Business Objects

BusinessObjects Enterprise XI

Architecture Revealed

Overview

This document discusses the workflow of the components within the BusinessObjects Enterprise XI architecture. Understanding the processes that occur within Enterprise XI will help you increase the resilience, performance and usability of your deployment. The concepts discussed will assist your ability to configure and better maintain an Enterprise XI implementation.

Contents

Introduction

BusinessObjects Enterprise XI is an information delivery system. As report developers create their reports, they publish them into Enterprise XI. It can deliver reports through web browsers, e-mail, mobile phones and more. Users using a web browser can click a hyperlink to the report they need and the report appears in their browser. The users do not need a thick client (executable) on their local machine. They do not need a database client or an understanding of the backend database structure. The backend servers do all the work of getting the data from the database and pushing the report down to the user.

Enterprise XI can integrate into any Internet, intranet or extranet application. Customization features, full support for standard scripting languages (JavaScript and VBScript) and application server integration (Java, .Net, XML) allow developers to build tailored web applications.

Enterprise XI can host interactive analytic content over the web, offering users total flexibility in navigating highly summarized, business-focused views of current and historical data.

For the Enterprise XI administrator, it is important to understand the workflow and the framework. This understanding enables the administrator to maintain and troubleshoot issues that may occur in Enterprise XI architecture.

This document can assist Enterprise XI administrators by revealing the relationship between the various system components. Enterprise XI is a complex system where components interact with each other, requests are received from users, and data gets returned from the databases.

An understanding of these interactions and requests will help in diagnosing where trouble may occur and in resolving the issues effectively.

The Enterprise XI Framework

Enterprise XI provides a framework for information delivery. The information can be in any form, from any place on the intranet or Internet, without causing compatibility problems. The Enterprise XI Framework has an open architecture that supports any kind of information entity. In Enterprise XI, information entities are called InfoObjects and are stored in the Central Management Server (CMS) memory space, also known as the Infostore.

The Enterprise Framework is intended to be seamless to the Enterprise XI administrator. In the Central Configuration Manager (CCM), the CMS has an option in its properties to determine a port number. This port information will set the listening port of the CMS service and affect how the servers talk to one another.

NOTE	No two server types should ever share the same port – be careful when assigning port numbers!

eBusiness Framework and CORBA

The Enterprise XI architecture is made possible by the eBusiness Framework. This framework is made up of a collection of services, which provide a series of Business Intelligence-related functions implemented by one or more Enterprise XI servers. The eBusiness Framework is the CORBA (Common Object Request Broker Architecture) communication layer that enables Enterprise XI components to interact with each other.

In order to do this, each of these servers has the Enterprise Framework dynamic link library (DLL) files added to the operating system during the installation. When you install any Enterprise XI Servers or client tools, the necessary eBusiness Framework CORBA libraries are installed automatically.

These libraries include Business Objects implementation of CORBA, an integration of the facilities of Enterprise XI (Security, Deployment, Administration) with the CORBA 2 Open Standard services (Naming, Trading, Event).

The Enterprise XI Framework is responsible for communication between all Enterprise XI components. It is not a protocol; CORBA uses TCP/IP to communicate. CORBA is a programming framework that Business Objects developers have implemented for communication between Enterprise XI components. This CORBA-based framework makes it easier for the developers to create Enterprise XI.

The eBusiness Framework is an open framework that enables various components to plug in to Enterprise XI:

- Crystal Reports (opening or saving reports from/to Enterprise XI)
- OLAP Intelligence (opening or saving reports from/to Enterprise XI)
- Publishing Wizard (adding reports to Enterprise XI)
- Import Wizard (importing objects from previous and current versions of Enterprise XI)
- Business View Manager (opening or saving Business Views from/to the System database)
- Universe Designer (exporting universes to Enterprise XI)
- CCM (enabling or disabling servers)

The Enterprise XI CORBA implementation does not conflict with other CORBA software on the machine. Each CORBA implementation is statically compiled into that vendor's software and is not available to be shared with other applications. This means that many CORBA applications can be installed on one server without conflicts between CORBA specifications. This is a design feature of CORBA.

NOTE	For more information on the CORBA technology, please refer to the following link: http://www.cs.indiana.edu/~kksiazek/tuto.html

Name Service and the CMS

All CORBA software requires a Name Server. The Name Server service is really a facility of the underlying CORBA architecture and is the tie that loosely binds the servers. It provides a directory of the servers registered in the Enterprise XI environment and helps establish connections between clients and these services. The Name Server service in Enterprise XI is provided by the CMS.

When any Enterprise XI server starts it registers itself with the Name Server. All Enterprise XI servers have a command line parameter of "-ns [CMSNAME]". This is how that server knows where the name server is. The only requirement on a network level is that the server starting will use a network name resolution to locate the name of the CMS. The server that is starting will tell the Name Server information about itself. Examples of the information it shares are its IP address, TCP port information and a description of the service it provides. An example of a service a server would provide is the Cache Server.

As servers in the Enterprise XI environment need to communicate with another server in the Enterprise XI environment, they will make contact with the Name Server service of the CMS. The CMS will respond back with the information on how to communicate with that server. For example, if the Page Server needs to talk to the Cache Server, it might ask the CMS for a list of registered Cache Servers. The Page Server

would then take that information and find a Cache Server to pass the work it needs it to do.

The CMS does not commit its name server information to a database or a file. The server information resides purely in the memory of the CMS. When the CMS stops, it loses all of this information. It is up to each of the servers in the Enterprise XI environment to register with, and connect to, the CMS on a regular basis to make sure the Name Server is still available and up to date. Generally, each server does this once a minute. If at some time a server finds the CMS to be unavailable, it will try to reconnect to the CMS immediately and every minute or so thereafter until the CMS becomes available again. Once the CMS is back up, the server re-registers with the CMS and again sends all pertinent information.

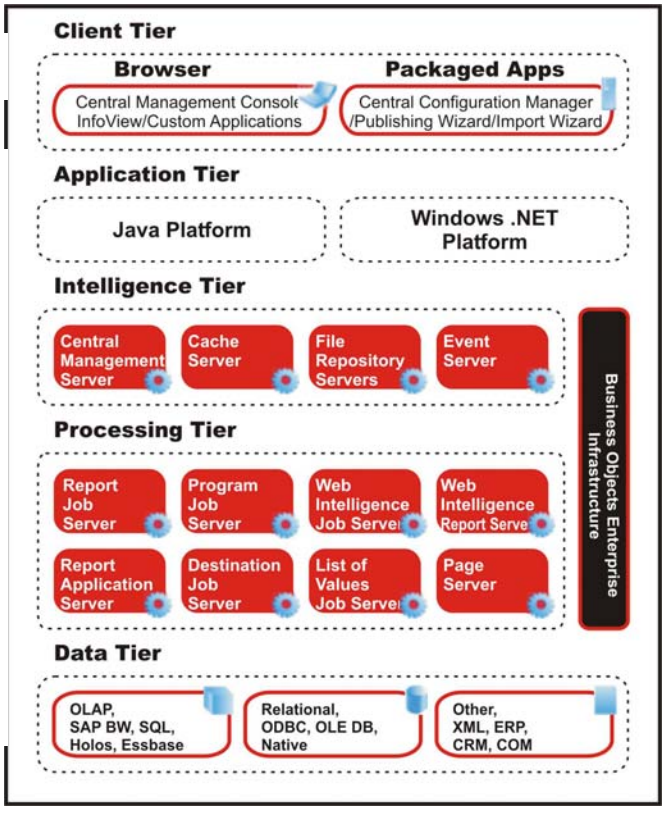
Tiers

Enterprise XI is a multi-tier system. Although the components are responsible for different tasks, they can be logically grouped based on the type of work they perform.

In Enterprise XI, there are five tiers:

- The Client Tier
- The Application Tier
- The Intelligence Tier
- The Processing Tier
- The Data Tier

The following diagram illustrates how each of the components fits within the multi-tier system.



The diagram also illustrates the relationship between the Java and .NET SDK, the Enterprise XI Framework, and the Enterprise XI servers. The Enterprise XI servers run as individual services on Windows machines while on UNIX the servers run as daemons.

To provide flexibility, the components that make up each of these tiers can be installed on one machine, or spread across many. Each tier is a layer of the system that is responsible for a role in the overall architecture. There are many servers involved in this architecture. Some of these servers are only responsible for doing work, others only responsible for managing the servers doing the work.

The services can be vertically scaled to take full advantage of the hardware that they are running on, and they can be horizontally scaled to take advantage of multiple computers over a network environment. This means that the services can all run on the same machine, or they can run on separate machines. The same service can also run in multiple instances on a single machine.

For example, you can run the CMS and the Event Server on one machine, while you run the Page Server on a separate machine. This is called horizontal scaling. If the Page Server is running on a multi-processor computer, then you may choose to run multiple Page Servers on it. This is called vertical scaling. The important thing to understand is that even though these are called servers, they are actually services and daemons that do not need to run on separate computers.

Client Tier

The client tier is the only part of the Enterprise XI system that administrators and end users interact with directly. This tier is made up of the applications that enable users to administer, publish, and view reports and other objects.

