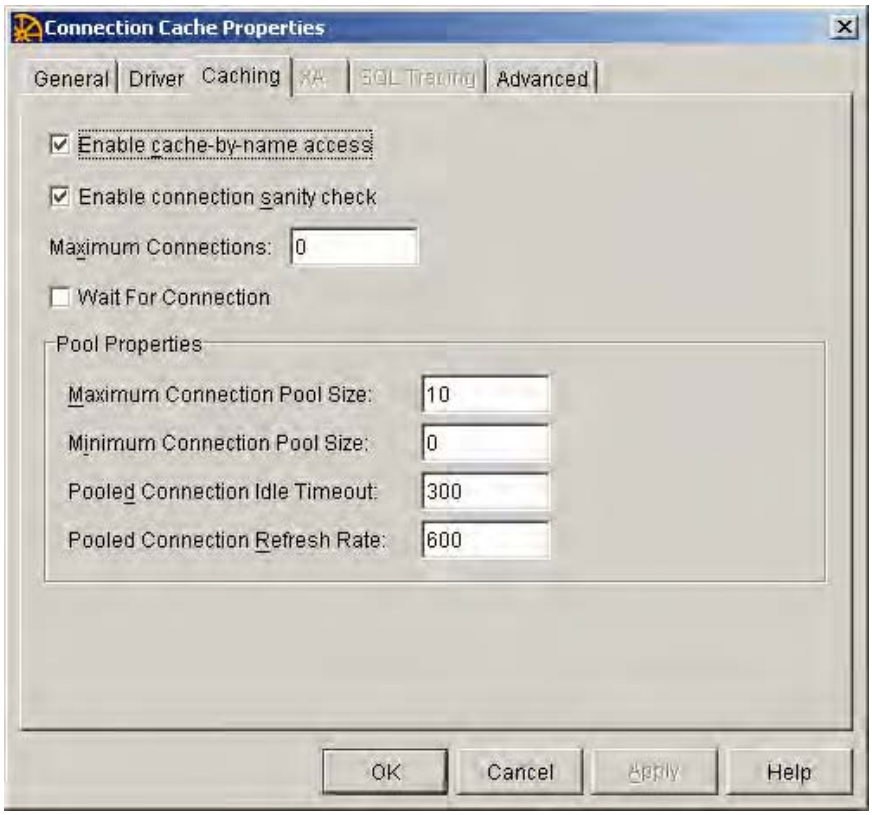
The string in the DLL or Class Name text box is for loading the JDBC driver. The connection cache will use the SQL Server JDBC driver.

**Figure 7-24: Connection Cache Properties**



STEP 6 – Click the *Caching* tab, as shown in Figure 7-25. Make sure you check both the *Enable cache-by-name access* and *Enable connection sanity check* options.

**Figure 7-25: Connection Cache Properties**



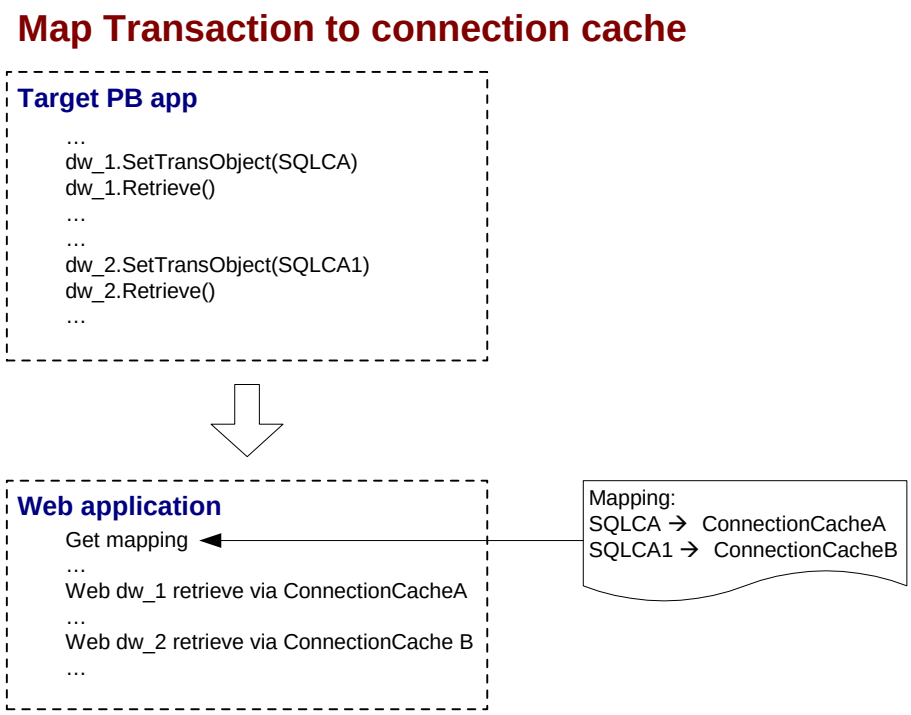

STEP 7 – Return to the General tab. Click the Refresh button, and then click Ping. Make sure the ping succeeds before you continue.

STEP 8 – Close the *Connection Cache Properties* dialog box. The connection cache has been added.

## 7.3 Setting up transaction object to connection cache mapping

The purpose of setting up the mapping is to make sure the configured connection cache can access the database server for the Appeon Web application as the replacement of the transaction object in the PowerBuilder application, as shown in Figure 7-26.

**Figure 7-26: Map transaction to connection cache**




Once Appeon Server connection caches are configured, you can set up the transaction object to connection cache mapping in two different ways:

- Higher priority: Dynamic transaction object to connection cache mapping via PowerScript.
- Lower priority: Static transaction object to connection cache mapping in AEM. The mapping in PowerScript has priority over the static mapping in AEM.

