Deltek

Deltek Costpoint® 8.0

Integration Overview

October 27, 2021

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This publication contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, or translated into another language, without the prior written consent of Deltek, Inc.

This edition published October 2021.

© Deltek, Inc.

Deltek's software is also protected by copyright law and constitutes valuable confidential and proprietary information of Deltek, Inc. and its licensors. The Deltek software, and all related documentation, is provided for use only in accordance with the terms of the license agreement. Unauthorized reproduction or distribution of the program or any portion thereof could result in severe civil or criminal penalties.

All trademarks are the property of their respective owners.

Contents

Overview

Deltek Costpoint offers sophisticated integration capabilities, giving you the ability to:

- Invoke Costpoint reports and processes (long running actions) from third-party applications.
- Import and export Costpoint data to and from third-party applications.

This functionality allows you to integrate Costpoint with other applications used in your environment. For example, if you enter contracts into a contracts system, you can invoke the Project User Flow integration component from your contracts product to insert projects into the Costpoint system. Furthermore, if Costpoint is your system of record for specific data entities, such as employees, projects, or vendors, you can invoke Costpoint integration functions from other applications, to retrieve records from Costpoint and enter them into your other products. Integration saves you the labor of entering the same data in multiple systems, and the possibility of introducing inconsistencies between these systems.

Using the Costpoint Integration Console, you can specify exactly which Costpoint data integration functions, reports, or processes you want to expose as Web services. You can group Costpoint functions into logical modules (for example, a module that uploads and posts journal entries) to make the administration and deployment of related functions more manageable in large installations. You can choose to make your integration interfaces secure by default, requiring all access to be done through SSL. Furthermore, the Costpoint Integration Console allows you to test your integration components.

A Costpoint application that you choose to expose as a Web service effectively becomes a Costpoint API (Application Programming Interface), which can be used by other products in your organization or, if you choose to expose Costpoint Web services outside of your company network, can be used to integrate with your partners, contractors, or clients.

An extensibility framework (which is a separate feature available with Costpoint) allows you to customize Costpoint applications and, as part of the customization, make calls from Costpoint applications to third party applications.

Combined together, the integration and extensibility frameworks allow you to create two-way integrations with third-party products—invoking Costpoint applications through Costpoint Web services and having Costpoint applications invoke third party products through the extensibility framework.

Process and Report Integration

Costpoint Process/Report Integration allows you to invoke processes (actions) or reports in Costpoint using a Web service API.

Regular, Saved, and Archived Parameters

You can choose several ways to pass parameters to Costpoint via Web services:

- **Using regular parameters**: This allows you to invoke a process or report by supplying all parameters at runtime.
- **Using saved parameters**: This allows you to invoke a process or report using a previously saved process or report configuration.
- **Archived Reports Parameters**: This allows you to retrieve previously generated and archived reports.

Regular Parameters

If you decide to use regular parameters, you need to specify which process or report-related parameters you will be passing at runtime. You can choose either of the following:

- All parameters that you can enter in the corresponding Process/Report Costpoint application
- A subset of that list

If you use a subset of that list, it makes sense to add all required parameters first and then add optional parameters. The same defaults are used for the parameters that you do not specify as would be used if you left the parameters empty when running the process or report using the browser client.

When you invoke a process or report using regular parameters, the integration routine performs steps equivalent to:

1. Logging into Costpoint using the supplied user name and password.
2. Opening the processing or reporting application.
3. Creating a new record.
4. Updating the record using supplied service parameters.
5. Invoking the process or report.

Saved Parameters

If you use saved parameters, you will have to pass only one process or report-related parameter at runtime: the Saved Parameter ID.

When you invoke a process or report using saved parameters, the integration routine performs steps equivalent to:

1. Logging into Costpoint using the supplied user name and password.
2. Opening the processing or reporting application.
3. Populating the application screen by executing a query using a Saved Parameter ID that you pass to Costpoint as a supplied-by-service parameter.
4. Invoking the process or report.

Archived Report Parameters

If you use archived report parameters, you will have to pass only one report-related parameter at runtime: the Archived Report ID.