Central Management Console

The Central Management Console (CMC) allows you to perform user management tasks such as setting up authentication and adding users and groups. It also allows you to publish, organize, and set security levels for all of your Enterprise XI content.

Additionally, the CMC enables you to manage servers, create server groups, monitor system metrics and control authentication and licensing. Because the CMC is a web-based application, you can perform all of these administrative tasks remotely. The CMC also serves as a demonstration of the ways in which you can use the administrative objects and libraries in the Enterprise XI SDK to create custom web applications for administering Enterprise XI.

InfoView

Enterprise XI comes with InfoView, a web-based interface that users access to view, export, print, schedule and track published reports. InfoView is the main user interface for working with reports through Enterprise XI.

The web client (InfoView) makes a request to the web server, which forwards the user request directly to an application server (on the application tier) where the request is processed by components built on the Enterprise XI Software Development Kit (SDK) – either Java or .NET.

Recognized users of Enterprise XI can customize a personalized version of InfoView. It also serves as a demonstration of the ways to use the Enterprise XI SDK to create a custom web application for end users.

Enterprise XI supports the viewing, printing, and exporting of reports without the need for installing Crystal Reports on the local machine. Report viewing is supported through different viewers compatible with the features of ActiveX, Java, and DHTML.

Custom Applications

Dashboard Manager, which provides the functionality to create dashboards, can be accessed from InfoView. A dashboard contains user-defined settings and can include web sites and objects, such as reports or documents. One or more dashboards can be created and displayed as needed.

For example, you can create a dashboard that contains web sites, Crystal reports or Web Intelligence documents that you frequently access. To view the dashboard, you can either make the dashboard your default view, or you can click its link in the navigation panel. The default name for a dashboard is My InfoView, and its default location is your Favorites folder.

Central Configuration Manager

The CCM is a server-management tool that allows you to configure each of your Enterprise XI server components. This tool can start, stop, enable, and disable servers. It allows you to view and to configure advanced server settings. On Windows, these settings include default port numbers, CMS database and clustering details, SOCKS server connections and more. On Windows, the CCM allows you to add or remove servers from your Enterprise XI system. On UNIX, some of these functions are performed using scripts and other tools.

Publishing Wizard

The Publishing Wizard is a locally-installed Windows application that enables both administrators and users to add reports to Enterprise XI. By assigning object rights to Enterprise XI folders, you control who can

publish reports and where they can publish them. The Publishing Wizard publishes reports from a Windows machine to Enterprise XI servers running on Windows or UNIX.

NOTE	Enterprise XI supports reports created in versions 6 through 10 of Crystal Reports. Once published to Enterprise, reports are saved, processed, and displayed in version XI (11) format.

Import Wizard

The Import Wizard is a locally-installed Windows application that guides administrators through the process of importing users, groups, reports, and folders from an existing Enterprise XI implementation. The Import Wizard runs on Windows, but you can use it to import information into a new Enterprise XI system running on Windows or on UNIX.

Application Tier

The application tier hosts the server-side components that are needed to process requests from the client tier as well as the components that are needed to communicate these requests to the appropriate server in the intelligence tier. The application tier includes support for report viewing and logic to understand and direct web requests to the appropriate Enterprise XI server in the intelligence tier. The components included in your application tier will vary because Enterprise XI is designed to support a variety of web development platforms.

J2EE and .NET Services

Enterprise XI provides integration with Java and Microsoft-based platforms through native J2EE, Microsoft .NET, and Web Services SDKs. These kits are made up of robust components, sample applications, and documentation. Developers install these components on web application platforms including BEA WebLogic, IBM WebSphere, Apache, Oracle 10g Application Server, Sun One, or Microsoft IIS. The SDKs provide a high-level API to control every aspect of Enterprise XI using the developer language of choice.

The application tier acts as the translation layer between the user and the Business Intelligence (BI) platform. The components process requests from the users in the client tier and then communicate these requests to the appropriate service in the intelligence tier. The application tier includes support for document viewing, scheduling, and logic to understand and direct web requests to the appropriate Enterprise XI component.

Enterprise XI systems use the Java SDK or the .NET SDK to run the system with a third-party application server. The application server acts as the gateway between the web server and the rest of the Enterprise XI components. The application server is responsible for processing requests from your browser, sending certain requests to the Web

Component Adapter (WCA), and using the SDK to interpret components in Java Server Pages (JSP) or in Active Server pages (ASP).

Enterprise XI continues to support Crystal Server Pages (CSP) for legacy system support. However, developers are encouraged to use industry standard JSP and ASP whenever possible when building web applications.

Java platform

Enterprise XI systems that use the Enterprise XI Java SDK run the SDK on a third-party web application server such as Apache TomCat or WebSphere. The web application server acts as the gateway between the web server and the rest of the components in Enterprise XI. The web application server is responsible for processing requests from your browser, through the WCA, and using the Java SDK to interpret components in JSP files. The web application server also supports Java versions of the Enterprise XI InfoView, other Enterprise XI applications and uses the SDK to convert report pages (.epf files) to HTML format when users view pages with a DHTML viewer.

Windows .NET platform

Enterprise XI installations that use the .NET Framework include Primary Interop Assemblies (PIAs) that allow you to use the COM Enterprise XI SDK with ASP.NET. Enterprise XI also includes a set of .NET Server Components that you can optionally use to simplify the development of custom applications. This configuration requires the use of a Microsoft IIS web server.

A Web Connector, Web Component Server (WCS), or a WCA is not needed for custom ASP.NET applications.

Web application environments

Enterprise XI supports ASP, CSP, and JSP files. Enterprise XI includes web applications developed in CSP and JSP such as the Enterprise XI Web Desktop/InfoView and the sample applications available for Enterprise XI Launchpads. It also supports the development of custom web applications that use ASP, CSP, JSP, and ASP.NET pages. CSP files provide functionality similar to that provided by Microsoft's ASP files. JSP files allow you to develop cross-platform J2EE applications that use Enterprise XI objects in conjunction with your own custom objects, or a wide variety of objects from third parties.

Enterprise XI includes PIAs that enable you to use the Enterprise XI SDK and the Report Application Server (RAS) SDK with ASP.NET. It also includes a set of .NET Server Components, which simplify development of custom Enterprise XI applications in ASP.NET.

Web Services

Enterprise XI includes a comprehensive Web Services SDK that allows developers to integrate documents directly into applications using industry-standard technology. It consists of a series of web-based functions that use .NET or J2EE platforms and developer environments.

Web Services also make it easier and faster to integrate Business Objects technology with other web-based applications, and facilitate the deployment of Business Objects with customized applications. Web Services are available for document display, refresh, and providing drill-down functionality to users. For developers, the Web Services provider is deployed on the server side with an Enterprise XI server. The API enables the creation of customized web sites, applications, or web services that access the Enterprise XI services.

The web application server also supports the InfoView portal and uses the SDKs to convert documents managed by Enterprise XI into HTML format when users view pages with a DHTML viewer.

Web Component Adapter

Enterprise XI provides a web application, the WCA, that allows your web application server to run Enterprise XI applications and to host the CMC. The web server communicates directly with the web application server that hosts the Enterprise XI SDK. The WCA runs on the web application server and provides all services that are not directly supported by the Enterprise XI SDK. The web server passes requests directly to the web application server, which then forwards the requests on to the WCA. The WCA supports the CMC and OLAP Intelligence document viewing and interaction.

There are two versions of the WCA:

- .NET - The .NET WCA must be installed on an IIS web application server.
- Java - The Java WCA must be installed on a J2EE web application server.

Enterprise XI supports a number of third-party application servers, so you can connect it to your existing infrastructure.

NOTE	The Web Connector that resided on the web server in earlier versions of Crystal Enterprise is no longer required. Requests are now handled by the web application server, and are passed on to the WCA. This solution also replaces the WCS.

Intelligence Tier

The Intelligence tier manages the Enterprise XI system. It maintains all of the security information, sends requests to the appropriate servers, manages audit information, and stores report instances.

The Intelligence tier of Enterprise XI consists of five servers:

- Central Management Server (CMS)

- Cache Server
- Input and Output File Repository Servers (FRS)
- Event Server

These servers will be discussed in more detail in the Enterprise XI Components section to follow.

Processing Tier

The Processing tier accesses the data and generates the reports.