Dynamic mapping is of higher priority, meaning that if a transaction object named "SQLCA" is both mapped to connection cache A via PowerScript and mapped to connection cache B in AEM, the transaction in effect is mapped to connection cache A.

### 7.3.1 Dynamic transaction object to connection cache mapping

Transaction object to connection cache mapping can be dynamically set up or changed by setting or changing the DBMS and DBParm properties of the Transaction object in the application source code.

To set or change the connection cache dynamically, code the DBParm property of the Transaction object in this format:

```
SQLCA.DBParm="CacheName='ASEConnectionCache1'"
```

"ASEConnectionCache1" can be replaced by the name of the connection cache you want to use for the Transaction object.

To set or change the database type dynamically, code the DBMS property of the Transaction object using this format:

```
SQLCA.DBMS = "odb-asa9"
```

The value of the DBMS property should be set based on the database type; refer to Table 7-6.

**Table 7-6: Setting the DBMS property based on the database type**

| Database Type | DBMS Setting (short name) | DBMS Setting (full name) |
|---|---|---|
| Sybase ASE | SYC | SYC Adaptive Server Enterprise |
| Sybase ASE for EAServer | SYJ | SYJ Adaptive Server Enterprise (EAServer Components) |
| Sybase ASA 7.x | Odb-asa7 (Appeon syntax) | odb-asa7 (Appeon syntax) |
| Sybase ASA 8.x | ODB or<br>odb-asa (Appeon syntax) or<br>odb-asa8 (Appeon syntax) | ODBC or<br>odb-asa (Appeon syntax) or<br>odb-asa8 (Appeon syntax) |
| Sybase ASA 9.x | odb-asa9 (Appeon syntax) | odb-asa9 (Appeon syntax) |
| Microsoft SQL Server | MSS | MSS Microsoft SQL Server |
| Oracle 8i | O84 | O84 Oracle8/8i |
| Oracle 9i | O90 | O90 Oracle9i |
| Oracle 10g | O90 | O90 Oracle9i |
| IBM DB2 UDB 7.2 | odb-db2 (Appeon syntax) | odb-db2 (Appeon syntax) |
| IBM DB2 UDB 8.1 | odb-db2 (Appeon syntax) | odb-db2 (Appeon syntax) |

In Table 7-6:

The short and full names are case-insensitive (for example: ODBC is the same as odbc).

If ODB or ODBC is set as the DBMS property, Appeon will regard the database type as Sybase ASA. The “odb-asa” and “odb-db2” are Appeon defined values. They can be recognized by Appeon without affecting the running of the PowerBuilder application, because only the first three letters of the DBMS setting are valid in PowerScript syntax.

### 7.3.2 Static transaction object to connection cache mapping

For an Appeon Web application, you can set up transaction object to connection cache mapping in the Application Properties settings in AEM. This is a static way for mapping the Transaction object to the connection cache.

For detailed instructions on how to set up database configuration in AEM, please refer to the *Appeon Enterprise Manager User Guide*.

## 7.4 Advanced configurations related with database connection

### 7.4.1 Application security

For typical PowerBuilder applications, security is implemented at two levels: script coded security and database security. After Web conversion, the Appeon system provides an additional built-in layer of Web application security on top of PowerBuilder application security. Appeon security is “either-or”: the user either has or does not have access to the Web application.

You can implement security for deployed Appeon Web applications in many ways. PowerBuilder script-coded security can convert direct to the Web, and it provides security for the Web applications. There are also ways to implement database security in Appeon Web applications. Finally, you can use the Appeon user/group management system to restrict access to Appeon Web applications.

In addition, a way to incorporate the Appeon user/group management for use with the coded security in PowerBuilder applications is discussed in Section 7.4.2.a. You can also implement your own Web security using other Web technologies.

### 7.4.1.a Database security

Depending which user logs into an application, a PowerBuilder application can dynamically change the Transaction properties (user ID and password etc.) and connect to the database with different identities that determine the user privileges to access, read or modify the database tables.

Appeon Web applications rely on the EAServer JDBC connection caches to interact with the Database Servers. In the Web application, transaction object to connection cache mapping can be dynamically set up or changed by setting or changing the DBMS and DBParm properties of the Transaction object in the application source code, or it can be statically set up in AEM database configuration. There is a limitation with connection cache configuration: the user ID and password of a connection must be pre-configured in EAServer. Due to this limitation, you may need to consider the workarounds introduced in this section to better migrate the database security in the original application.

***Workaround one: Predefined connection caches***

To work around this unsupported feature, you can pre-define in EAServer Manager a certain number of connection caches that correspond to different security access levels in the database with different user IDs and passwords. When the user logs in, the application decides which transaction object to connection cache mapping to use for establishing the database connection.

You should set up an equal number of connection caches in EAServer that connect to the database with different privileges, and map the connection caches dynamically using the Transaction DBParm property to the PowerBuilder Transaction objects. Transaction object to connection cache mapping can be dynamically set up or changed by setting or changing the DBMS and DBParm properties of the Transaction object in the application source code. See Section 7.3.1: *Dynamic transaction object to connection cache mapping* for the details.

***Workaround two: The distributed application technique***

The distributed application technique is supported by Appeon to work around the database security features in a PowerBuilder application. The methodology is to encapsulate the PowerBuilder source code that implements the database securities into NVOs and run them on EAServer. The NVOs may perform actions like creating database connections (manipulation of Transaction object properties), manipulating DataStore objects or executing SQL statements, and transferring the returned information to the Client.

For example, you can encapsulate the user authentication logic into an NVO, and then deploy the NVO to EAServer. In the application source code, you only need to pass the username and the password from the login window to the NVO.