When you invoke a process or report using archived report parameters, the integration routine performs steps equivalent to:

1. Logging into Costpoint using the supplied user name and password.
2. Opening the Archived Reports (SYMRPTAR) application.
3. Populating the application screen by executing a query using an Archived Report ID that you pass to Costpoint as a supplied-by-service parameter.
4. Returning the selected archived report as part of an integration method return.

Tip: Using the saved parameters method may be preferable to using regular parameters in report methods because the saved parameters method allows you to specify custom Print Options, whereas the regular parameters method uses the default Print Options for the username specified.

Synchronous vs. Asynchronous Invocations

You can choose between two modes for invoking processes and reports:

- **Synchronous**: Equivalent to invoking processes and reports in **real-time**, using the Costpoint browser client
- **Asynchronous**: Equivalent to invoking processes in **batch mode**, using the Costpoint browser client

Synchronous Invocation

When you use synchronous mode, you must wait for the process or report to be completed to see results and continue with other work. This approach works well when you invoke a simple process or report that executes quickly, or when you need to know immediately if your process or report ran successfully.

Asynchronous Invocation

When you use asynchronous mode, you can continue with other work as soon as Costpoint submits your request to execute a process or report; you don’t have to wait for the process or report to run. This approach works well when you invoke a long-running process or report and don’t need to analyze results immediately or know exactly when the process or report ran. (For reports, you can configure Print Options to send you an email when a report finishes running.)

Data Integration

Costpoint Data Integration does more than just expose an underlying database using Web services. It allows you to export/import data directly from/to our applications. It performs the same data validation steps that occur when you enter data using the Costpoint browser client.

In fact, you can use Data Integration to effectively replace the Costpoint browser client with a Web services API.

When you **export** data from Costpoint using Data Integration, the integration engine performs steps equivalent to the following:

1. Logging into Costpoint using the supplied user name and password.
2. Opening the application.
3. Executing a query by filtering data using conditions on some result set columns.

When you **import** data into Costpoint using Data Integration, the integration engine performs steps equivalent to the following:

1. Logging into Costpoint using the supplied user name and password.
2. Opening the application.
3. Creating a new document or executing a query to modify an existing document.
4. Saving the data.

Costpoint Data Integration relies on established standards for importing and exporting data:

- Data is exported in XML format.
- All Web services are generated using the same algorithm, making it easy to create client programs that invoke Costpoint services.
- Client programs that invoke Costpoint services are automatically generated in Java language.
- To generate client programs for other languages (such as .NET), you can take a Web service WSDL file that we expose and generate a client program from it.

Support for Extensibility

Costpoint Data Integration allows you to add columns from extensibility units and use them as regular columns in export and import Web services.

Exporting/Importing as a String or Complex Data Object

Depending on your needs, you can export/import data in either of two ways: as a String or as a Complex Data Object.

String

Exposing data as a string lets you generate very simple Web services that:

- Treat data as a string
- Assume that both the client and server programs recognize that the string represents an XML document and know the rules needed to parse the document

This approach is convenient if you deal with XML documents on a regular basis (especially if you store your data internally as XML). It also lets you take advantage of a data archiving algorithm, which sends data in archived (zipped) format, shrinking documents 5 to 10 times on average.

