WebADE Report Generation Extension
Technical Guide

Prepared for:

Ministry of Forests
Information Management Group

Version 4.1.0
December 1, 2005

Prepared by:

VIVID SOLUTIONS

Document Change Control

REVISION NUMBER	DATE OF ISSUE	AUTHOR(S)	BRIEF DESCRIPTION OF CHANGE
2.3.0	April 1, 2002	Jason Ross	Updated documentation to WebADE Version 2.3
2.3.1	May 10, 2002	Jason Ross	Updated Report Queue Manager initialization procedure
2.5.2	Feb 10, 2003	Jason Ross	No changes
2.5.3	Feb 20, 2003	Jason Ross	Combining Reporting documentation into a single WebADE extension document.
3.1.0	Dec 10, 2003	Jason Ross	Updated to reflect the changes for Reporting Extension 3.1.0
4.1.0	Dec 1, 2005	Jason Ross	Updated to reflect the changes for Reporting Extension 4.1.0

Table of Contents

1. OVERVIEW

1.1 WEBADE REPORT GENERATION

WebADE applications often have requirements to support on-line and on-request batch reporting. The Report Generation API was developed in order to provide a common reporting interface as an extension to the existing WebADE framework.

1.1.1 Application Requirements

- Must support generation of reports on-line, and deliver reports to a user's browser
- Must support generation of reports off-line, and deliver reports to a user via e-mail
- Must provide capabilities to deliver reports in multiple file formats
- Must handle requests gracefully, i.e. the framework should isolate the application from the details of the underlying reporting tool, and should handle underlying error conditions wherever possible

1.1.2 Technical Requirements

- A simple, tool-independent API for the Web ADE to request reports from an external reporting engine (e.g., Crystal Reports) through a distinct set of public methods. The design of the API should support multiple underlying implementations (including incorporation of new reporting engine software) without requiring changes to the application.
- Ability to configure requests through the API in order to specify:
    1. *Name* of the report
    2. *Parameters* for the report
    3. *Output format* (e.g., PDF, RTF, CSV, etc.)
    4. *Delivery method* (e.g., direct (stream), SMTP, file)
    5. *Maximum wait time*
    6. *Other parameters*
- A queue manager to allow graceful handling of requests. The application must ensure that the number of concurrent requests forwarded to Crystal Reports is limited to the available Crystal licenses.
- Modularity to allow for selective implementation of the features described above so that features can be added on a priority basis as time and resources permit.
- Thread-safe.