STEP 1 – Create an NVO and declare a method in the NVO for user authentication. The method compares the user information entered in the application login window with that retrieved from the system table.

The method declared in the NVO:

```
public function boolean of_checkuser (string as_userid, string
as_password);
String  ls_DBPass
SELECT fpassword
      INTO :ls_DBPass
      FROM t_user
      WHERE t_user.fuserno = :as_userid;
If sqlca.SQLCode <> 0 Then
      Return false
End If
If ls_DBPass <> as_password Then
      Return false
End If
Return true
End function
```

STEP 2 – Deploy the NVO to EAServer and create a proxy object for the NVO in the Client application.

STEP 3 – Modify the user authentication logic in the Client application to get the user information from the login window and pass it to the NVO.

The modified script in the Client application:

```
String  ls_User,ls_Pass,ls_DBPass,ls_Err
Long  ll_ID
String ls_Mess = 'Please enter a valid user ID/password.'

If Trim(sle_name.Text) = '' Then
      sle_name.SetFocus()
      MessageBox('Login Not Valid',ls_Mess,exclamation!)
      Return
End if
ls_User = Trim(sle_name.Text)
ls_Pass  = Trim(sle_Pass.Text)
If i_logsec.of_checkuser(ls_User,ls_Pass) then
      MessageBox("Infomation","Login successful!")
Else
      MessageBox("Infomation","Login failure!")
End if
```

#### 7.4.1.b Using INI files for connection security

You can set connection properties for a PowerBuilder application either by assigning values to the properties in the application script or using PowerScript Profile functions to read from an initialization (INI) file. It is recommended in Appeon that you set connection properties by reading from INI files only if your environment meets the following requirements:

- The browser for accessing the application must be cookie-enabled.

  Reason: Appeon Developer deploys the INI files as XML to Appeon Server. When a Client accesses the deployed application that uses the INI file profiles, a copy of the

original XML file is specially created and carries all the profile information for that Client. The cookie on the Client browser enables the Client to read the correct copy of its XML file located on Appeon Server.

- Make sure the Windows user account profile on the Client is only used by one user for accessing the application.

  Reason: As the Cookie will reside in the Windows user profile cookie directory (for example, *C://Documents and Settings/Administrator/Cookies*) any user with full access rights who also uses the Client computer will be able to gain access to another user's Web application identity.

If the same Windows user account profile will be used by multiple users on the Client, consider using another security method, Database security, as introduced in Section 7.4.1.a, *Database security*.

The initialization file should consist of several sections including *PB*, *Application*, and *Database*:

```
[Database]
variables and their values
...
```

The following script example assigns connection properties to SQLCA. The database connection information is stored on the Web Server after application deployment; on some network configurations this can leave the database server insecure:

```
SQLCA.DBMS = "MSS Microsoft SQL Server"
SQLCA.Database = "appeon_test"
SQLCA.ServerName = "192.0.0.246"
SQLCA.LogId = "sa"
SQLCA.AutoCommit = False
...
```

To set the Transaction object to connect to a database, the following script example reads values from App.INI, an initialization file. This method is much more secure in comparison to the preceding script.

```
sqlca.DBMS = ProfileString(App.INI, "database",&
   "dbms", "")
sqlca.database = ProfileString(App.INI,&
   "database", "database", "")
sqlca.userid = ProfileString(App.INI, "database",&
   "userid", "")
sqlca.dbpass = ProfileString(App.INI, "database",&
   "dbpass", "")
...
```

### 7.4.2 Appeon security

Appeon security features are set in Appeon Enterprise Manager (AEM), the Web application that manages the Appeon system and deployed Web applications. Appeon security is at the Web application level and is "either or": the user either has or does not have access to the Web application. By default, Appeon security is turned off for each deployed Web application.

When the security for a Web application is turned on, the Appeon Login Web dialog box pops up at the beginning of the Web application startup and prompts the user to enter the user

name and password. The user name and password is verified by Appeon Server against the authentication schema that can be set in an LDAP server or in Appeon system database. If the user name or password is not correct, the user is not allowed to access the Appeon Web application.

For more information on using Appeon security features for Appeon Web applications, please refer to the *Appeon Enterprise Manager Guide*.

### 7.4.2.a Incorporate Appeon security in PowerBuilder code

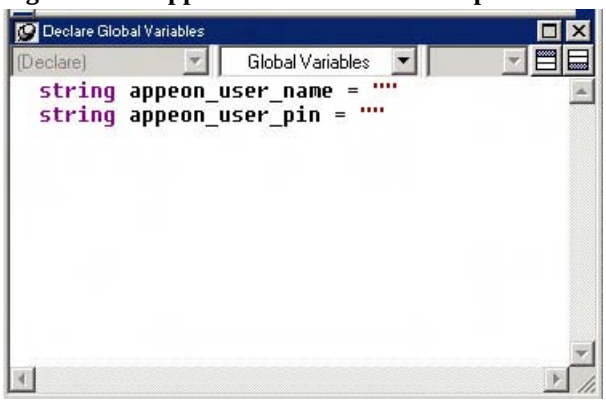
If your PowerBuilder application has not coded user name/password verification at application startup that restricts access to the application, you can utilize Appeon's built-in user group management. When the Web application runs, the user is prompted to enter the Appeon Web user name and password in the Appeon Login Web dialog box.

There is a way to pass the Appeon Web user name and password to the PowerBuilder application so that you can utilize the user name and password to implement script coded security features for your PowerBuilder application.

STEP 1 – Define two global variables in your PowerBuilder application. The variables are String data type, and their names should be exactly *appeon_user_name* and *appeon_user_pin*.