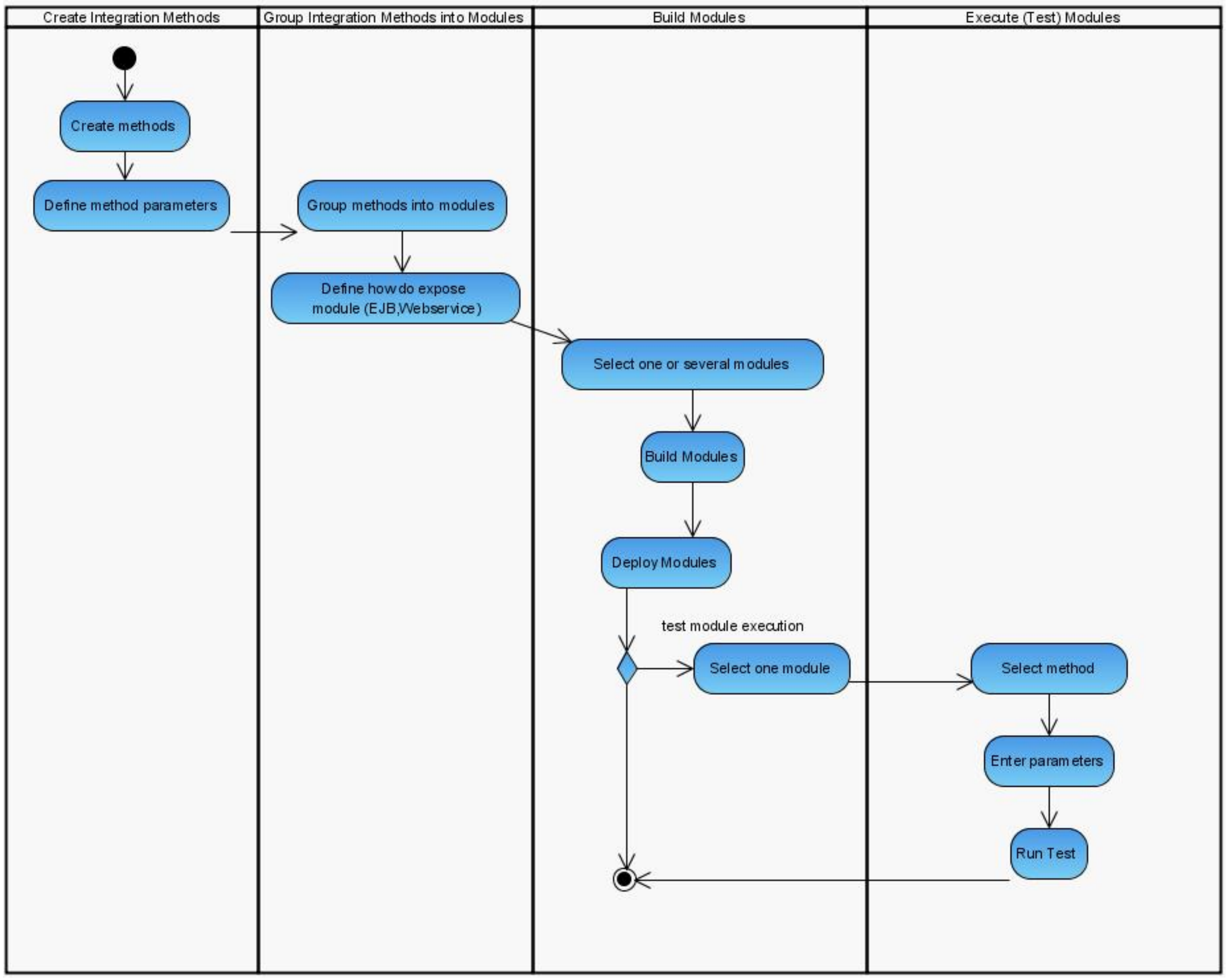
Complex Data Object

Exposing data as a Complex Data Object lets you generate Web services that declare the exact structure of the documents. This approach is convenient if you don't use XML documents on a regular basis or don't want to create programs to parse XML documents manually. This is also the preferred approach to exposing Web services if you are planning to register Web services in an Enterprise Service Bus.

By declaring the document's structure in the service definition, you make it possible for a third party tool (for example, .NET and SOA products) to automatically generate a client program for the Web service that represents an XML document as a collection of classes in the language that particular tool supports (for example, Visual Basic or Java). You can use Web services without being familiar with XML documents or knowing how to parse them; the Web services will appear as programs written in their native languages that accept/return objects.

Costpoint Integration Workflow

The diagram below shows you the steps you must follow to create and deploy any type of integration method.



Data Integration Export Services

Because Costpoint Data Integration exposes our applications (not just the underlying database), Deltek has provided an easy way for you to emulate the steps of opening an application and executing a query on the main screen or on a subtask to get the data you need. You can also choose which result sets (and columns in those result sets) you want to export. You perform these steps through the Integration Console user interface, with no coding needed.

To declare a Data Integration Export Service:

1. Create an Integration Method by selecting the Costpoint application with which you plan to work.
2. Specify the result sets that you want to export.
3. (Optional) Specify the columns in those result sets that you want to export.

   Group one or many Integration Methods into an Integration Module and generate a Web service from it.