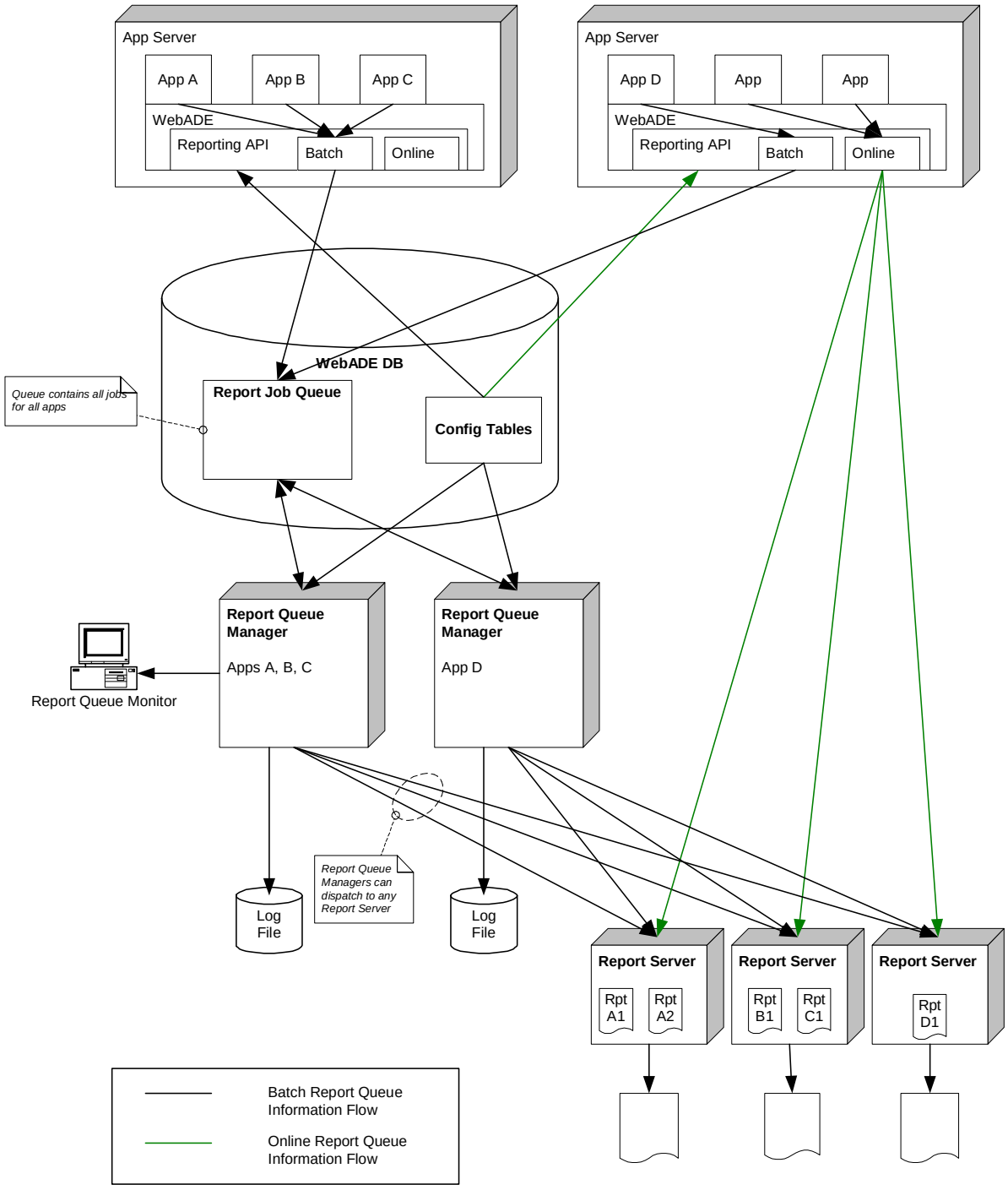
1.1.3 Report Generation Architecture

The following solution meets the basic reporting requirements, and provides a platform on which to provide enhanced features in the future.

Basic requirements include the capability to produce reports both on-line and in "batch" (or "off-line") mode, the latter delivered to the user via e-mail. In the initial release, the request will not include additional parameters such as "priority" and "wait time". The architecture will provide a generic interface, so that reporting tools other than Crystal Reports may be integrated in the future, but other reporting tools will not be investigated at this time.

The proposed solution can be illustrated as follows:

WebADE Reporting System Architecture
Prepared By: Martin Davis, Vivid Solutions Inc.
Last Revision: 6 February, 2002



When the user makes a request for a report, the WebADE Application determines whether the report is to be returned to the user immediately, or is to be submitted for processing off-line.

The available Crystal Enterprise licenses have to be allocated between on-line access and batch access. All applications which require on-line access to reports must be allocated sufficient licenses to support on-line reporting volumes.

1.1.3.1 On-line Scenario

If the report is to be returned to the user immediately, then WebADE submits this request to an internal queue, and then forwards the request to the Crystal Enterprise report server as soon as a session is available. When the report is available, it is returned to the user. This is suitable for small reports with short processing times.

If the number of requests exceeds the capability of Crystal Enterprise to generate the reports in a timely manner, that a user's request will time out. In this case, the user will be given a message that the server is currently busy, and has the following options:

- They can re-try their request at a later time
- They can submit the request for off-line processing

1.1.3.2 Off-line (Batch) Scenario

If the report is to be processed off-line, then WebADE inserts this request into a database queue, and returns a confirmation page to the user immediately, indicating that the request has been received. A separate batch process retrieves requests from the database, forwards the requests to the Crystal Enterprise report server, and e-mails the reports to the requesting user. This is suitable for large reports with long processing times, or reports which are generated in "data format" (i.e. formatted to be read by another application).

1.1.3.3 Phase 1 Supported Functionality

The following requirements are supported in the initial release:

- Immediate on-line delivery of reports
- Off-line (batch) delivery of reports via e-mail
- Support for Crystal Enterprise server-generated reports
- Support for Oracle Stored Proc-generated reports
- extensible design to allow future integration of additional reporting engines
- Report formats via Crystal: PDF, DOC, XLS, RTF and HTML
- Report formats via Oracle Stored Proc: CSV, any text-based format
- Queue manager within the Crystal Enterprise interface

The following requirements are NOT supported:

- Prioritization of requests (requests are processed in the order received)
- Report delivery via FTP
- Drill-down reports; Crystal-format reports