When the PowerBuilder application is converted to the Web, the Appeon system automatically assigns the values of the Appeon Web user name and password to *appeon_user_name* and *appeon_user_pin* respectively, as shown in Figure 7-27, after the user successfully logs into the Web application.

**Figure 7-27: Appeon user name and user pin**




STEP 2 –Code anything in the PowerBuilder application, based on the values of the two variables, that is needed for carrying out security actions like saving account information in INI upon the initial login or hiding/displaying menu items. )

The Appeon Web user name and password can be set in the *Security* section of *Appeon Enterprise Manager User Guide*.

### 7.4.2.b Database auditing

In Client/Server architecture, the database can easily keep track of every logged-in user if you enable the AUDITING option in the database.

Appeon deployed Web applications run in a three-tier architecture. Each time the Client wants to connect with the database, the call reaches Appeon Server first. Appeon Server will validate the user ID and password of the call. If the validation passes, Appeon Server connects with the Database Server using a unified user ID and password. The user ID and

password that the database keeps track of is not the user ID and password that makes the call at the Client.

***Passing user ID/password to database from EAServer connection cache***

If you are using a Sybase ASE database, you can use the SSA connection cache property. This property changes the ID at the database to whatever user ID/Password is used by end users for accessing the server. If you are using a Sybase database, you can set this property in your connection cache props file. This cannot be used if you are using a different database type.

The following information is taken from the EAServer Administrator Guide Appendix B - Connection Cache Properties; please refer to the EAServer documentation for more detailed instructions.

The connection cache property, *com.sybase.jaguar.conncache.ssa,* enables set-proxy support for connections to databases that support this feature. By default, the property is set to false, which disables set-proxy support.

This feature can be used with any database that recognizes this command:

```
set session authorization "login-name"
```

When proxy support is enabled, connections retrieved from the cache are set to act as a proxy for the user name associated with the EAServer client. To set the proxy to another user name, use the Java JCMCache.getProxyConnection() method or the C JagCmGetProxyConnection() routine in your component.

The user name specified in the cache properties (com.sybase.jaguar.conncache.username) must have set-proxy privileges in the database and/or server used by the cache.

In EAServer Manager, set this property using the All Properties tab in the Connection Cache Properties dialog box.

***Re-configuring database auditing functionality***

To work around the database auditing functionality, you can also re-configure the auditing information that is saved on the database by adding a new field to it: user ID.

With the Client/Server application, make sure that a combination of user ID and password cannot hold multiple connections with the database at one time.

Add in the necessary code in the Client Server application so that every time the user wants to connect with the database, the call sent to the Database Server includes user ID information. For example, when sending the user ID as a column in the DataWindow or to the Stored Procedure, the user ID information in the call from the client-side will be saved in the user ID field on the Database Server.

# 8 Enhancing an application with Web or Appeon features

## 8.1 Overview

Table 8-1 gives you a general idea on how to enhance applications with Web or Appeon features.

**Table 8-1: Enhance a deployed application with Web/Appeon features**

| Type of the Feature | Description of the Feature | How to add it in the deployed application |
|---|---|---|
| Typical Web features | • Accessibility of the application via URLs or Sybase Enterprise Portal<br>• Firewall compatibility. The Web file transfer is HTTP over port 80.<br>• SSL and digital certificate security<br>• Superior scalability. Typical applications support 60-80 concurrent users per server CPU that maps to 300-500 named/end users per server CPU.<br>• More… | Such features are automatically added in the deployment application, with no need for any special coding.<br>Refer to Section 8.4: *Loading an application in Sybase Enterprise Portal* for details on how to load the application in Enterprise Portal. |
| Appeon Server open interfaces | Use Appeon Server open interfaces to manage Appeon-deployed Web applications using PowerBuilder code. | The interface(s) should be called in the PowerBuilder application to take effect in the deployed application.<br>Refer to Section 8.2: *Appeon Server open interfaces* for details. |
| Appeon client functions | Use Appeon client functions to get Client information or enable Appeon DataWindow menu for DataWindow(s) in the applications. | The client function(s) should be called in the PowerBuilder application to take effect in the deployed application.<br>Refer to Section 8.3: *Appeon client functions* for details. |
| Web integration | • Provides flexible & open Java, .NET, & Web Services integration<br>• Connectivity to Java/EJB, PB NVO, C/C++ DLL, COM/ActiveX Components on Application Servers<br>• Connectivity to Web Services, J2EE, and .NET<br>• Connectivity to Messaging Queues (MQSeries, JMS, etc.) | There are different ways for implementing the integration of an Appeon Web application with other applications.<br>Refer to Section 8.5: *Single sign-on* and 8.6: *Integrating Appeon Web applications with JSP/ASP* for details. |

## 8.2 Appeon Server open interfaces

### 8.2.1 Overview

Appeon Server open interfaces give you the opportunity to manage services provided by Appeon Server through PowerBuilder code. In Appeon 2.8 and earlier versions, you could only use the Appeon Server AEM system to manage Appeon-deployed Web applications. In Appeon 3.1, you can easily manage Appeon-deployed Web applications using PowerBuilder code.

Appeon Server provides three open interfaces:

- GetSessionCount
- KillAllSessions
- RollbackAllTransactions

### 8.2.2 Description of the open interfaces

Refer to *Appeon Help | Web enhancements and differences* for the syntax of Appeon Server open interfaces. Note that you can write code for calling the interface in the PowerBuilder application, but the code does not take effect in the PowerBuilder application; it only takes effect in the Appeon-deployed application.

#### 8.2.2.a GetSessionCount

With GetSessionCount method, you can get the following three types of information.

- The total number of active sessions opened for the specified application in the specified Appeon Server.
- The total number of active sessions in a specified Appeon Server.
- The total number of sessions opened for the specified application in an Appeon Server cluster.