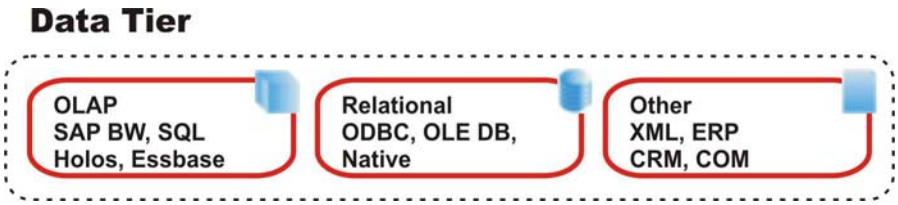
The Processing tier of Enterprise XI consists of eight servers:

- Report Job Server
- Program Job Server
- Web Intelligence Job Server
- Web Intelligence Report Server
- Report Application Server (RAS)
- Destination Job Server
- List Of Values Job Server (LOV)
- Page Server

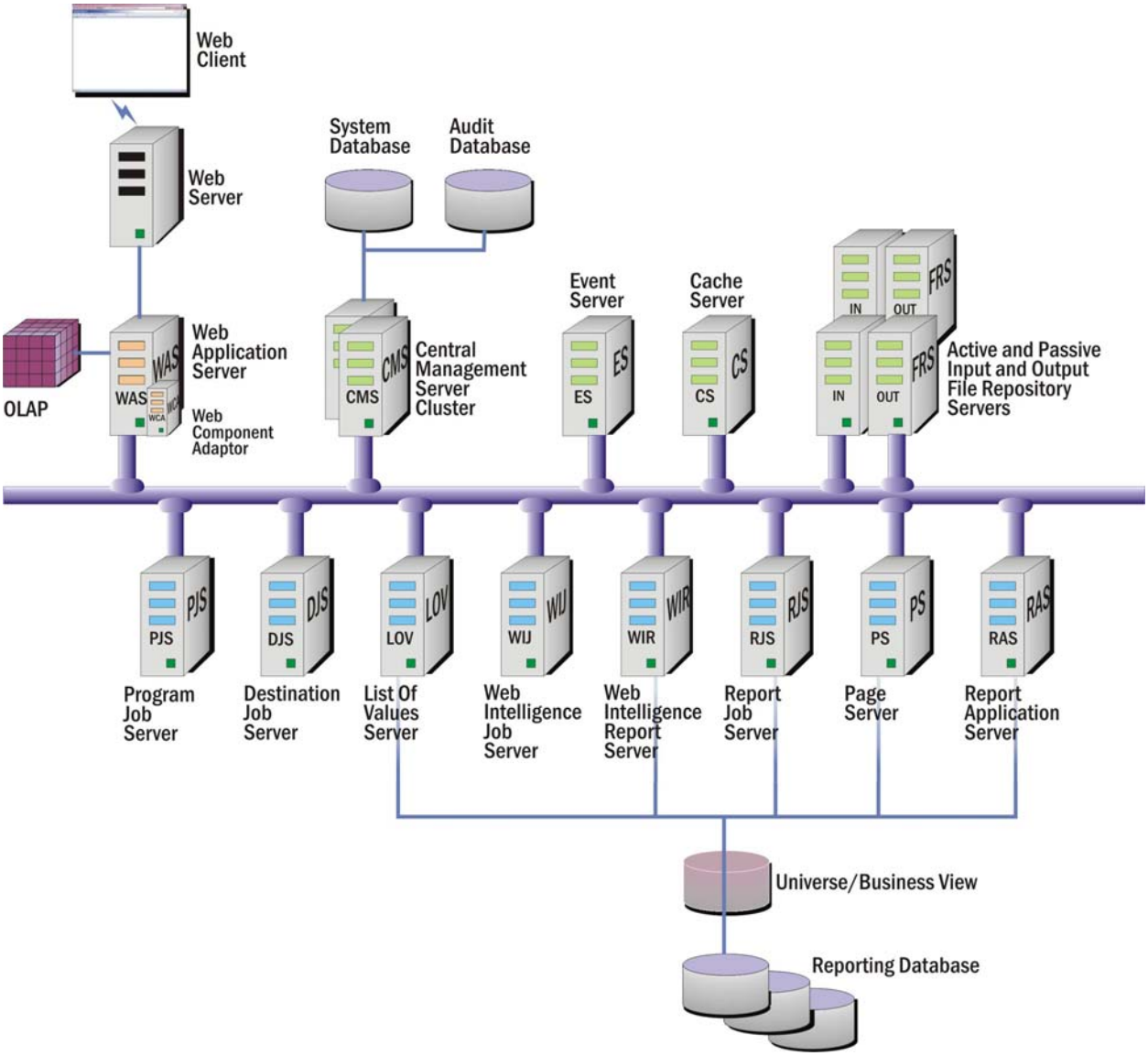
These servers will be discussed in more detail in the Enterprise XI components section to follow.

Data Tier

The data tier is made up of the databases that contain the data used in the reports. Enterprise XI supports a wide range of corporate databases.



Enterprise Framework Diagram



Components

Web Client

A web browser is a software application that enables a user to display and interact with HTML documents hosted by web servers or held in a file system. A web client is hosted within the web browser and presents the user with a specific gateway to the main application.
The web clients, InfoView or the CMC make requests to the web server, which forwards the user requests directly to a web application server where the requests are processed by components built on the Enterprise XI SDK – either Java or .NET.

Web Server

The web server manages communication between the web browser and the Enterprise XI SDKs.

The primary responsibility of the web services is to receive and to interpret user requests from user interfaces such as the InfoView or the CMC. The web services then contact other Enterprise XI servers to determine the response to the request and format the response so that it can be returned to the web client.

When using the DHTML viewer or Advanced DHTML viewer, the web services convert Encapsulated Page Format (EPF) files to DHTML. Report viewing in Enterprise XI uses Page On Demand technology that sends each page of a report as it is requested by the user. The EPF files hold the individual report pages.

When viewing an OLAP Intelligence report using the DHTML OLAP Intelligence viewer, the web services connect to the OLAP data source to retrieve the views of data required for the report. When viewing an OLAP Intelligence report using the ActiveX OLAP Intelligence viewer, the web client makes a direct connection to the OLAP data source to retrieve the views of data.

NOTE	In Crystal Enterprise 10 on Windows, the communication between the web server and the web application server was handled through the Web Connector; the functionality of the WCA was provided through the WCS. In Enterprise XI, the web server communicates directly with the application server and the WCA handles the WCS functionality, both on Windows and UNIX platforms.

Web Application Server

Enterprise XI systems that use the Enterprise XI Java SDK or the Enterprise XI .NET SDK run on a third-party web application server. Read the updated Platforms file at the Supported Platforms web site for a complete list of tested web application servers and version requirements. The platforms.txt included with your product distribution is less current than the web site.

The web application server acts as the gateway between the web server and the rest of the components in Enterprise XI. The web application server is responsible for processing requests from your browser. It also supports InfoView and other Business Objects applications, and uses the SDK to convert EPF files to HTML format when users view pages with a DHTML viewer.

NOTE	In the period since CSP was developed, the cross-platform application server market has become dominated by the J2EE platform and Microsoft's ASP.NET. As a result, Business Objects has moved away from the proprietary CSP language and has instead developed tools and applications in ASP.NET and J2EE.

Web Component Adapter

The web server communicates directly with the web application server that hosts the Enterprise XI SDK. The WCA runs within the web application server and provides all services that are not directly supported by the Enterprise XI SDK. The web server passes requests directly to the web application server, which then forwards the requests on to the WCA.

The WCA has two primary roles:

- It processes ASP.NET (.aspx) and JSP files.
- It also supports Business Objects applications such as the CMC and Crystal Reports viewers that are implemented through viewrpt.aspx requests.

The Java Web Component Adapter

The Java WCA is a Java web application that runs on your Java web application server. The WCA hosts web components, including a CSP plug-in that allows you to run CSP applications on your Java web application server. You must deploy the WCA to run the CMC, and other CSP applications in a Java Enterprise XI environment.

The Enterprise XI setup program configures the Web archive file that implements the WCA webcompadapter.war file with the following information:

- The name and location of your CMS.
- The default display name of the WCA.
- The location of the directories where the WCA can find CSP applications.
- The location it should use for log files.

To deploy the WCA, you must first configure your web application server to use the cewcanative.jar file, typically by adding its path to the CLASSPATH of the web application server. Then you must deploy the WCA archive webcompadapter.war file as a web application.

The .NET Web Component Adapter

The .NET WCA is a web application that runs on your web application server. The WCA hosts web components, including a CSP plug-in that allows you to run CSP applications. You must deploy the WCA to run the CMC, and other CSP applications, in a .NET Enterprise XI environment.

CSP-based application hosting is now handled by your Java or IIS web application server. If you want to use existing CSP-based applications with Enterprise XI, you need to configure them to work with the WCA.

NOTE	If your default viewer is the Advanced DHTML Viewer, the report is processed by the RAS.

Central Management Server

The CMS maintains a database of information that allows you to manage the Enterprise XI Framework. The data stored by the CMS includes information about users and groups, security levels, content, and servers. The CMS also maintains the Repository, and a separate audit database of information about user actions.

The CMS has four main functions:

- Maintaining security

  By maintaining a database of users and their associated object rights, the CMS enforces who has access to Enterprise XI and the types of tasks they are able to perform. This also includes enforcing and maintaining the licensing policy of your Enterprise XI system.

- Managing objects

  The CMS keeps track of the location of objects and maintains the folder hierarchy. By communicating with the Report and Program Job Servers, the CMS is able to ensure that scheduled jobs run at the appropriate times.

- Managing servers

  By staying in frequent contact with each of the servers in the system, the CMS is able to maintain a list of server status. The web client (InfoView or CMC) accesses this list, through the SDK, to identify which Cache Server is free to use for a report viewing request.

- Managing auditing

  By collecting information about user actions from each Enterprise XI server, and then writing these records to a central audit database, the CMS acts as the system auditor. This audit information allows system administrators to better manage their deployment.

Typically, you provide the CMS with database connectivity and credentials when you install Enterprise, so the CMS can create its own system database using your organization's preferred database server.