2. REPORT GENERATION API

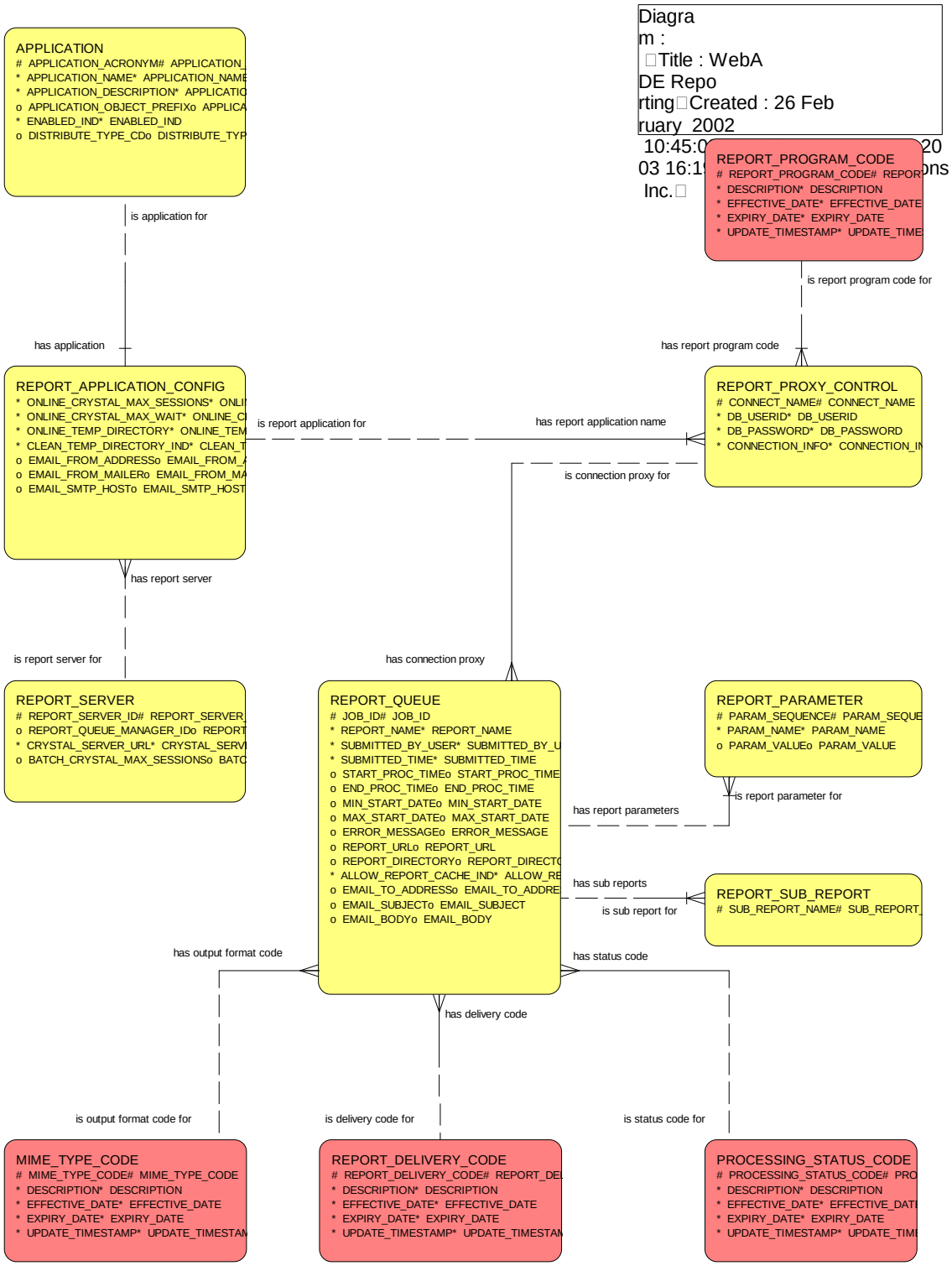
The Report Generation API allows applications to use Crystal Reports and Oracle Stored Procedures to generate reports and deliver them over the Web or Email to users.

Reports can be processed and returned in one of two ways:

Online: Reports are generated at the time the request is submitted and returned directly to the user.

Batch: Report requests are queued in a database table. One or more separate processes running the Report Queue Manager application process requests from this queue, generating the reports and returning them to the user via email.

2.1.1 Data Model



2.1.2 Table Description

2.1.2.1 Report_Application_Config

Contains the application-specific configuration information necessary to generate reports for that application.

Field	Description
Application_Acronym	The Web ADE application acronym.
Online_Crystal_Max_Sessions	The maximum number of Crystal Web Server sessions available to the online part of the Web ADE application.
Online_Crystal_Max_Wait	The maximum time that an online request will wait for a session.
Online_Temp_Directory	The local temporary directory to store generated reports before delivering them.
Clean_Temp_Directory_Ind	A flag indicating whether to delete report files from the temp directory after a successful delivery. Valid values are 'Y' and 'N'
Email_From_Address	The "From" address to set for reports delivered by Email.
Email_From_Mailer	The Mailer field to set for reports delivered by Email for this application.
Email_SMTP_Host	The SMTP host to use to deliver reports for this application.
Admin_Email_Address	The application administrator's email address.
Default_Report_Cache_Ind	The default setting of the report cache indicator, if a request does not specify one of its own.
Attempt_Threshold	The maximum number of attempts per request, before an email is sent to the application's administrator.