  To get the number of sessions in an Appeon Server cluster, you need to first configure the cluster in AEM.

By using the AppeonGetSessionCount method, you can easily get the total number of active sessions in a specified Appeon Server using PowerBuilder code and apply the information in other open interfaces such as KillSession to manage the sessions. For example, you can first call AppeonGetSessionCount and then call KillSession in the PowerBuilder application to make the deployed application kill all sessions in Appeon Server when there are up to 100 active sessions in the server.

#### 8.2.2.b KillAllSessions

KillAllSessions kills all active sessions in an Appeon Server or an Appeon Server cluster and rolls back all associated transactions. To kill all sessions in an Appeon Server cluster, you need to first configure the cluster in AEM.

#### 8.2.2.c RollbackAllTransactions

RollbackAllTransactions rolls back all transactions in an Appeon Server or an Appeon Server cluster. To roll back all transactions in an Appeon Server cluster, you need to first configure the cluster in AEM.

### 8.2.3 Applying Appeon Server open interfaces in Appeon-deployed applications

For both Appeon Pure-JavaScript and Appeon Xcelerator deployments, using Appeon Server open interfaces in the Web application is supported.

To apply an Appeon Server open interface in an Appeon-deployed Web application, you only need to perform the following two steps.

1. Call the Appeon Server open interface in the PowerBuilder application. Refer to Section 8.2.3.a for how to call Appeon Server open interfaces in a PowerBuilder application.
2. Deploy the PowerBuilder application to Appeon Server the same way you would deploy a normal PowerBuilder application.

#### 8.2.3.a Calling Appeon Server open interfaces

Appeon Server open interfaces (methods) are encapsulated in a standard EAServer component named *OpenInterface* that is included in the *ASInterface* package. You can call the open interfaces in PowerBuilder scripts the same way you would call any EAServer component interface.

To invoke an open interface (method) in the *OpenInterface* component, perform the following steps in the PowerBuilder application:

STEP 1 – Connect to the EAServer where Appeon Server is installed.

STEP 2 – Generate a proxy object for the component *OpenInterface*.

STEP 3 – Create an instance of the component *OpenInterface* by using the CreateInstance method.

STEP 4 – Invoke one or more methods in the component, GetSessionCount, KillAllSessions or RollbackAllTransactions.

Example:

The following example demonstrates how to call Appeon Server open interface *killAllSessions* in PowerBuilder script.

STEP 1 – Connect to the EAServer where Appeon Server is installed

```
Long       ll_rc
myconnect = create connection
myconnect.driver = "jaguar"
myconnect.location = "192.0.2.39:9100"
myconnect.application = "ASInterface"
myconnect.userID = "jagadmin"
myconnect.password = ""
ll_rc = myconnect.ConnectToServer()
IF ll_rc <> 0 THEN
   //MessageBox("Connection failed", ll_rc)
END IF
```

STEP 2 – Generate proxy object for EAServer component *OpenInterface*

For details on how to generate proxy object in PowerBuilder, refer to *Generating EAServer proxy objects | building an EAServer client* in PowerBuilder Help.

STEP 3 – Create an instance of the component

```
int rc
rc = myconnect.CreateInstance(mycomp, "ASInterface/OpenInterface")
IF IsValid(mycomp) = FALSE THEN
      Messagebox(‘’,’create instance failed’)
End if
```

STEP 4 - Invoke the method on the component

```
mycomp.AppeonGetSessionCount(“192.0.2.39”, “testApbInterface”)
```

## 8.3 Appeon client functions

### 8.3.1 Overview

Appeon client functions are a set of PowerBuilder global functions encapsulated in Appeon workaround PBLs. You can add Appeon client functions in your PowerBuilder applications to achieve the following in Appeon-deployed applications:

- Get client information such as the IP address, or the browser type of the client
- Enable Appeon DataWindow menu for specified DataWindows.

### 8.3.2 Description of Appeon client functions

Table 8-2 provides the client functions and their return values. Refer to *Appeon Help* | *Web enhancements and differences* for more detail. Note that some Appeon Client functions are just interfaces provided by the Appeon system; they have empty function bodies and empty return values for the PowerBuilder applications.

**Table 8-2: Client functions with their return values**

| Functions | Return value when executed in PowerBuilder | Return value when executed in Web application |
|---|---|---|
| AppeonPopMenu (Datawindow adw_dw, Integer nx, Integer ny) | None.<br>This function has no effect in PowerBuilder. | None.<br>At the execution of the function, Appeon DataWindow menu is displayed at the specified position on the specified DataWindow control.<br>AppeonPopMenu has higher priority than AppeonPopMenuOn. |
| AppeonPopMenuOn (Datawindow adw_dw, Boolean ab_show) | None.<br>This function has no effect in PowerBuilder. | None.<br>At the execution of the function, Appeon DataWindow menu pops up when you right click on the specified DataWindow. |
| AppeonGetAppeonUserName() | “” | The user name typed into the Appeon Web Login dialog. |
| AppeonGetBrowserVersion() | “” | Client Internet Explorer version |
| AppeonGetClientID() | Unique session identifier for PowerBuilder client | Unique session identifier for the Internet Explorer client |
| AppeonGetClientIP() | IP address of PowerBuilder client machine | IP address of the Internet Explorer client machine |
| AppeonGetClientType() | “PB” | “WEB” |
| AppeonGetHttpInfo(string attribute)<br>Note: This function is only available in Appeon Xcelerator deployment. | “” | The IP address of the Internet Explorer client. |
| AppeonGetOSType() | The type of the OS that runs the PowerBuilder client application. | The type of the OS that runs the Internet Explorer browser. |
| AppeonGetSessionCount | “” | The total number of active sessions opened for the specified application in the specified Appeon Server. |

### 8.3.3 Using Appeon Client functions

Although the main purpose of Appeon Client functions is to implement certain functionalities in Appeon-deployed applications, you must call the functions in your PowerBuilder application to make the functions effective.