NOTES	
	• It is strongly recommended that you back up the CMS system database and the audit database frequently. The backup procedure depends upon your database software. If you are unsure of the procedure, consult with your database administrator. • The CMS database should not be accessed directly. System information should only be retrieved using the calls that are provided in the Enterprise XI SDK. • You can access the audit database directly to create custom audit reports.

On Windows, the Setup program can install and configure its own Microsoft Data Engine (MSDE) database if necessary. MSDE is a client/server data engine that provides local data storage and is

compatible with Microsoft SQL Server. If you already have the MSDE or SQL Server installed, the installation program uses it to create the CMS system database. You can migrate your default CMS system database to a supported database server later.

Cache Server

The Cache Server is responsible for handling all report viewing requests. The Cache Server checks whether or not it can fulfill the request with a cached report page. If the Cache Server finds a cached page that displays exactly the required data, with data that has been refreshed from the database within the interval that you have specified as the default, then the Cache Server returns that cached report page.

If the Cache Server cannot fulfill the request with a cached report page, it passes the request along to the Page Server. The Page Server runs the report and returns the results to the Cache Server. The Cache Server then caches the report page for future use, and returns the data to the viewer. By storing report pages in a cache, Enterprise XI avoids accessing the database each and every time a report is requested.

If you are running multiple Page Servers for a single Cache Server, the Cache Server automatically balances the processing load across Page Servers.

Page Server

The Page Server is primarily responsible for responding to page requests from the Cache Server by processing reports and generating EPF files. The EPF files contain formatting information that defines the layout of the report. The Page Server retrieves data for the report from an instance or directly from the database (depending on the user's request and the rights he or she has to the report object). When retrieving data from the database, the Page Server automatically disconnects from the database after it fulfills its initial request and reconnects if necessary to retrieve additional data to conserve database licenses.

The Cache Server and Page Server work closely together. Specifically, the Page Server responds to page requests made by the Cache Server. The Page Server and Cache Server also interact to ensure cached EPF files are reused as frequently as possible and new pages are generated as soon as they are required. Enterprise XI takes advantage of this behavior by ensuring that the majority of report viewing requests are made to the Cache Server and Page Server. The Page Server only allows viewing of reports. However, if a user's default viewer is the Advanced DHTML Viewer, the report is processed by the RAS or if you want users to create and modify reports in your web applications, you must use the RAS. The Page Server also supports COM, ASP.NET, and Java viewer SDKs.

The Page Server architecture was redesigned in Enterprise XI for increased reliability. It functions like the Job Server in that it spawns pageserver.exe sub-processes to handle page requests. The Page Server

preloads a pageserver.exe sub-process (often referred to as a child process) on start-up. Each sub-process can handle 10 concurrent jobs and each sub-process loads a copy of the crpe32.dll and other required files.

The Page Server automatically load balances jobs, sends previously requested requests to the same Page Server sub-process and uses the default printer driver for the account running the Page Server for formatting and layout.

You cannot audit the Page Server. All requests go through the Cache Server, so to audit page requests you must enable and configure auditing on the Cache Server.

Report Job Server

A Job Server processes scheduled actions on objects at the request of the CMS. You can configure a Job Server to process reports, programs, Web Intelligence documents, LOV objects or destinations when you add it to your Enterprise XI system.

If you configure a Job Server to process report objects, it becomes a Report Job Server. The Report Job Server processes scheduled reports, as requested by the CMS, and generates report instances (instances are versions of a report object that contain saved data). To generate a report instance, the Report Job Server obtains the report object from the Input FRS and communicates with the database to retrieve the current data. Once it has generated the report instance, it stores the instance on the Output FRS.

The Job Server will spawn a child process for each concurrent report job and it is these child processes, which connect through CORBA to the Input and Output FRS.

The Job Server can be configured to send scheduled reports to their scheduled destinations – including four specific destinations:

- SMTP
- Inbox
- FTP
- Unmanaged Disk

The Job Server can also export the report in various file formats including the following:

- Crystal Reports (RPT)
- Excel
- PDF
- Rich Text Format (RTF)

It is the same JobServer.exe used for all Job Servers. Different –lib and –objType switches are used to identify the type of Job Server.

The Job Server can be audited with settings in the CMC and can audit three types of events:

- A job has been run successfully.
- A job has failed to run.
- A job failed but will try to run again.

Program Job Server

A Job Server processes scheduled actions on objects at the request of the CMS. If you configure a Job Server to process program objects, it becomes a Program Job Server. The Program Job Server processes scheduled program objects, such as Java or Script programs, as requested by the CMS. Program objects allow you to write, publish, and schedule custom applications. This includes scripts or Java programs that run against, and perform maintenance work on, Enterprise XI.

To run a program, the Program Job Server first retrieves the files from storage on the Input FRS, and then runs the program. By definition, program objects are custom applications. Therefore the outcome of running a program will be dependent upon the particular program object that is run. Unlike report instances, which can be viewed in their completed format, program instances exist as records in the object history. Enterprise XI stores the program's standard out and standard error in a text output file. This file appears when you click a program instance in the object history.

Event Server

The Event Server manages file-based events. This includes monitoring for file-based events and notifying the CMS when an event has occurred.

When you set up a file-based event within Enterprise XI, the Event Server monitors the directory that you specified. When the appropriate file appears in the monitored directory, the Event Server triggers your file-based event: that is, the Event Server notifies the CMS that the file-based event has occurred. The CMS then starts any jobs that are dependent upon your file-based event.

After notifying the CMS of the event, the Event Server resets itself and again monitors the directory for the appropriate file. When the file is newly created in the monitored directory, the Event Server again triggers your file-based event.

NOTE	Schedule-based events and custom events are managed by the CMS.

Report Application Server

The RAS processes reports that users view with the Advanced DHTML Viewer. The RAS also provides the ad-hoc reporting capabilities that allow users to create and modify reports over the Web, as well as saving reports to the CMS.

The RAS is very similar to the Page Server; it too is primarily responsible for responding to page requests by processing reports and generating EPF pages. But the RAS uses an internal caching mechanism that involves no interaction with the Cache Server.

As with the Page Server, the RAS supports COM, ASP.NET, and Java viewer SDKs. The RAS also includes an SDK for report creation and modification, providing you with tools for building custom report interaction interfaces.

File Repository Servers

There is an Input FRS and an Output FRS in every Enterprise XI implementation.

The Input FRS manages all of the report objects and program objects that have been published to the system by administrators or users. Objects are published to the system in many ways such as using the Publishing Wizard, CMC, Import Wizard, Crystal Reports, or Web Intelligence.

NOTE	If you use the Enterprise XI SDK, you can also publish reports from within your own code.

The Output FRS stores and manages all of the report and program instances generated by the Report Job Server or the Web Intelligence Report Server, and the program instances generated by the Program Job Server.

The FRS are responsible for listing files on the server, querying for the size of a file, querying for the size of the entire file repository, adding files to the repository, and removing files from the repository.

NOTE	• The Input and Output FRS cannot share the same directories. This is because one of the FRS could then delete files and directories belonging to the other. • In larger deployments, there may be multiple Input and Output FRS, for redundancy. In this case, all Input FRS must share the same directory. Likewise, all Output FRS must share a directory. • Objects with files associated with them, such as text files, Microsoft Word files, or PDFs, are stored on the FRS.

When a Crystal report is added to the system, you have a choice to keep saved data in the report or to have all data removed from the report. The default is to have the data removed from the report. That is why reports in the Input FRS are sometimes referred to as report templates or report definitions.

If a report object with no data is viewed by a user then the report must be refreshed and populated with data in order to create pages. If a report object is added with saved data, when the report object is viewed, the data that is saved in the report will be displayed, and the report data will not be refreshed.

Instances are the results of an object being scheduled. A report object may have many report instances. The two types of objects that can be scheduled are Crystal reports and Program objects. A Crystal Report instance is a report with saved data (as of the time that the report was processed). A Program instance is a text file to which any output of the program is written.

CAUTION	It is important to back up your FRS as your report objects and historical data would be costly or impossible to replace if lost.

Web Intelligence Job Server

The Web Intelligence Job Server processes scheduling requests it receives from the CMS for Web Intelligence documents. It forwards these requests to the Web Intelligence Report Server, which will generate the instance of the Web Intelligence Document (WID), stores the instance in the Output FRS and reports the status of running WID Jobs to the CMS. The Web Intelligence Job Server does not actually generate object instances. When a user requests to view and interact with a stored document instance, the Web Intelligence report server accesses that instance directly.

As with all job servers, it uses the same main executable as the other job servers and manages sub-processes, launching a job server child (jobserverchild.exe) sub-process for each concurrent scheduled job. The Web Intelligence Job Server sends successful instances to their scheduled destinations including Inbox and SMTP.

The Web Intelligence Job Server can be audited with settings in the CMC and can audit three types of events:

- A job has been run successfully.
- A job has failed to run.
- A job failed but will try to run again.

Web Intelligence Report Server

The Web Intelligence Report Server is a new service in Enterprise XI and provides core Web Intelligence display and interaction within the platform for end-user query and analysis. The Web Intelligence Report Server is accessed when the CMS requests the creation or viewing of a Web Intelligence document for further interaction.