2.1.2.2 Report_Server

Contains the server-specific configuration information necessary to generate reports for a given report server to be used by the controlling Report Queue Manager.

Field	Description
Report_Server_Id	A unique key identifying the target Report Server.
Report_Queue_Manager_Id	The name of the controlling Report Queue Manager for this Report Server.
Crystal_Server_Url	The URL location of the Crystal Web Server used to generate reports for this Report Server.
Batch_Crystal_Max_Sessions	The maximum number of Crystal Web Server sessions available to the Report Server.

2.1.2.3 Report_Proxy_Control

Contains the connection information for application/reporting program/connect name combinations.

Field	Description
Application_Acronym	The Web ADE application acronym.
Report_Program_Code	The Report Program Code (Crystal or Oracle).
Connect Name	The Connect Name, generally the Role name as used in the Web ADE.
Connection_Info	For Crystal Reporting, the ODBC name of the report's database.

	For Oracle Reporting, the JDBC URL to the report's database.
Db_Userid	The user name used to connect to the Connection Info database.
Db_Password	The password used to connect to the Connection Info database.

2.1.2.4 Report_Queue

Contains the report requests in all stages of completion.

Field	Description
Job_Id	The unique identifier of the report request.
Application_Acronym	The Web ADE application acronym that generated this request.
Report_Program_Code	The program to generate the report.
Connect_Name	The connect name to use to look up the connection information for this request.
Report_Name	The name of the report on the Report Server for this request.
Email_To_Address	(Optional) the Email address to deliver the report to after completion.
Email_Subject	The subject of the Email message sent when a report is successfully generated.
Email_Body	The body of the Email message sent when a report is successfully generated.
Mime_Type_Code	The desired format of the report.
Processing_Status_Code	A flag indicating the current status of the request.
Report_Delivery_Code	The delivery method to use to deliver the completed request.
Report_Directory	The server side directory the Report Manager should place the report after being generated.
Report_URL	The Uniform Resource Locator needed by the user to retrieve the file once it has been generated.
Min_Start_Date	The date on or after the requested report should be executed and delivered.
Max_Start_Date	The latest date in which the report should be executed by. Reported as an error if date passes.
Allow_Report_Cache_Ind	A flag that indicates whether a report supports caching of generated reports.
Report_Filename	The name of the actual file generated by the Reporting API, and delivered to the user.
Admin_Email_Sent_Ind	A flag indicating if, when the request's attempt count exceeds the attempt threshold of this application, an email has been sent to the application administrator, to alert them of a problem with the report queue request.
Attempt_Count	The number of times this request has been attempted to be processed.

2.1.2.5 Report_Parameter

Contains the report request parameters to be passed to the Report Server for a given request.

Field	Description
Job_Id	The unique identifier of the report request.

Parameter_Sequence	A key used to order the parameters.
Parameter_Name	The name of the given parameter.
Parameter_Value	The value of this parameter.

2.1.2.6 Report_Sub_Report

Contains the request's sub reports for a given report.

Field	Description
Job_Id	The unique identifier of the report request.
Sub_Report_Name	The name of the given sub report.

2.1.3 Stored Procedures

The Report Generation API interacts with the ADE database through stored procedures. The stored procedures allow the Report Generation API to create, submit, retrieve, and process report requests.

To install a stored procedure or package, the script for the object must be compiled on the database. Stored Procedures can be compiled by executing the script in SQL Plus, or if a tool like Toad is being used to interface with the database, then the stored procedure or package can be compiled using the Stored Procedure editor.

The Web ADE application code corresponds to the Web ADE database application acronym field in the application table.

Stored Procedure/Package	Description
REPORTING_TYPES package	The REPORTING_TYPES package defines the cursor type. The cursor type is used by the stored procedures to return a record set to the calling application. The REPORTING_TYPES package must be installed first before installing any other stored procedures.
RPT_DELETE_REPORT	Removes the report request from the database queue.
RPT_INSERT_PARAMETERS	Inserts the list of parameter names and values for a given request into the database.
RPT_INSERT_SUB_REPORTS	Inserts the list of sub reports for a given request into the database.
RPT_SUBMIT_REQUEST2	Enters the report request into the report queue (Also uses the RPT_INSERT_PARAMETERS and RPT_INSERT_SUB_REPORTS procedures).
RPT_RESET_QUEUE	Resets all requests whose processing state is "Processing" to "Submitted" for the given Report Server.
RPT_UPDATE_REQUEST	Sets the processing status and (optional) error code for the target report request to the given values.
RPT_COMPLETE_ONLINE_REQUEST	Gets the connection info and (optional) Email settings for a report request that is to be processed online.
RPT_FIND_NEXT_REQUEST	Returns the next request on the queue for a given Report Server. This method also sets the status of the report to "Processing".