To use Appeon Client functions in Appeon-deployed applications, perform the following three steps.

STEP 1 – Add Appeon Workaround PBL in your PowerBuilder project.

For Pure-JavaScript deployment, use the *appeon_workarounds_js.pbl* file in the *\Appeon\Developer\appeon_workarounds\JS* folder.

For Appeon Xcelerator deployment method, use the *appeon_workarounds_ax.pbl* file in the *\Appeon\Developer\appeon_workarounds\AX* folder.

STEP 2 – Use Appeon Client functions anywhere in the PowerBuilder application source code.

After any Appeon Client function is added, make sure the PowerBuilder application can be successfully built.

STEP 3 – Migrate your PowerBuilder Client/Server application into Web application with Appeon.

## 8.4 Loading an application in Sybase Enterprise Portal

### 8.4.1 Overview

Sybase Enterprise Portal (EP) is an open and scalable portal framework that provides an intuitive, secure and customized environment for end-users; rapid development and deployment tools for developers; and an easy-to-use console for administrators. For more description about Sybase Enterprise Portal, refer to http://www.sybase.com/products/developmentintegration/enterpriseportal.

Appeon-deployed applications can run in Sybase Enterprise Portal or Unwired Accelerator.

### 8.4.2 Restrictions on supporting Enterprise Portal

Appeon supports Enterprise Portal 6.1 Info Edition and Enterprise Edition. Be aware of the following restrictions when you load Appeon Web applications in Enterprise Portal:

- Only ONE Appeon Xcelerator Web application can be viewed inside one EP Page. Multiple Appeon Xcelerator Web applications can be viewed within multiple pages. For example, you can have Sales App Demo on one EP Page and Appeon Code Examples on another EP Page and both function correctly.

  Multiple applications deployed in Pure-JavaScript can be viewed inside multiple EP Portlets (IFRAMES). There can be many portlets (IFRAMES) on a single EP page.
- Exiting from an application deployed with Appeon Xcelerator or Pure-JavaScript does not close Internet Explorer.
- Only one response window is allowed for an Internet Explorer browser at a given time.

### 8.4.3 Tasks required to load the application in Enterprise Portal

You need to perform the following four tasks to get a deployed application loaded in the portal after the application migration:

Task #1: Add the application into a portlet. Create an HTML element in the portlet and assign the URL of the Appeon-deployed application to the HTML element.

Task #2: Add the application into a page. Create a new page, add the application into it, and approve the application.

Task #3: Add the page into a page group. Create a new page group and add the page created in task #2 into the page group.

Task #4: Create accounts for viewing the application. Create accounts and assign rights to the accounts for running the application.

For step-to-step instructions on conducting each of the preceding tasks, refer to the relevant documentation provided by Sybase Enterprise Portal (available at http://www.sybase.com/detail?id=1028453).

## 8.5 Single sign-on

### 8.5.1 Method 1: Using a server component to manage user information

STEP 1 – Create an EAServer shared component (either PowerBuilder component or Java component) for storing user information. This shared component is shared between the Appeon application and other applications such as COBOL apps, Web Services, SOA, etc.

STEP 2 – Add the following logic to the Appeon application: when a user logs in, the user information (user ID and password) is passed to the EAServer shared component.

STEP 3 – When the user accesses another application (COBOL apps, Web Services, SOA, etc), the other application gets the user information from the shared component and authenticates the user, so the user does not need to enter multiple user IDs and passwords.

The same method can be used to enable users to first access a non-Appeon application and then access an Appeon application with single sign-on.

### 8.5.2 Method 2: Applying command line argument

Appeon supports the CommandParm function and the command line argument in the Open event of a PowerBuilder application. These features can be applied for implementing single sign-on.

The command line argument can be passed to an Appeon application in the following way:

http://192.0.1.94:8080/MyTest/index.htm?user=appeon&password=appeon

This attaches the string “index.htm?arguments” to the end of the original application URL (“index.htm” must be included in the string).

If the user wants to launch an Appeon application after logging on to an LDAP based application, the LDAP based application passes the user information via the URL of the Appeon application, and the user starts the Appeon application without further login procedures.

### 8.5.3 Method 3: Passing the session ID only in the command line argument

Instead of directly passing the user information in the command line argument, it is possible to pass the session ID only. A table is created in the database for keeping the session information of the LDAP based application, with a session ID assigned to each session, and the session information containing user information. When the Appeon application is launched with the session ID as its command line argument, the application reads user

information from the database table and authenticates the user. Again, the user starts the Appeon application without further login procedures.

## 8.6 Integrating Appeon Web applications with JSP/ASP

### 8.6.1 Applying Appeon CommandParm and Hyperlink features

#### 8.6.1.a Two-way communication (Appeon ←→ JSP/ASP)

ASP or JSP Web pages can be accessed by an example URL http://www.x.x/index.asp?aid=x&bid=y&cid=z and Appeon Web applications can be accessed by an example URL http://192.168.168.18:8080/sales_application_demo/index.html?id1=a&id2=b&id3=c.

You can include parameters in the URLs and therefore pass them between Appeon and JSP/ASP applications.

**Passing parameters from a JSP/ASP application to Appeon application**

Apply the JSP/ASP programming method to simply add the parameters desired to the URL of the Appeon application. Based upon the parameters, any functionality can be built in the Appeon web application, such as opening any window desired and retrieving any data desired to provide client-side integration.

The Appeon application will receive the parameters using the CommandParm and CommandLine functions. The Appeon application will be refreshed while receiving the parameter.