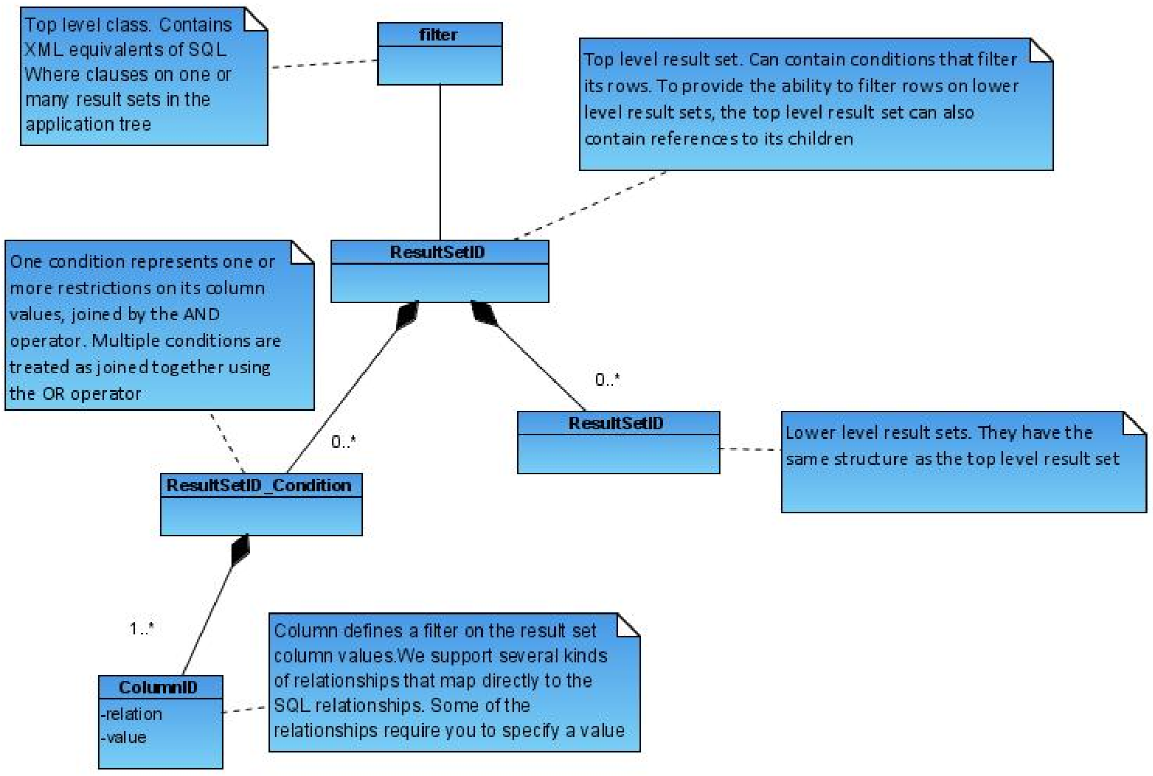
Regardless of the way in which parameters are passed to a Web service (String or Complex Data Object), the structure of the filter condition and the data accepted or returned remains the same. In other words, whenever you pass the same filter condition, you will get the same data back, regardless of how you generated the service.

Filter Conditions

You can specify filter conditions on one or multiple result sets within the application you select. You can think of a filter condition as a command to go to the application and automatically execute several queries on the application result set.