RPT_GET_PARAMETERS	Returns the report parameter names and values for the given report request.
RPT_GET_SUB_REPORTS	Returns the sub reports for the given report request.
RPT_GET_APPLICATION_CONFIG	Returns the online configuration information for the given application.
RPT_GET_REPORT_SERVERS	Returns the report server configuration information for all report servers assigned to the given Report Queue Manager.
RPT_GET_STATUS_COUNT	Returns the number of reports with the given status for the given Report Queue Manager
RPT_GET_REQUEST	Fetches the specified request from the queue.
RPT_SEARCH_REQUESTS	Searches the batch request queue for reports matching the search criteria.

2.1.4 Crystal Enterprise Interface

2.1.4.1 Crystal HTTP Request

The HTTP request which is sent to Crystal Enterprise is of the following format:

```
http://<host>/<crystal>/<dir>/<report.rpt>?<export>&<database>&<report params>
```

Where:

- <host> = server name and port number running Crystal Enterprise
- <crystal> = top-level reports directory
- <dir> = application-specific report directory
- <report.rpt> = the specific report requested (note that the application requests "<dir>/<report.rpt>"
- <export> = requested output format
- <database> = database connection information for the report (and sub-reports)
- <report params> = parameters for the report

Export format info is in the following format:

```
cmd=export&export_fmt=U2FPDF:0
```

Database connection info is in the following format (e.g. showing one sub-report):

```
user-HBSDEV.HBSDEV=HBS_DEV&password-HBSDEV.HBSDEV=HBS_DEV&user-
   HBSDEV.HBSDEV@HBS00001a=HBS_DEV&password-HBSDEV.HBSDEV@HBS00001a=HBS_DEV
```

Report parameters are in the name=value format where the parameter name is prefixed with "promptex-". For example:

```
promptex-param1=acb&promptex-param2=def&promptex-param3=ghi
```

Note that a "promptex-REF_CURSOR=0" parameter is required if the report accesses a stored procedure.

2.1.4.2 Session Management

Crystal Enterprise allows a limited number of sessions. A session is created automatically when a report is requested, and is then expired after a pre-defined time limit (default 20 minutes, can be configured as low as 1 minute).
The Reporting Application will manage sessions as follows:

7. When a response is received from Crystal Enterprise, the Reporting Application will extract and remember the following two cookies: “WCSID” and “apstoken”
8. On a subsequent request to Crystal Enterprise, the Reporting Application will include these two cookies in the request

In this way, the Reporting Application will re-use an existing session, or create a new one as necessary

3. INSTALLING THE WEBADE REPORT GENERATION EXTENSION

This section describes how to:

1. Install the database component for the WebADE Report Generation extension.
2. Install the WebADE Report Generation extension Java framework.
3. Deploy and configure web applications that use the framework.

3.1 WEBADE REPORT GENERATION DATABASE INSTALL

The following steps need to be performed to set up the WebADE Report Generation database environment:

- Obtain the WebADE Report Generation DDL Scripts used to create the WebADE Report Generation tables, indexes, foreign keys, stored procedures, and packages in the same ORACLE database as the WebADE Database.
- Install the database objects to the same database schema that hosts the main WebADE Database by executing the DDL scripts in SQL Plus, or an equivalent interface.

3.2 INSTALLING THE REPORT QUEUE MANAGER

The Report Queue Manager is a stand-alone process from the WebADE. Thus, the Report Queue Manager is installed separately from the WebADE application.

To install a Report Queue Manager, follow these steps:

- Acquire the WebADE.jar class library.
- Copy the WebADE.jar file to the desired Report Queue Manager working directory.
- Ensure that a properly configured "Log4J.properties" file exists in the working directory.

4. INSTALLING AND CONFIGURING REPORT GENERATION

4.1 INSTALLING AND CONFIGURING REPORT GENERATION

Configuring an application that uses the Report Generation API involves the following steps:

1. Populate the reporting tables with application configuration and database connection information.
2. If the application will be batching report requests, configure the report server and application configuration tables with the report queue configuration information.
3. Configure the desired Report Queue Manager to process this application's batch requests.
4. Deploy the application's reports to the Crystal Report Server instance responsible for managing the application's report requests. This association is bound by the Report_Application_Config table's Server_ID column in the WebADE database.

Once these steps are complete, the application can request online and batch reports.