**Passing parameters from an Appeon application to JSP/ASP application**

Appeon supports PictureHyperLink and StaticHyperLink window controls. For example, the developer can statically or dynamically assign the URL of the JSP/ASP Web application to http://www.x.x/index.asp?aid=x&bid=y&cid=z in the Clicked event of a PictureHyperLink or StaticHyperLink window control, and send the parameter from the Appeon application to a JSP/ASP application.

Appeon also supports the HyperLinkToURL PowerScript function. The developers can also apply this function to pass the parameter to a JSP/ASP application through automatic code rather than being user-initiated. The Web application needs to be refreshed while receiving the parameter.

#### 8.6.1.b Using Internet Explorer frame

Appeon applications can be accessed in an Internet Explorer frame. You can set the Appeon application and the ASP/JSP application in two different frames of the same browser.

When a user tries to access an Appeon application with Internet Explorer from an ASP/JSP application in another frame,) the Appeon application directly opens in the frame instead of opening up in a new browser. When the user tries to access the ASP/JSP app, the app opens in the frame as well.

### 8.6.2 Integration through intermediate n-Tier server

- The information can be stored and read by the Client PC's operating system through the signed and secure Appeon ActiveX plug-in.

Both applications write and read a normal Client PC operating system file. The JSP/ASP application needs the ability to access the Client PC operating system file through ActiveX or a Plug-in, etc.

- The intermediate information can be stored in a database table.

  Both applications can read and write a normal RDBMS database table. Information such as orders, products, customers, or loans can be stored in a database table. After the information is stored in the database table, other applications can trigger a user-initiated event or simple automatic timer event to get updated information.

- The intermediate information can be stored in a file on the application server.

  The Appeon application uses an n-Tier NVO for accessing the data stored on the application server. JSP/ASP has its own set of file functions for working on an existing file. The application calls to the fileread and filewrite functions in the n-Tier NVO for reading and writing to the file. Immediately after being stored in the Application Server, the information can be read by both of the Web applications. Updates can be shown immediately and the integration is easily performed using either user-initiated events or a simple automatic timer event refresh.

- The information can be stored and read on the Client PC operating system DLL through the signed, secure Appeon ActiveX.

  Appeon calls to a Client PC operating system DLL file in the same way as in PowerBuilder using the Appeon ActiveX:

```
Var objForm
set objForm=Server.CreateObject("Scripting.Dictionary")
set tStream = Server.CreateObject("adodb.stream")
```

  If both the Appeon and JSP/ASP applications call to the same DLL, the developer can make use of functionality provided by DLLs for setting up communication between the applications.

# 9 Web Application Debugging

## 9.1 Overview

When a PowerBuilder application is deployed to the Web, the Web application may not function properly, even though the same application deployed to a Client Server environment has no problems. These issues are generally caused by unsupported features that are contained in your application or erroneous configuration of the environment such as database connection configuration.

There are four possible ways for you to debug an application:

- Debug according to the execution information recorded in Appeon Developer Web Debug report
- Refer to the *Appeon Troubleshooting Guide*
- Study the log files generated by Appeon Server
- Analyze the Web error and pinpoint the error source based on the analysis

## 9.2 Web Debug report

Appeon provides the Web Debug Report to record execution information during Web application's runtime. You can apply the Web Debug report to easily locate problematic code in the Web files, and fix the errors by rewriting the code.

You can enable or disable the report generation option on the Web Debug Report Configuration window in Appeon Developer. The Web Debug report provides sufficient detailed execution information for debugging, and also provides information on error messages, unsupported features, variables, arguments, etc. For detailed instructions, please refer to Chapter 8: *Generating and Using Runtime Reports* in the *Appeon Developer User Guide*.

## 9.3 Issues recorded in the Appeon Troubleshooting Guide

The Appeon Troubleshooting Guide provides a library of issues and solutions covering topics such as product installation, Web deployment, AEM, and Web application runtime. If you see an error message while working with Appeon, search for the error message in the Appeon Troubleshooting Guide to get a satisfactory solution for the error in the document.

## 9.4 Studying Appeon Server log files

The following log files are generated during the running of Appeon Web applications:

- %JAGUAR%\appeon\log\LogSystem.log
- %JAGUAR%\appeon\log\AppeonServer.log
- %JAGUAR%\bin\Jaguar.log
- %JAGUAR%\bin\Jaguarhttperror.log
- %JAGUAR%\bin\Jaguarhttpservlet.log

If an error occurs, set the log mode to debug in the *AEM* | *Server Properties* | *Web* page. Appeon Server log files generated in debug mode record every system activity in detail and can provide information for troubleshooting obscure or hard to find issues.

## 9.5 Locating and correcting error source

Appeon suggests you debug your Web application through analyzing the Web errors and finding the error source via the corresponding PowerBuilder application. This method of debugging is based on the premise that the Appeon deployed Web application has the same application execution order and similar application structure as the corresponding PowerBuilder application.

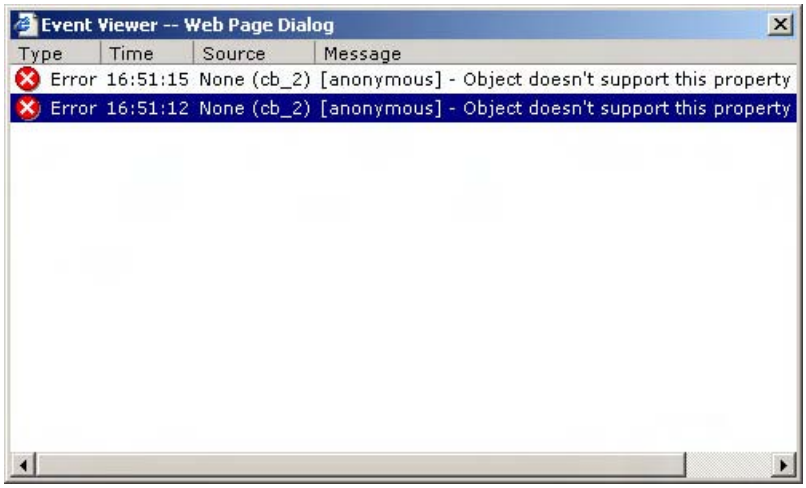
The following sections introduce outline the steps needed to debug an Appeon Web application.