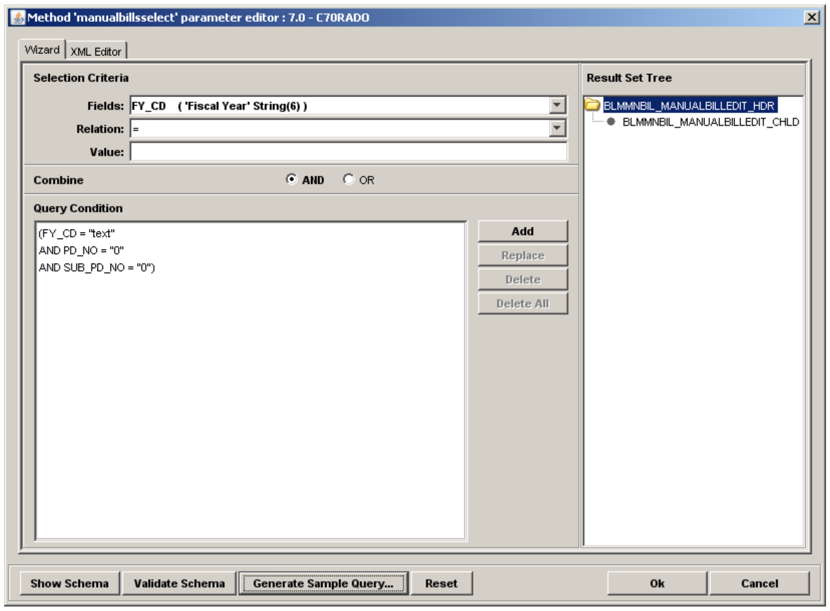
The diagram below shows how a filter condition works:

- If you pass data as a String (XML document), the classes in this diagram represent XML tags in your document.
- If you pass data as a Complex Data Object and use Java as the language for your client application, the classes are regular Java classes.
- If you pass data as a Complex Data Object and use some other language for your client application, your client generation tool will create similar classes in your native language.


Top level class. Contains XML equivalents of SQL Where clauses on one or many result sets in the application tree
filter
Top level result set. Can contain conditions that filter its rows. To provide the ability to filter rows on lower level result sets, the top level result set can also contain references to its children
One condition represents one or more restrictions on its column values, joined by the AND operator. Multiple conditions are treated as joined together using the OR operator
ResultSetID
0..*
0..*
ResultSetID
Lower level result sets. They have the same structure as the top level result set
ResultSetID_Condition
1..*
ColumnID
-relation
-value
Column defines a filter on the result set column values.We support several kinds of relationships that map directly to the SQL relationships. Some of the relationships require you to specify a value

To get a better idea of how to create a filter condition, use the Test Integration Module screen in the Data Integration application to generate a sample filter condition. Use the Query Wizard screen to create a custom filter condition and see how it is translated into an XML document.

This is how the condition appears in the Query Wizard.



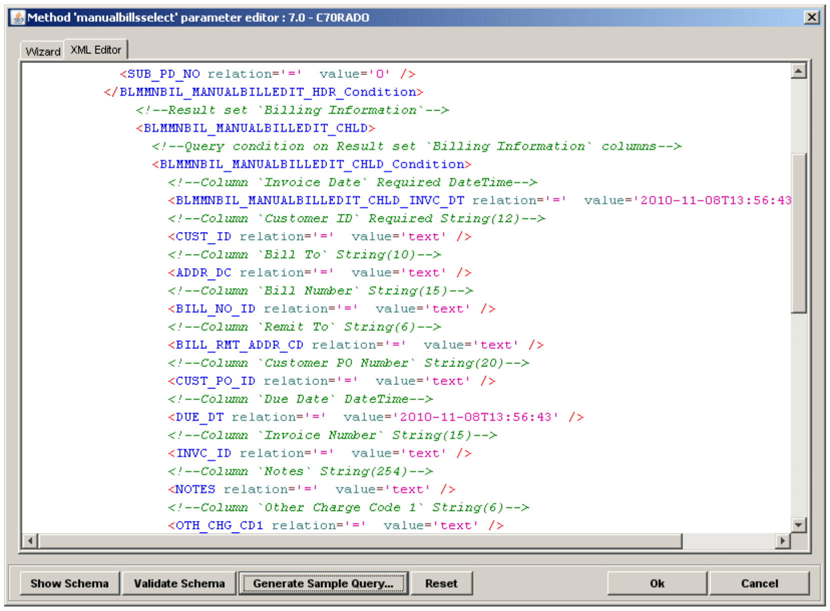
Fields: FY_CD ('Fiscal Year' String(6))
Relation: =
Value:
Combine [✓ AND] OR