4.1.1 Populating the Database Tables

The Report_Application_Config table contains the application-specific information for the Report Generation API. These parameters must be set as follows:

Parameter Name	Description
APPLICATION_ACRONYM	Must match the WebADE Application Acronym
REPORT_SERVER_ID	Set to the desired Report Server id(Only set if this application is using the report queue to process batch requests).
ONLINE_MAX_SESSIONS	Maximum number of concurrent Crystal sessions available to the web application
ONLINE_MAX_WAIT	Maximum time to wait (in seconds) if no Crystal session is available
ONLINE_TEMP_DIRECTORY	Temporary directory for reports generated by the web application.
CLEAN_TEMP_DIRECTORY_IND	Flag to delete reports from temp directory after a successful delivery(Values are 1 or 0 for true or false).
EMAIL_FROM_ADDRESS	The Email address to put in the From header for all reports delivered by Email for this application.
EMAIL_FROM_MAILER	The Mailer application value to put in the header for all reports delivered by Email for this application.
EMAIL_SMTP_HOST	The SMTP host used to deliver reports via Email.
ADMIN_EMAIL_ADDRESS	The application administrator's email address.

DEFAULT_REPORT_CACHE_IND	The default setting of the report cache indicator, if a request does not specify one of its own.
ATTEMPT_THRESHOLD	The maximum number of attempts per request, before an email is sent to the application's administrator.

The Report_Proxy_Control table must be populated with database connection information used to run the reports. This connection information is specific to the type of program generating the reports (e.g. Crystal Enterprise vs. Oracle Reports).

Column Name	Description
CONNECT_NAME	Name by which the application will refer to the database connection.
APPLICATION_ACRONYM	Must match WebADE acronym
REPORT_PROGRAM_CODE	Reporting program, must match a program code in the Report_Program_Code table
DB_USERID	User name used to log in to the database
DB_PASSWORD	Password for the given user name.
CONNECTION_INFO	For Crystal, this is an ODBC connection name. For Oracle, this is the JDBC connection string for the database.

The Report_Server table is populated with the report server-specific information used to perform report queue processing.

Column Name	Description
REPORT_SERVER_ID	A unique id that identifies the specific report server.
SERVER_URL	The HTTP URL location of the Crystal Web Server.
REPORT_QUEUE_MANAGER_ID	The name of the Report Queue Manager that will be responsible for feeding this report server report requests.
BATCH_MAX_SESSIONS	The maximum number of Crystal sessions available to this report server.

5. MANAGING THE REPORT QUEUE MANAGER

5.1 CONFIGURING AND RUNNING A REPORT QUEUE MANAGER PROCESS

A Report Queue Manager process can be created by executing the ReportQueueManager class at the command line, or as an NT service. ReportQueueManager can be executed using the following command line syntax:

"java –classpath <required WebADE and supporting libraries>
ca.bc.gov.mof.webade.reportgeneration.ReportQueueManager"

An example of this command line would be:
"java –classpath WebADE.jar;classes12.jar;mail.jar;log4j.jar;activation.jar
ca.bc.gov.mof.webade.reportgeneration.ReportQueueManager"

Before a Report Queue Manager can be run, however, it must first be configured. The ReportQueueADEConnection class contains the Report Queue Manager process-specific settings needed to run a Report Queue Manager process. The ReportQueueADEConnection class has the following static fields that require the appropriate values for the given Report Queue Manager.

Static Field	Description
WEBADE_JDBCURL	The JDBC database location URL of the ADE database
WEBADE_USERID	The user name used to log on to the ADE database
WEBADE_PASSWORD	The password for the user name
RQM_NAME	The unique name of this Report Queue Manager as stored in the Report_Queue_Manager_Id column of the Report_Server table.
TEMP_DIR	The local directory to temporarily store generated reports before delivery.
CLEAN_TEMP_DIR	Flag to delete reports after delivery (true or false)
LOG_FILE	The local file location to create the log file for the process. Ex: "C:\temp\rqm.log"
LOG_PRIORITY	The Priority class instance indicating the level of priority for log messages. One of: Priority.DEBUG, Priority.INFO, Priority.WARN, Priority.ERROR

First set these values in the ReportQueueADEConnection class, and then compile that class, similar to what is done for the WebADEConnection class.

5.2 READING THE REPORT QUEUE MANAGER LOG FILE

The location of the log file for the Report Queue Manager process is defined as a command line parameter. This log file will contain all log messages generated by the Report Queue Manager. If an error occurs, this should be the first place to look for details. See Section 6 for details on how to interpret and troubleshoot error messages.

5.3 MANAGING THE DATABASE QUEUE

If the database queue's length gets too large, processed report requests can be removed from the database queue. To remove requests, simply remove the rows from the Report_Queue table, along with any associated rows in the Report_Parameter and Report_Sub_Report tables.

Note: The records in the report queue can be useful for troubleshooting purposes. The database queue should be backed-up before rows are removed.

6. USING THE REPORT GENERATION EXTENSION API

6.1 INTRODUCTION

The Report Generation Extension API is a set of functionality that allows an application to generate reports via Crystal Reports and/or Oracle stored procedure calls. As a WebADE Extension, it implements the Extension API, and can be retrieved from the Application object by the getWebADEExtension() method. For more information about WebADE Extensions, see the WebADE documentation.