### 9.5.1 Analyzing Web errors

When a Web error occurs, it is usually represented in different ways. An Event Viewer window may appear, or Internet Explorer may report the error.

An *Event Viewer* Web page dialog is displayed, as shown in Figure 9-1, detailing the error location. These errors are reported by the Appeon Server Web Component (formerly called Appeon Web Library) that has its own exception-handling mechanism.

**Figure 9-1: Event Viewer Web dialog**




Internet Explorer may display errors in the Internet Explorer status bar or ask if you want to debug the application using the Microsoft Development Environment, as shown in Figure 9-2.

**Figure 9-2: Error reported by Internet Explorer**


The Web application may stop responding, or it may be terminated without an error message being displayed.

Regardless of how errors are represented, it is possible to judge where the errors may exist according to when the errors occur. Take an MDI Web application as an example. If you run the Web application and you get a blank browser screen with error messages in the Internet Explorer status bar, the error may exist somewhere in the start of the Web application.

If the *Event Viewer* dialog box pops up after a sheet is opened by clicking a menu item and before that sheet's DataWindow displays data, the error may lie in the beginning of the opening of the sheet window to the place where data is retrieved in the sheet window.

Understanding the scope of the Web errors helps to correctly define a scope for debugging the corresponding PowerBuilder application.

### 9.5.2 Defining debugging scope

To define the debugging scope, map the scope of the Web errors to the scope in the corresponding PowerBuilder application that is to be debugged. Here are two examples.

**Example 1**

Web error: this occurs when an Appeon MDI Web application runs, and you get a blank browser screen with error messages in the Internet Explorer status bar.

Web error scope: the starting point of the Web application.

PowerBuilder script debugging scope: From the first line of code in the Open event of the Application object to the line of code that uses the Open function to open the MDI window.

Note: This Web error may be caused by another known reason. For example, the MDI Menu may contain invalid Menu names like "m-.." (the dash) this will cause a Web error.

**Example 2**

Web error: the *Event Viewer* dialog box pops up after a sheet is opened by clicking a menu item and before that sheet's DataWindow displays any data.

Web error scope: from the beginning of opening the sheet window to the place where data is retrieved in the sheet window.

PowerBuilder script debugging scope: suppose that the DataWindow Retrieve() function is written in the Open event of the sheet window, and that before the line of Retrieve() the application calls an NVO that gets the current user profile for the data retrieval. Here the debugging scope should be from the first line in the Open event of the sheet window to the Retrieve() function in the Open event of the sheet window, and should include the script in the NVO that gets the current user profile for data retrieval

When you want to debug a Web error, it is very important to know the debug scope (where the problem may occur and where it ends). You should also clearly know the logic in this scope: how events are triggered in sequence, what functions are called, and what objects are referenced etc.

### 9.5.3 Pinpointing error source

Once the debugging scope has been settled, gradually reduce the scope until you can pinpoint the line(s) of PowerBuilder code that cause the errors on the Web.

STEP 1 – Carefully understand the script logic in the scope to be debugged.

STEP 2 – Use the following methods.

Remove or work around some unsupported features in the scope. *Appeon Help* provides detailed information on supported and unsupported features.

This method is especially useful in handling the problem described in Example 1 in the previous section. The problem is that the entry point of the Web application fails to function, causing the whole application to terminate. Correspondingly, for a typical PowerBuilder application, its entry point usually starts from the first line in the Open event of the Application object to the place where the first window (a login window) is opened. This scope may contain many Appeon unsupported features.

Appeon strongly recommends that you work around the functionality carried out in the application entry point and make it as simple as possible. This functionality may include some data initialization, database connection, or user verification actions. Minimize the possibility that functionality is inaccessible only because of the failure of the application's entry point.

Note: You can use the MessageBox PowerScript function to display debugging information at proper places within the current debugging scope.

In addition, check the Developer Web Report option so that critical information on a Web application's execution will be logged.

Add one or more MessageBox functions where you think the erroneous code exists, and the message boxes should display information on code execution, such as the values of critical variables.

STEP 3 – Perform an incremental deployment of the PowerBuilder application.

STEP 4 – Run the deployed Web application and see if the Web errors disappear. If the Web errors still exist, you should be able to reduce the debugging scope by analyzing the information displayed in the Web message boxes.

STEP 5 **–** Repeat steps 2-4 until the Web errors disappear, or until you can be sure about finding the erroneous code that causes the Web errors.

### 9.5.4 Modifying problematic code

Problematic PowerBuilder code is code that is syntactically valid, but causes Web errors after the Appeon translation. The code is problematic for one of two reasons:

- The code may use features that are not supported by Appeon.

  Solution: remove or work around the unsupported features using *Appeon Help*.

  Note: Some unsupported features are not detected by the Unsupported Features Analysis.

- The code results from an Appeon software or documentation defect. Please report this to Sybase technical support.

  Solution: Remove it, or work around the code in another syntax to avoid the Appeon product bug. Log the bug case and have it sent to Technical Support in order to let Appeon fix the bug as soon as possible.

# Index

**T**



**U**



**W**