Query Condition

```
(FY_CD = "text"
AND PD_NO = "0"
AND SUB_PD_NO = "0")
```

BLMMNBIL_MANUALBILLEDIT_HDR
BLMMNBIL_MANUALBILLEDIT_CHLD

This is the resulting XML code:



```
        <SUB_PD_NO relation='='  value='0' />
      </BLMMNBIL_MANUALBILLEDIT_HDR_Condition>
        <!--Result set `Billing Information`-->
        <BLMMNBIL_MANUALBILLEDIT_CHLD>
          <!--Query condition on Result set `Billing Information` columns-->
          <BLMMNBIL_MANUALBILLEDIT_CHLD_Condition>
            <!--Column `Invoice Date` Required DateTime-->
            <BLMMNBIL_MANUALBILLEDIT_CHLD_INVC_DT relation='='  value='2010-11-08T13:56:43
            <!--Column `Customer ID` Required String(12)-->
            <CUST_ID relation='='  value='text' />
            <!--Column `Bill To` String(10)-->
            <ADDR_DC relation='='  value='text' />
            <!--Column `Bill Number` String(15)-->
            <BILL_NO_ID relation='='  value='text' />
            <!--Column `Remit To` String(6)-->
            <BILL_RMT_ADDR_CD relation='='  value='text' />
            <!--Column `Customer PO Number` String(20)-->
            <CUST_PO_ID relation='='  value='text' />
            <!--Column `Due Date` DateTime-->
            <DUE_DT relation='='  value='2010-11-08T13:56:43' />
            <!--Column `Invoice Number` String(15)-->
            <INVC_ID relation='='  value='text' />
            <!--Column `Notes` String(254)-->
            <NOTES relation='='  value='text' />
            <!--Column `Other Charge Code 1` String(6)-->
            <OTH_CHG_CD1 relation='='  value='text' />
```

Data Integration Import Services

Because Costpoint Data Integration exposes our applications, Deltek has provided an easy way for you to emulate the steps of opening an application, entering new records or executing a query on the main screen or on a subtask, and then modifying existing records. You can also choose which result sets (and columns in those result sets) you want to import. You perform these steps through the Integration Console user interface, with no coding needed.

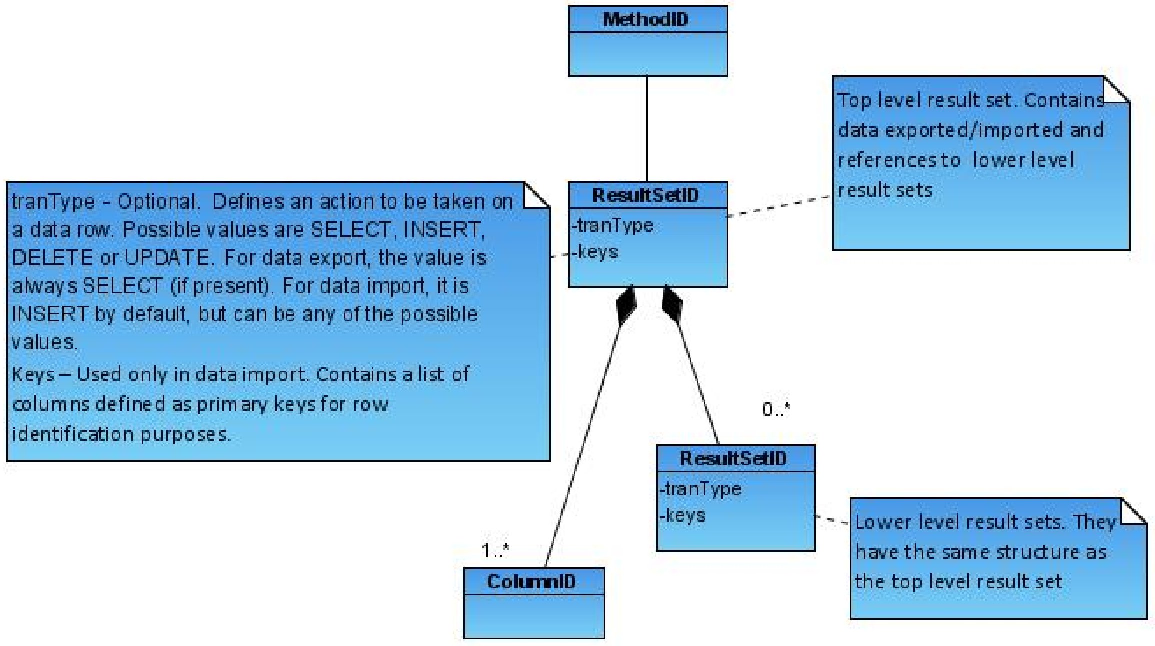
To declare a Data Integration Import Service:

1. Create an Integration Method by selecting the Costpoint application with which you plan to work.
2. Specify the result sets that you want to import.
3. Specify the columns in those result sets that you want to declare as primary keys.