The Web Intelligence Report Server is used to create, edit, view, and analyze WID. It also processes scheduled WID and generates new

instances of the document, which it stores on the Output FRS. Depending on the user's access rights and the refresh options of the document, the Web Intelligence Report Server will use cached information, or it will refresh the data in the document and then cache the new information. The caching is done in XML format with each page in a separate directory.

The Web Intelligence Report Server connects to universes and reporting databases and creates XML pages of the documents to send back to the SDK and can export documents to Excel and PDF formats

For users who want to conduct ad-hoc query and analysis, the Web Intelligence Report Server requests a predefined metadata object, called a universe, from the repository and opens an HTML or Java-based query panel. Users can then drag and drop requested fields, filters, or objects onto the Web Intelligence user interface. The Web Intelligence report server handles report viewing, modification, and interaction. This includes advanced on-report analysis functionality now available with Web Intelligence XI. Due to the interactive nature of Web Intelligence, no separate RAS is required.

List of Values Job Server

The LOV Job server is a new component within Enterprise XI. Its purpose is to support the scheduling of predefined LOV or prompts. These lists support dynamic and cascading prompts for Crystal Reports XI. Future versions will support scheduled dynamic and cascading prompts lists for Web Intelligence and other tools.

The key benefit of having predefined and scheduled lists of values is that the report does not have to query the database to gather the prompts every time a user requests a specific view of a report. On-demand LOV that query the database are handled by the RAS. Wherever possible, prompt lists should be scheduled in cases where the cascading prompt levels do not change regularly such as for country, state, city, and customer lists.

The LOV Job Server processes scheduled LOV objects. These are actually RPT files in the Input FRS that contain the values of specific fields in a Business View. LOV are used to implement dynamic prompts and cascading LOV within Crystal Reports. LOV objects do not appear in the CMC or InfoView.

The LOV Job Server behaves similarly to the Report Job Server in that it retrieves the scheduled objects from the Input FRS and saves the instance it generates to the Output FRS. There is never more than one instance of a LOV object. On demand LOV objects are processed by the RAS.

Destination Job Server

When you add a job server to your Enterprise XI system, you can configure it to process report objects or program objects, or to send

objects or instances to specified destinations. If you configure it to send objects or instances, it becomes a Destination Job Server. A Destination Job Server processes requests that it receives from the CMS and sends the requested objects or instances to the specified destination:

- If the request is for an object, it retrieves the object from the Input FRS.
- If the request is for a report or program instance, it retrieves the instance from the Output FRS.

The Destination Job Server can send objects and instances to destinations inside the Enterprise XI system, a user's inbox, or outside the system by sending a file to an email address. The Destination Job Server does not run the actual report or program objects. It only handles objects and instances that already exist in the Input or Output File Repository Servers.

BusinessObjects Enterprise XI workflow

The following sections contain various Enterprise XI workflow descriptions. These sequences will detail the communications between the various components of the Enterprise system. Understanding workflow will allow an administrator to fine tune, troubleshoot and even enhance the performance of an Enterprise XI implementation.

NOTE	The following Workflow sequences may be better understood by viewing • The PowerPoint presentation - Enterprise XI Architecture Revealed, and • The Enterprise XI Framework diagram. The presentation includes animations, which show the process flow and use the same headings as this document.

Client communication

Connecting to the Enterprise XI environment can be done with a thin client – such as InfoView or the CMC through a web browser. Connection can also be made with a thick client application such as the Import Wizard or the Publishing Wizard. This section will review the connectivity processes of the two methods and their relationship to the Enterprise XI Framework.

Web (Thin) clients

To use the Enterprise XI web clients (InfoView or CMC), all that is required is a web browser. When a user enters the URL for the InfoView or CMC the web browser will send that request to the specified server through TCP/IP.

Application (Thick) clients

Thick client applications are the executables that are installed on the local machine. These applications need to connect to Enterprise XI and include the Enterprise Framework DLLs with the application and latch on to the Framework upon connection.

These are the common thick client applications:

- Import Wizard
- Publishing Wizard
- Business View Manager
- Universe Designer
- Crystal Reports
- OLAP Intelligence

They connect to the CMS to verify username and log on, do the work they need to do and disconnect from the system. Developers can easily create a client application that accesses the Enterprise system in this way because the COM objects are intentionally left exposed for developers to use. An example would be VB applications that silently connect to the CMS to trigger a custom event and log off.

Another example of this is OLAP Intelligence. An analytic report has been designed and is ready to be published to the system. The **Save to CMS** option opens a dialog box asking for username, password, authentication type and the CMS to connect to. It then loads the Enterprise Framework DLLs, connects to the Enterprise XI environment through the Framework and retrieves a list of folders. After the user chooses a folder to save the CAR file in, the thick client uploads it to the FRS through the Framework. OLAP Intelligence then closes the connection to the Framework and allows the user to continue using the application.

Workflow - Logging on to Enterprise

1. The web client sends the logon request in a URL typically through the web server to the web application server.
2. The web application server will interpret the JSP or ASP page and the values sent in the URL request and will determine that the request is a logon request. The web application server will send the username, password, and authentication type to the specified CMS for authentication.
3. The CMS will validate the username and password against the appropriate source such as Enterprise, Windows AD, or LDAP. For example, Enterprise authentication would be authenticated against the System database. Upon successful validation, the CMS creates a session for the user in its own memory and a license is used.
4. The CMS sends a response to the web application server to let it know that the validation was successful. The web application server generates a logon token for the user session in its memory. From the rest of this session, the web application server will use the logon token to validate the user against the CMS.

5. The web application server formats the response to send to the client. The web application server sends the response back to the user's machine where it is rendered in the web client.

Publishing a report to Enterprise XI

Crystal Reports is used to create reports for the Enterprise XI system to distribute. Once the designer has built the reports, the reports can then be added to the Enterprise XI system so that they can be distributed and shared. The Publishing Wizard is one of the tools that can be used to add existing reports to the Enterprise XI system. It is a 32-bit executable, rather than a web-based application. In order to use the Publishing Wizard, the user will need to have the application installed locally onto their machine.

Connecting to the CMS

Once you have selected the reports that you wish to publish, it is necessary for the Publishing Wizard to authenticate you against the Enterprise XI system. Authentication is necessary as the Enterprise XI system will ensure that the user publishing reports will only be able to publish reports to the folders that they have appropriate security access to.

NOTE	Users of the Publishing Wizard will only be able to add reports to folders where they have been given Full Control access or the Add objects to the folder advanced right.

When you add a report, the report file is automatically copied to the Enterprise XI environment from the report designer's machine. Administration becomes easier for Enterprise XI as all report objects are in a single location that can easily be secured and backed up on a regular basis.

The Publishing Wizard uses the print engine (crpe32.dll) and opens up each report, one at a time, getting the structure information of the report, such as report title, record selection, parameter, database connection information, and sends it to the CMS.

The report becomes a report template and is written to the Input FRS. The CMS will then update its record of the object in the system database including location and folder information.

If the report information such as record selection, database connectivity or parameter settings needs to be changed, the changes must be made in the actual report, in the designer, and then the report saved back to Enterprise XI.

Viewing a report

This section describes the viewing mechanisms that are implemented in Enterprise XI.

When you view a Crystal report (.rpt file), the processing flow varies depending upon your default report viewer, the type of report, and the rights you have to the report. In all cases, however, the request that begins at the web server must be forwarded to the web application server. The actual request is constructed as a URL that includes the report's unique ID.

This ID is passed as a parameter to a server-side script that, when evaluated by the application server verifies the user's session and retrieves the logon token from the browser. The script then checks the user's Enterprise XI InfoView preferences and redirects the request to the viewing mechanism that corresponds to the user's default viewer.

Different report viewers require different viewing mechanisms:

- The zero-client DHTML viewer is implemented through the viewreport.csp, viewreport.aspx or viewreport.jsp file. When evaluated by the web application server, this script communicates with the framework (through the published SDK interfaces) in order to create a viewer object and retrieve a report source from the Cache Server and Page Server.
- The zero-client Advanced DHTML viewer is implemented through viewreport_ia.csp or viewreport_ia.jsp. When evaluated by the web application server, this script communicates with the framework (through the published SDK interfaces) in order to create a viewer object and retrieve a report source from the RAS.
- The ActiveX and Java client-side report viewers are implemented through viewrpt.cwr, hosted by the web application server. The Crystal Web Request is executed internally through viewer code on the web application server. The viewer code communicates with the framework in order to retrieve a report page in EPF format from the Cache Server and Page Server. If they haven't already done so, users are prompted to download and install the appropriate viewer software.

Report viewing with the Cache Server and Page Server

Upon receiving a report-viewing request, the Cache Server checks to see if it has the requested pages cached. Cached pages are stored as EPF files. If a cached version of the EPF file is available, the Cache Server checks with the CMS to see if the user has rights to view the report. If the user is granted the right to view the report, the Cache Server sends the EPF file to the web application server.

If a cached version of the EPF page is unavailable, the Cache Server requests new EPF pages from the Page Server. The Page Server retrieves the report from the Input FRS, first checking with the CMS to see if the user has rights to view the report.