Before an application can use the Report Generation API to generate reports, the Report Generation database component must be installed to the same location that the WebADE database is installed. In addition there is some application-specific database configuration that needs to be performed. To find out more about the specifics of this installation and configuration, see the WebADE Administrator's Guide.

6.1.1 Changes from the Report Generation API

To submit a request using the new API, there are two significant changes; the way the ADEReportManager is retrieved, and the way the delivery target parameters are set.

1) ADEReportManager retrieval.

The ADEReportManager is now retrieved the same as all extensions:

```
ADEReportManager manager =
(ADEReportManager)app.getWebADEExtension(ADEReportManager.class);
```

To enable this, the package for reporting had to be changed to "ca.bc.gov.mof.webade.reportgeneration" instead of the old "ca.bc.gov.mof.webade.reporting" package. Make sure your code reflects this package change.

2) Delivery target initialization.

As variations for configuring a delivery target have increased, the implementation of having multiple constructors for each variation has become unwieldy. Therefore, all constructors that had delivery target parameters have been deprecated. (Delivery parameters are those parameters that apply to how the report is delivered; i.e.: response object, email-to address, drop-directory, etc.) The choices for initializing the ADEReportRequest are now as follows:

1) Use the default constructor for the ADEReportRequest, and then call the setters for all parameters.
2) User the constructor that requires all parameters for a request (except for delivery parameters).

No matter which constructor you use, after the report is configured, create and initialize an instance of the appropriate ReportDelivery subclasses (HttpDelivery, EmailDelivery, DirectoryDelivery) and call the request's setReportDelivery() method, passing in the delivery object.

6.1.1.1 Sample Code

A sample request looks like this pseudo-code:

```
ADEReportRequest  request = new ADEReportRequest(USER_ROLE,
                                                 getApplication().getCode(),
                                                 REPORT_PROGRAM,
                                                 REPORT_NAME
                                                 REPORT_OUTPUT_FORMAT,
                                                 SUBMITTED_BY_USER,
                                                 REPORT_PARAMETER_NAMES[],
                                                 REPORT_PARAMETER_VALUES[],
                                                 REPORT_SUBREPORTS[],
                                                );
if (isHTTPRequest) {
    request.setReportDelivery(new HttpDelivery(responseObject));
} else if (isEmailDelivery) {
    request.setReportDelivery(new EmailDelivery(emailToAddress, emailSubject,
emailBody));
} else if (isDirectoryDelivery) {
    request.setReportDelivery(new DirectoryDelivery(emailToAddress, dropDirectory,
pickupUrl));
}

//Optional
request.setMinStartTime(minStartTimeStamp);
//Optional
request.setMaxStartTime(maxStartTimeStamp);

Application app = this.getApplication();
ADEReportManager manager =
(ADEReportManager)app.getWebADEExtension(ADEReportManager.class);
if (isOnlineSubmission) {
    manager.submitOnlineReportRequest(request);
} else {
    manager.submitBatchReportRequest(request);
}
```

6.1.1.2 Loading the Reporting Extension with WebADE

Originally, the legacy Reporting API was included with the WebADE library. However, WebADE now supports the concept of "WebADE Extensions", allowing a variety of add-on API to be dynamically loaded into the WebADE to supply optional functionality that a WebADE application might require. As these are optional components, you will need to instruct the WebADE to load the WebADE Reporting Extension at application start-up, so you can retrieve the extension using the instructions above. To perform this configuration, see the following section in this guide

6.2 CREATING A REPORT REQUEST

To create a report request, you need to perform the following tasks:

- Retrieve the ADEReportManager extension from the Application object.
- Create an ADEReportRequest object, using the parameters and configuration settings for the particular report request.
- Create the appropriate ReportDelivery object, using the appropriate values for the particular delivery target.
- Pass the ADEReportRequest object to the ADEReportManager class, using either an online or batch request.

6.2.1 Retrieving the ADEReportManager

The Application class has a new method called getWebADEExtension() that returns the singleton object of the extension specified by the class passed in as a parameter. Call this method, passing in the ADEReportManager.class object, and cast the object returned to a ADEReportManager variable.

The ADEReportManager manages all report requests for a WebADE application's web component. It allows a report request to be processed immediately(online), or submitted to the queue to be processed at a later date(batch).

6.2.2 Registering the WebADE Reporting Extension

Before you can retrieve the ADEReportManager extension from the Application singleton, the report-generation-preferences.sql database script must be run against the WebADE database for your application.

When running the script, you will be prompted for the APPLICATION_ACRONYM. Enter the WebADE Application Acronym for your application, as it appears in the WebADE APPLICATION table.