   This step is optional if you intend to use this method only to insert data, and not to perform updates or deletions.
4. (Optional) Specify the columns in those result sets that you want to import.
5. Group one or many Integration Methods into an Integration Module and generate a Web service from it.

Structure of Exported and Imported Data

The diagram below represents the data exported and imported via the Data Integration application. The data is in the same format whether you are exporting or importing, which makes it easy to export data, make necessary changes to it, and then import it back into Costpoint.


MethodID
ResultSetID
-tranType
-keys
Top level result set. Contains data exported/imported and references to lower level result sets
tranType - Optional. Defines an action to be taken on a data row. Possible values are SELECT, INSERT, DELETE or UPDATE. For data export, the value is always SELECT (if present). For data import, it is INSERT by default, but can be any of the possible values.
Keys – Used only in data import. Contains a list of columns defined as primary keys for row identification purposes.
0..*
ResultSetID
-tranType
-keys
Lower level result sets. They have the same structure as the top level result set
1..*
ColumnID

Because the Data Integration application supports result set trees that can contain many levels, data is arranged in the “Russian Doll” format, in which child rows belong to their parent row. The opposite would be the “Flat” format, in which all result set rows are on the same level.

Rules for Specifying Data

XML language is sensitive to the order of the elements in the document. This means that you have to pass result set columns ordered in the same way as they appear in the XML schema that defines the method. The same rule applies to the result sets if you have multiple subtasks at the same level in the result sets tree.

When you pass data as a String vs. a Complex Object, this data is defined in the Web service WSDL as a string, not XML. This means that it has to be encoded if you are invoking Web services using a low-level programming model by manually generating SOAP envelopes.

For your convenience, you can use the **Generate Sample Document** or **Generate Sample Query** button in the Test Integration Module to generate a sample document or query. When you click this button, the application automatically generates a sample document or query that conforms to the document structure you specified when you created the method, but is populated with random data.

Data Export Methods

- When you generate a service to export data, you can expose as few columns as you want, meaning that you select only the data that you need.
- Whenever possible, a special LAST_MODIFIED pseudo-date/time column is exposed that allows you to select only recently modified data. Whenever a user updates documents in Costpoint, the application automatically updates certain system columns to keep track of who modified the data and when. While exposing such columns directly to the users is not usually done, it was decided to provide Integration Console users with functionality built on top of such system columns.

  The way this works is that the application tries to determine if data can be filtered in a particular result set using system hidden columns. If so, an extra column named RESULT_SET_ID + '_LAST_MODIFIED' is created in the data export method Query schema. This column is only available in query conditions; you cannot export this column as part of the result set data. This column has a date/time data type, and you can use it in the same way as any other date/time column from a query standpoint. You can build a filtering query that uses this column and includes any operators (=,>,<, between, etc.) that are supported for a date/time field. You can set multiple conditions on this column to support date ranges and other uses.

Data Import Methods

- When you generate a service to import data, you can also choose the columns that you want to import, but some rules govern whether or not you have to include the column in the service definition. The general rule is that you must pass as much data to Costpoint as you do when you use the Costpoint browser client. In other words, when you INSERT new data into Costpoint, you must pass at least all required columns (marked with an * in the Data Integration application).
- When you want to UPDATE, DELETE, or SELECT data in Costpoint, you must provide enough information so that the Data Integration application can uniquely identify the rows that you want to modify. When you define a service, you must specify a list of method primary key columns that will be used to identify a row in the result set. (Columns that are typically used as primary keys are marked PK in the Data Integration application, but, in some cases, you will have to specify additional/different columns as method primary keys.) When you invoke a service, you must specify values for these columns to identify the rows that will be modified.
- When you UPDATE data in Costpoint, you use method primary key columns to identify the rows that you want to change and use the rest of the columns to update data. (You cannot change primary key columns). The Data Integration application assumes that you don't want to update

the values of the columns that you don’t pass to Costpoint, meaning that the application will not change the value of any column that is not identified in the document that you pass.