If the report is an instance, and the user only has View rights, the Page Server will generate pages of the report instance using the data stored in the report object. That is, the Page Server will not retrieve the latest data

from the database. If the report is an object, the user must have View On Demand rights to view the report successfully (because the Page Server needs to retrieve data from the database).

If the user has sufficient rights, the Page Server generates the EPF pages and forwards them to the Cache Server. The Cache Server then caches the EPF files and sends them to the application server.

If the initial request was made through a CSP file (viewreport.csp), the viewer SDK (residing on the web application server) is used to generate HTML that represents both the DHTML viewer and the report itself. The HTML pages are then returned through the web server to the user's web browser. If the initial request was made through a Crystal Web Request (viewrpt.cwr), the web application server forwards the EPF pages through the web server to the report viewer software in the user's web browser.

Report viewing with the Report Application Server

Upon receiving a report-viewing request, the RAS checks to see if it has the requested report data in cache. The RAS has its own caching mechanism, which is separate from the Cache Server. If cached report data is available, the RAS checks with the CMS to see if the user has rights to view the report. If the user is granted the right to view the report, the RAS returns EPF pages to the application server. If a cached version of the page is unavailable, the RAS retrieves the report from the Input File Repository Server, first checking with the CMS to see if the user has rights to view the report. The RAS then processes the report and returns the EPF pages to the application server.

If the user is granted View rights to the report object, then the RAS will only ever generate pages of the latest report instance. That is, the RAS will not retrieve the latest data from the database. If, however, the user is granted View On Demand rights to the report object, then the RAS will refresh the report against the database.

NOTE	The interactive search and filter features provided by the Advanced DHTML viewer are available only if the user has View On Demand rights (or greater) to the report object.

When the application server receives the EPF pages from the RAS, the viewer SDK is used to generate HTML that represents both the Advanced DHTML viewer and the report itself. The HTML pages are then returned through the web server to the user's web browser.

Workflow - Viewing a report on demand

1. The web client sends the schedule request in a URL typically through the web server to the web application server.
2. The web application server will interpret the requested page and the values sent in the URL request. It will determine that it is a request to view the first page of the selected report object.

3. The web application server will send a request to the CMS to ensure that the user has rights to view the object. The CMS checks the System database to verify the user rights.
4. The CMS sends a response to the web application server to confirm the user has sufficient rights to view the object.
5. The web application server sends a request to the Cache Server requesting the first page of the report object.
6. The Cache Server checks to see if the page already exists. Unless the report meets the requirements for On Demand report sharing (within 15 minutes of another On Demand request, same rights, DB logon, parameters), the Cache Server sends a request for the Page Server to generate the page.
7. The Page Server parent process determines whether it should start a new Page Server child to process the report or if there is already one available to process it. The Page Server Child then retrieves the report template from the Input FRS.
8. The Input FRS streams a copy of the instance to the Page Server child. The Page Server child opens the report in its memory and checks to see if the report contains data.
9. Since a report object does not have data, the Page Server child will connect to the database to query for data. The database returns data to the Page Server child. The Page Server child processes the report and then generates the first page of the report. The Page Server holds the report in temp files in memory until it reaches a 60-minute idle time. The temp files are then deleted from memory.
10. The Page Server sends the EPF page to the Cache Server. The Cache Server stores a copy of the EPF page in its cache directory.
11. The Cache Server sends the EPF page to the web application server. If the DHTML Viewer were used, the web application server would convert the EPF to DHTML.
12. The web application server sends the EPF page to the web server. The web server sends the EPF page to the user's machine where it is rendered in the viewer on the web client.

Workflow - Viewing a Web Intelligence report

1. The web client sends the schedule request in a URL typically through the web server to the web application server.
2. The web application server will interpret the requested page and the values sent in the URL request and will determine that it is a request to view a Web Intelligence report.
3. The web application server will send a request to the CMS to ensure that the user has rights to view the object. The CMS checks the System database to verify the user rights.
4. The CMS sends a response to the web application server to confirm the user has sufficient rights to view the object.

5. The web application server sends a request to the Web Intelligence Report Server requesting the report.
6. The Web Intelligence Report Server requests the report from the Input FRS. The Input FRS streams a copy of the report to the Web Intelligence Report Server. The Web Intelligence Report Engine opens the report in its memory. The QT.dll generates the SQL from the Universe that the report is based on.
7. The Web Intelligence Report Server connects to the database to run the query. The query data is passed through QT.dll to the Report Engine where the report is processed.
8. The Web Intelligence Report Server sends the finished report to the web application server.
9. The web application server sends the finished report to the web server. The web server sends the finished report to the user's machine where it is rendered in the web client.

Scheduling a report

When you schedule a report, you instruct Enterprise XI to process a report object at a particular point in time, or on a recurring schedule. For example, if you have a report based off of your web server logs, you can schedule the report to run every night.

NOTE	Enterprise XI also allows you to schedule jobs that are dependent upon other events.

When a user schedules a report using Enterprise XI InfoView, the request is sent to the application tier (web application server) and on through the WCA. Since the request was to schedule a report, the request is passed to the CMS.

When the CMS gets the request, it checks to see if the user has sufficient rights to schedule the report. If the user has sufficient rights, the CMS schedules the report to run at the specified time(s). When it is time, the CMS passes the job to the Report Job Server. The Report Job Server retrieves the report from the Input FRS and runs the report against the database, thereby creating an instance of the report. The Report Job Server then saves the report instance to the Output FRS, and tells the CMS that it has completed the job successfully.

NOTE	• The Cache Server and the Page Server do not participate in scheduling reports or in creating instances of scheduled reports. This can be an important consideration when deciding how to configure Enterprise, especially in large installations. • When you schedule program objects or object packages, the interaction between servers follows the same pattern as it does for reports. However, program objects are processed by the Program Job Server. • Users without schedule rights on an object will not see the schedule option in Enterprise.

Choosing between live and saved data

When reporting over the Web, the choice to use live or saved data is one of the most important decisions you'll make. Whichever choice you make, however, Enterprise XI displays the first page as quickly as possible, so you can see your report while the rest of the data is being processed.

Live data

On-demand reporting gives users real-time access to live data, straight from the database server. Use live data to keep users up-to-date on constantly changing data, so they can access accurate information. For instance, if the managers of a large distribution center need to keep track of inventory shipped on a continual basis, then live reporting is the way to give them the information they need.

Before providing live data for all your reports, however, consider whether or not you want all of your users connecting to the database server on a continual basis. If the data isn't rapidly or constantly changing, then all those requests to the database do little more than increase network traffic and consume server resources. In such cases, you may prefer to schedule reports on a recurrent basis so that users can always view recent data (report instances) without connecting to the database server.

NOTE	Users require View On Demand access rights to refresh reports against the database.

Saved data

Report instances are useful for dealing with data that is not continually updated. When users navigate through report instances, and drill down for details on columns or charts, they do not access the database server directly; instead, they access the saved data. Consequently, reports with saved data not only minimize data transfer over the network, but also lighten the database server's workload.

You can schedule these reports in Enterprise XI so that they automatically refresh from the database on a predetermined basis. For example, if your sales database is only updated once a day you can run the report on a similar schedule. Sales representatives then always have access to current sales data, but they are not connecting to the database every time they open a report.

NOTE	Users require only View access to display report instances.

Workflow - Scheduling a report

1. The web client sends the schedule request in a URL through the web server to the web application server.

2. The web application server will interpret the request and the values sent, and will determine that the request is a schedule request. The web application server will send the schedule time, database logon values, parameter values, destination, and format to the specified CMS.
3. The CMS will ensure that the user has rights to schedule the object. If the user has sufficient rights, the CMS will add a new record to the System database. The CMS will also add the instance to its list of pending schedules.
4. The CMS checks its pending schedule list every 15 seconds. When the CMS finds a report that is ready to be scheduled, the CMS evaluates whether there is an available Report Job Server. The CMS sends the schedule request along with the report location, database logon, parameter, format, and destination information to the Report Job Server.
5. The Report Job Server requests the report from the Input FRS.
6. The Input FRS streams the report to the Report Job Server.
7. The Report Job Server spawns a Report Job Server Child executable to run the report. The Report Job Server Child opens the report and connects to the database to query for the report data.
8. The database returns the report data to the Report Job Server. The Report Job Server Child then processes the report.
9. Once the report is processed, the Report Job Server Child saves (submits a CORBA request to stream) the report to the Output FRS.
10. The Report Job Server Child removes itself from memory. The Report Job Server reports back to the CMS to let the CMS know that the report has been processed successfully.
11. The CMS updates the instance record in the system database to change the instance status to Success.

Workflow - Viewing a successfully scheduled instance

1. The web client sends the view an instance request in a URL through the web server to the web application server.
2. The web application server will interpret the requested page and the values sent in the URL request and will determine that it is a request to view the first page of the selected report instance. The web application server will send a request to the CMS to ensure that the user has rights to view the instance.
3. The CMS checks the system database to verify the user rights.
4. The CMS sends a response to the web application server to confirm the user has sufficient rights to view the instance.
5. The web application server sends a request to the Cache Server requesting the first page of the report instance.
6. The Cache Server checks to see if the page already exists. If the page does exist, the Cache Server can return the page to the web

application server. If the page does not exist, the Cache Server sends a request for the Page Server to generate the page.