For more information on WebADE Extension configuration, please see the WebADE User's Guide.

6.2.3 Create an ADEReportRequest Object

In order to create a report request, simply create and initialise an instance of the ADEReportRequest, using the appropriate constructor and setter methods.

A properly initialised request object requires the following:

- The name of the report to call.
- A list of parameter names and values to be passed to the report
- A list of sub-reports
- The output format of the generated report
- The program that will generate the report (Crystal or Oracle)

In addition to the request object, create a ReportDelivery sub-class of the appropriate type and the appropriate parameters:

HTTPDelivery constructor requires:

- HttpServletResponse object.

Email constructor requires:

- Email address to deliver the report to.
- Email Subject (optional)
- Email Body (optional)

Directory constructor requires:

- Email address to deliver the email containing the location of the file to.
- The server side path to a directory to drop the report into.
- The URL for the user to use to retrieve the file.

Finally, call the setReportDelivery() method of the request, passing in the initialised delivery object.

Refer to the javadoc for the Report Generation API for complete details.

6.2.4 Submitting the Report

To submit the report, simply call the appropriate method of the ADEReportManager object.

The submitOnlineReportRequest() method generates the report immediately and returns the result to the appropriate delivery target. If the delivery method is passed an HttpServletResponse object, the Report Generation API handles the remainder of the response to the requesting user.

The submitBatchReportRequest() method takes the request object and places it on the database queue to be processed when the appropriate ReportQueueManager gets a chance to complete it.

6.2.4.1 Batch Report Min/Max Start Time

Before submitting a batch report request, you may set the Min and Max Start Times by calling the setMinStartTime() and setMaxStartTime() methods on the ADEReportRequest object, providing a date and time. These settings prevent a report from being processed before the given min start date and after the given max date. These values are mutually independent, meaning that setting one does not require setting the other.

7. WEBADE REPORT GENERATION TEST APPLICATION

The WebADE Report Generation Test application provides a demonstration of an application using the Report Generation API to create and process report requests.

Before installing the application, there needs to be one test Crystal report and one test Oracle stored procedure created and configured to be called by the Report Generation Test application.

7.1 INSTALLING THE APPLICATION

- Create a new application in the servlet container called reporttest.
- Deploy the reportTest.war to this application.
- Create entries in the WebADE Database for the application that create the application name, report application configuration information, and report proxy control information so that there are two report connections configured, one for Crystal and one for Oracle.
- Edit and recompile the ReportRequestAction class for the application settings the appropriate constants needed to call the test Crystal and Oracle reports.

7.2 FEATURES DEMONSTRATED

The demo allows report requests to be created both for online and batch processing, using both Crystal and Oracle to generate reports.

8. TROUBLESHOOTING

8.1 REPORT GENERATION API TROUBLESHOOTING

8.1.1 Error Handling

This section describes the various errors thrown by the Report Generation API, the cause of those errors, and the appropriate actions to take to fix the problem.

ADEDatabaseException

Source	Report Generation API
Cause	The ADE database connection encountered an error.
Action	This probably means the ADE database connection was lost. When the database comes back up, the web application will have to be restarted. When Report Queue Managers encounter an ADEDatabaseException, they stop processing and wait until they can reconnect with the database. When they re-establish a connection, they will reset any requests they were processing when they lost connection and resume request processing.

ServerConfigurationException

Source	Report Generation API
Cause	An error was encountered while initializing the application.
Action	Check the report generation configuration settings for the application in the ADE database and restart the application.

SessionInitializationException

Source	Report Generation API
Cause	A Crystal session could not be retrieved before the MAX_WAIT time expired.
Action	This is a result of the request load being too great for the number of sessions allocated to this application. Increase the MAX_SESSIONS value. **Caution:** Increasing this number may cause the total number of allocated sessions for a Crystal Web Server to exceed the number of licenses. If this occurs, the server will start refusing requests. Another possibility is some requests may be taking too long to process and cause timeouts. If this is happening, steps should be taken to prevent those requests from being submitted.

ServerRequestException

Source	Report Generation API
Cause	An error occurred while processing a report request.

Action	This is probably a result of an error in the request's parameters and settings. Check the values before resending the request.

ReportDeliveryException

Source	Report Generation API
Cause	An error while delivering the report.
Action	This occurs after a report has been successfully retrieved from the report server, but an error occurred while delivering the report. If the delivery was by Email, check that the Email to address is correct and the SMTP host is functioning properly.

NullPointerException

Source	Report Generation API
Cause	The Application singleton returns null when getWebADEExtension() is called while trying to retrieve the ADEReportManager extension.
Action	This occurs when the ADEReportManager has not been properly loaded at application initialization. This is possibly due to the report-generation-preferences.sql file not being run or properly configured for your application. See this section for more information.