7. The Page Server parent process determines whether it should start a new Page Server child to process the report or if there is already one available to process it. The Page Server child requests the report instance from the Output FRS.
8. The Output FRS streams a copy of the instance to the Page Server child. The Page Server child opens the report in its memory and checks to see if the report contains data. Since an Instance has data, the Page Server child will find data and generate pages.
9. The Page Server sends the EPF page to the Cache Server. The Cache Server stores a copy of the EPF page in its cache directory.
10. The Page Server sends the EPF page to the web application server.
11. If the DHTML Viewer is used, the web application server sends the DHTML page to the web server. The web server sends the DHTML page to the user's machine where it is rendered in the web client.
12. If the ActiveX Viewer is used, the web application server sends the EPF page to the web server. The web server sends the EPF page to the user's machine where it is rendered in the ActiveX Viewer in the web client.

Workflow - Viewing a successfully scheduled instance in the Advanced DHTML Viewer

1. The web client sends the request to view an instance in a URL typically through the web server to the web application server.
2. The web application server will interpret the requested page and the values sent in the URL request and will determine that it is a request to view the first page of the selected report instance.
3. The web application server will send a request to the CMS to ensure that the user has rights to view the object. The CMS checks the system database to verify the user rights.
4. The CMS sends a response to the web application server to confirm the user has sufficient rights to view the object.
5. The web application server sends a request to the RAS requesting the first page of the report instance.
6. The RAS requests the report instance from the Output FRS. The Output FRS streams a copy of the instance to the RAS. The RAS opens the report in its memory and checks to see if the report contains data. Since an instance has data, the RAS will find data and generate pages. The RAS holds the report in temp files in memory until it reaches a 30-minute idle time. The temp files are then deleted from memory.
7. The RAS sends the EPF page to the web application server. The web application server converts the EPF to DHTML.

8. The web application server sends the DHTML page to the web server. The web server sends the DHTML page to the user's machine where it is rendered in the web client.

Workflow - Processing a scheduled Web Intelligence report

1. The CMS checks its pending schedule list every 15 seconds. When the CMS finds a report that is ready to be scheduled, the CMS evaluates whether there is an available Web Intelligence Job Server. The CMS sends the schedule request along with the report location and other processing information to the Web Intelligence Job Server.
2. The CMS sends a create job message to the Job Server and Job Server creates (or contacts existing) a child process to run the job. The child connection is returned to the CMS
3. The CMS sends the run job message (including destination information) to the Job Server Child. The Job Server Child (using the procWebi processing plug-in) contacts the Web Intelligence Report Server asking it to open/refresh/save the report.
4. The Web Intelligence Report Server requests the report from the Input FRS. The Input FRS streams a copy of the report to the Web Intelligence Report Server.
5. The Web Intelligence Report Engine opens the report in its memory. The QT.dll generates the SQL from the universe that the report is based on.
6. The Web Intelligence Report Server connects to the database to run the query. The query data is passed through QT.dll to the Report Engine where the report is processed.
7. The Web Intelligence Report Server sends the finished report to the Output FRS.
8. The Web Intelligence Report Server notifies the Web Intelligence Job Server Child that the instance was a success.
9. The Job Server Child gets the processed report from the CMS/FRS and delivers it to the specified destination.
10. The Web Intelligence Job Server Child updates the instance status to the CMS.
11. The CMS updates the instance record in the system database to change the instance status to success.

Workflow - Viewing an OLAP Intelligence report

1. The web client sends the schedule request in a URL typically through the web server to the web application server.
2. The web application server will interpret the requested page and the values sent in the URL request and will determine that it is a request to view an OLAP Intelligence report object.
3. The web application server will send a request to the CMS to ensure that the user has rights to view the object. The CMS checks the system database to verify the user rights.

4. The CMS sends a response to the web application server to confirm the user has sufficient rights to view the object.
5. The web application server sends a request to the Input FRS to retrieve a copy of the OLAP Intelligence report. The Input FRS streams a copy of the OLAP Intelligence report to the web application server.
6. If using the DHTML OLAP Intelligence Viewer:
   a) The web application server opens the report in its memory, and then connects to the OLAP server to request the view of data.
   b) The OLAP server returns the view of data to the web application server.
   c) The web application server formats the data through the WCA into a DHTML page.
   d) The web application server forwards the DHTML page to the web server.
   e) The web server sends the DHTML page to the user's machine where it is rendered on the web client.
7. If using the ActiveX Viewer:
   a) The web application server sends the report file to the web server.
   b) The web server sends the report file to the web client. The web client opens the report file in memory.
   c) The web client connects to the OLAP server. The OLAP server returns the view of data to the web client.
   d) The ActiveX Viewer formats the data to display to the client and this last step is repeated as the user requests new views of the data.

Workflow - Viewing a Microsoft Word document

1. The web client sends the schedule request in a URL typically through the web server to the web application server.
2. The web application server will interpret the requested page and the values sent in the URL request and will determine that it is a request to view an instance that has been exported to Word format.
3. The web application server will send a request to the CMS to ensure that the user has rights to view the object. The CMS checks the system database to verify the user rights.
4. The CMS sends a response to the web application server to confirm the user has sufficient rights to view the object.
5. The web application server sends a request to the Output FRS to retrieve a copy of the Word document. These same steps apply for third-party objects that have been added to the system, except that they are stored in the Input FRS. The Output FRS streams a copy of the Word doc to the web application server.
6. The web application server forwards the Word document to the web server. The web server sends the Word document to the user's machine where it is rendered in the Word viewer on the web client.

Workflow - Scheduling to a destination

1. The web client sends the schedule request in a URL typically through the web server to the web application server.
2. The web application server will interpret the JSP page and the values sent in the URL request and will determine that the request is a request to schedule a document to a destination. The web application server will send the schedule time and destination to the specified CMS.
3. The CMS will ensure that the user has rights to schedule the object. If the user has sufficient rights, the CMS will add a new record to the system database. The CMS will also add the instance to its list of pending schedules.
4. The CMS checks its pending schedule list every 15 seconds. When the CMS finds a report that is ready to be scheduled, the CMS evaluates whether there is an available Destination Job Server. The CMS sends the schedule request along with the report location and destination information to the Destination Job Server.
5. Depending upon whether the report being sent to a destination is a report object or a report instance, the Destination Job Server requests the report from the Input or Output FRS. If the report is a report instance, the Destination Job Server requests the report from the Output FRS. The Output FRS streams the report to the Destination Job Server.
6. The Destination Job Server spawns a Job Server Child executable to run the report. The Job Server Child opens the report.
7. The Job Server Child sends the report to the specified destination. If the destination is to send a shortcut to a user inbox the Job Server Child will update the shortcut in the CMS database to point to the object location. If the destination is to send a copy to a user inbox, then the Job Server Child will copy the object to the Input File Repository Server.
8. The Job Server Child removes itself from memory. The Job Server reports back to the CMS to let the CMS know that the report has been sent successfully.
9. The CMS updates the instance record in the system database to change the instance status to success.

Workflow - Processing a scheduled List of Values

1. The LOV object scheduled and the schedule request is sent to the CMS.
2. The CMS checks the system database (if it does not have the information in memory) to verify that the user has permissions to schedule this object.
3. The CMS records the instance in the system database and adds the pending instance to its queue of pending schedules

4. When it is time for the LOV object to be processed, the CMS sends the schedule request to the LOV Job Server. The LOV Job Server retrieves the LOV object from the repository in the system database through the CMS.
5. The LOV Job Server spawns a Jobserverchild.exe and retrieves the LOV report template from the Input FRS.
6. The LOV Job Server updates the report in the Input FRS if there are any changes to the Business View or LOV definitions in the repository.

NOTE	If there is no report file in the Input FRS (the LOV object was added to Enterprise XI through the CMC, Import Wizard, or Publishing Wizard), the schedule will fail.

7. The Jobserverchild.exe connects to the reporting database to retrieve the records and populates the report with data.
8. The Jobserverchild.exe sends the report instance with saved data to the Output FRS.
9. The LOV Job Server notifies the CMS that the LOV object was successfully processed and the CMS updates the status of the instance.

Workflow - Viewing a report with prompts - List of Values

1. A request to view the report goes from web client to web application server.
2. The web application server checks the user's rights to view the report against the CMS.
3. The CMS checks the system database to confirm the user has appropriate rights.
4. The CMS responds to the web application server confirming that the user has appropriate rights.
5. The web application server sends a request for the first page of the report to the Cache Server. The Cache Server determines that it does not have the page.
6. The Cache Server requests the page from the Page Server.
7. The Page Server retrieves a copy of the report instance from the Input FRS. It opens the report in memory and checks to see if the report has data. In this case it does not, so the Page Server needs to process the report. The Page Server determines that it requires prompt information before it can process the report.
8. The Page Server sends a request to the web application server (through the Cache Server) notifying the web application server that a prompt is required.
9. The web application server sends a request to the Page Server (through the Cache Server) to determine which prompt is required.

10. The Page Server reads the report to determine which prompt is required and then notifies the web application server (through the Cache Server) which prompt is required.
11. The web application server sends a request to the Page Server (through the Cache Server) asking for a prompt window. The Prompt Engine on the Page Server retrieves the LOV object from the repository in the system database.

NOTE	If no report file is identified in the properties of the LOV object, (the LOV object was added to Enterprise XI through the CMC, Import Wizard, or Publishing Wizard) the prompt engine on the RAS will use the EROM API to create the report file on the RAS.

12. The Page Server requests that the RAS process the LOV report.
13. The RAS retrieves the LOV report from the Input FRS.
14. The RAS processes the report. If data requires updating, the RAS runs the query against the database.
15. The RAS returns the LOV value rowset to the prompt engine on the Page Server.
16. The prompt engine uses the EROM API to navigate through the values in the rowset returned by the report. The prompt engine on the Page Server creates the prompt window with the values included and sends the prompt window back through the Cache Server to the web application server.
17. The web application server sends the prompt window to the user so that the user can select values. The user selects values that are returned to the web application server.

NOTE	If dynamic cascading prompts are being used, the Prompt Engine will navigate through the values in the LOV report rowset to drill down to the next level and return the next level of values.

18. The web application server sends the values to the Page Server (through the Cache Server).
19. The Page Server now has the values and can process the report. Once it has processed the report, the Page Server can generate pages that are sent via the Cache Server and web application server to the web client.

Workflow - Scheduling a report - List of Values

1. The user requests the schedule window in InfoView. The request for the schedule window is sent to the web application server.
2. The web application server requires the prompt section of the schedule window, so it sends a request to the prompt engine on the RAS.
3. The prompt engine on the RAS retrieves the LOV object from the repository in the system database.

NOTE	If no report file is identified in the properties of the LOV object, (the LOV object was added to Enterprise XI through the CMC, Import Wizard, or Publishing Wizard) the prompt engine on the RAS will use the EROM API to create the report file on the RAS.

4. The RAS retrieves the LOV report from the Input FRS.
5. The RAS processes the report and if data requires updating it runs the query against the database. The RAS returns the LOV value rowset to the RAS prompt engine.
6. The prompt engine uses the EROM API to navigate through the values in the rowset returned by the report. The prompt engine creates the scheduling prompt window with the values included and sends the prompt window back to the web application server.
7. The web application server sends the scheduling prompt window to the user so that the user can select values.

NOTE	If dynamic cascading prompts are being used, the Prompt Engine will navigate through the values in the LOV report rowset to drill down to the next level and return the next level of values.

8. The user selects values, which are returned to the web application server.

Workflow - Scheduling a report - List of Values (CMC)

1. The user requests the schedule window in the CMC. The request for the schedule window is sent to the Web application server.
2. The Web application server requires the prompt section of the schedule window, so it sends a request to the Prompt Engine that resides on the Web application server.
3. The Prompt Engine requests the LOV object from the CMS.
4. The CMS retrieves the LOV object from the repository in the system database.

NOTE	If there is no report file identified in the properties of the LOV object, (the LOV object was added to Enterprise XI through the CMC, Import Wizard, or Publishing Wizard) the prompt engine on the web application server will use the EROM API to create the report file on the RAS.

5. The CMS returns the LOV object to the web application server.
6. The Web application server sends a request to the RAS Server to process the LOV report.
7. The RAS Server retrieves the LOV report from the Input FRS.
8. The RAS Server processes the report. If data requires updating, the RAS Server runs the query against the database.
9. The RAS Server returns the LOV value rowset to the Prompt Engine on the web application server.

10. The prompt engine on the web application server uses the EROM API to navigate through the values in the rowset returned by the report. The prompt engine creates the scheduling prompt window with the values included. The web application server sends the scheduling prompt window to the user so that the user can select values.
11. The user selects values, which are returned to the web application server.

NOTE	If dynamic cascading prompts are used, the prompt engine will navigate through the values in the LOV report rowset to drill down to the next level and return those values.

Performance Notes

This section has performance information and comparisons for Enterprise XI.

Scheduled reports compared to On Demand reports

The biggest performance issue specific to On Demand reports is the time it takes the reporting database to return uncompressed data across the network and the Page Server to complete the processing of the report. If the database takes 5 minutes to process the request, the user will wait that time and more for the requested page.

The time it takes to process a viewing request is going to be completely visible to the end user. If that time is too long, this report is a bad candidate for On Demand reporting.

When processing a request for a scheduled report, the biggest performance hit is moving a report with saved data from the Output FRS to the Page Server.

The time it takes for the database and Job servers to process the scheduling request is moved to the back end and away from the end user. Reports that take too long to process are candidates for scheduled reporting and viewing.

In almost all cases, viewing a scheduled report will be faster than viewing an On Demand report. Streaming a report with saved data from the Output FRS to the Page Server will be faster than retrieving a report template and getting new data from the database. Creating cached data from the saved data within a report is always faster than opening a report template, connecting to the database, waiting for the data to return and then creating the cached data.

Group Tree

When viewing reports using Page on Demand, the Group Tree populates the left frame of the viewers. It is used to quickly navigate to specific

groups. The cache file representing the Group Tree is called the totaller.etf.

The size of the totaller.etf file is in direct proportion to the number of groups and subgroups that are contained in the report. If the report has a large number of groups and subgroups, this file can grow to megabytes (MB) in size.

When viewing a report with a large group tree over the internet, the additional time may be required to download the totaller.etf.

The idea of grouping is to organize records into logical groups. The first design consideration is to make sure you are not grouping on unique records. If your report returns 500,000 records and you are grouping on Social Security Number, you are going to create 500,000 groups. The Group Tree is going to be unmanageable.

Report Formats

When you view reports in Crystal Reports format, you are utilizing the power of Page on Demand. To view a page requires the browser to download very small cache files.

When you view or export reports in third-party file formats such as Microsoft Word, Adobe PDF, or text, to view such a report, you must download the entire file.

Extremely large reports create extremely large third-party files. A 20,000 page report might create a 16MB PDF file so you may want to use alternatives such as email or FTP.

Servers components

The Intelligent Tier consists of the CMS, Cache Server, Event Server and Input and Output FRS. Aside from the Output FRS, the communication between these servers is in short bursts and they expect a timely response to their queries. They are memory and processor intensive.

The Processing Tier consists of all the Report and Job Servers, the RAS and the Page Server. These servers will use a great deal of disk space and memory, and are also very processor intensive.

For more information on performance settings for Enterprise XI please refer to the Sizing Recommendations Guide.

Reference and troubleshooting information

The following sections provide assistance in configuring, maintaining, diagnosing and repairing common issues with Enterprise XI.

Product guides

Much of the material in this document has been taken from the Enterprise XI Administration Guide (xir2_bip_Admin_en.pdf) and the

Installation Guide (xir2_bip_install_en.pdf). These and other reference documents (ODBC, SDK, Java) are found in the \Docs folder on the Enterprise XI installation CD. They can also be downloaded from our product guides web site.

Business Objects knowledge base

The Business Objects Customer Support web site features a wealth of information in knowledge base articles, white papers and downloadable files. The site includes a search engine which allows you to find information on the particular issue you are experiencing.

The support site is available at this link:
http://technicalsupport.businessobjects.com/

White papers

Among the documents located on the Business Objects Customer Support web site are some white papers that are recommended reading in relationship to Enterprise XI architecture.

Business Intelligence system scalability

The purpose of this paper is to provide an overview of scalability in general as it applies to BI systems. It reviews the evolution of the multi-tier BI delivery system, examining the purpose and function of each tier in the modern BI system, and finally examines how each tier needs to scale to meet consumer demands. Although specific benchmark measures are not defined by the document, it should serve as a framework around which these specific and meaningful measures can be developed. Download the white paper here.

Sizing Recommendations guide

There are three different aspects to consider when designing a scalable Enterprise reporting application: the application itself, the security and system data, and the actual hardware and configuration. Large deployments are often quite complex and it is critical to recognize the set of factors that can influence these three aspects of scalability. This document focuses on the actual hardware and configuration of an Enterprise XI deployment. The document can be used to estimate requirements for large deployments; however, we do recommend engaging Business Objects Consulting Services to help with planning in these situations. Download the white paper here.

Improving performance & scalability in OLAP Intelligence XI

This document discusses how to configure your OLAP Intelligence XI installation for greater performance and scalability. Download the white paper here.

Tools

There are many tools that can be used to diagnose and maintain your Enterprise XI system. These tools are regularly used by Business Objects customer support, but they are not supported applications because they are 3rd party applications.

Logging the workflow processes can be a valuable method of maintaining and diagnosing issues within Enterprise XI. Besides Enterprise XI logs, there are also Event viewer logs created by the operating system as well as web server logs. Download the white paper here.

Auditing

Auditing allows you to monitor and record key facts about your Enterprise XI system. It also allows you to better administer individual user accounts and reports by giving you more insight into what actions users are taking and which reports they are accessing. This information lets you be more proactive in managing the operation and deployment of your Enterprise XI system. This helps you better evaluate the value that Enterprise XI provides to your organization.

Auditing sample reports can be downloaded from our web site. For a complete list of the audit actions you can enable in Enterprise XI read the BusinessObjects Enterprise Administrator's Guide on pages 539-543.

Finding more information

For more information and resources, refer to the product documentation and visit the support area of the web site at: http://www.businessobjects.com/

www.businessobjects.com