- When you DELETE data in Costpoint, you use method primary key columns to identify the row that you want to delete. The Data Integration application ignores any other columns that you pass into Costpoint for that row.
- When you SELECT data in Costpoint, you need to use method primary key columns to identify the row that you want to select. The Data Integration application ignores any other columns that you pass into Costpoint for that row.
- In data import methods, some operations are not allowed on particular result sets because they don’t make sense from a business logic standpoint. You can also further limit the number of operations that the result set supports. For example, if you know that you will be using a Web service only to INSERT data, you can tell the tool that you only need to support INSERT. In this case, you will not have to specify method primary keys for method result sets.
- There are two special types of result sets that can only have one row: Filter and Single Row result sets. For such result sets, you don’t need to specify method primary keys because Costpoint knows that these result sets have a single row.
- Some result set columns have runtime defaulting logic when you INSERT data. If you don’t include a column in an input document, Costpoint will assume that you want to use the default value for this column (if it has any) or the null value if no default value exists.
- For performance reasons, Costpoint imposes certain restrictions on which columns in the top level result set can be treated as Method PK columns. To support UPDATE, DELETE, and SELECT operations, the system will have to identify a unique row at runtime with which you want to work.

  Since many top level result sets can be very large, Costpoint uses method primary key columns to modify the original SQL statement that populates data for the result set. By doing this, the application only needs to fetch one row from the database instead of fetching perhaps millions of rows and then filtering them. In most cases, you will not notice this restriction because columns you generally want to treat as method primary keys will be available to you.

  Lower level result sets do not have such restrictions because Costpoint assumes that such result sets generally do not have as much data. Data is already filtered on lower levels because the application only selects rows that belong to a particular top level result set row. Therefore, it is acceptable to filter data in the memory.
- You have ability to modify Primary key columns. The ability to modify primary key columns in the Data Import web services does not exist. Their values are used to find out which row the web service needs to modify. To get around this restriction, a separate optional portion has been added to the import document where you can set new values to the primary key columns.

  The application will continue to use primary key values listed in the main portion of the import document to select which row web service needs to modify. If the optional RS_NAME'+'_MODIFIED_KEYS portion is present, the application will assume that it contains new values for all primary key columns. This means that in case you have a compound primary key and you want to change the value of one primary key column, you will have to provide the values for all the other primary key columns even if you intend not to change them.
- Two new modes of importing data—MERGE and INSERT_OR_SELECT—exist. The application still supports the existing data synchronization modes (INSERT, UPDATE, DELETE, and SELECT).
  - MERGE mode first checks whether or not a row with ga iven primary key values exists. If so, it tries to update it by switching to the UPDATE mode. Otherwise, the system switches to the INSERT mode and tries to insert a new row.

- INSERT_OR_SELECT first tries to use SELECT mode to find a row based on the primary key values. If such row does not exist, the system switches to the INSERT mode and tries to insert anew row.

Data Compression for String Services

To minimize the amount of data transferred during Web service execution, the Data Integration application allows you pass and get data in compressed (zip) format.

The compression algorithm, described below, is available only for String-based services. You must complete an extra step to pass/retrieve data in Archived format.

- When you invoke a service to **export** data, data comes back as a byte array in the form of a zipped data string (using the standard java java.util.zip.Deflater class). Web services pass this data as a standard XML base64Binary data type. To convert data that you receive as a result of service invocation, you must unzip it. (For Java clients, you can use the java.util.zip.Inflater class; for .NET C# clients you can use the System.IO.Compression.DeflateStream class).
- When you invoke a service to **import** data, you must perform the opposite step to pass data to Costpoint services. You need to compress data into a byte array. (For Java clients, you can use the java.util.zip.Inflater class; for .NET C# clients you can use the System.IO.Compression.DeflateStream class.)

To implement archiving with Java clients, you can use a sample Java client for Web services that is generated by the Integration Console. Review the sample to see how the steps are done in Java and either use them as is, or recreate the same steps using another language, such as .NET.

Return Messages

You may receive error messages when you invoke a service. These messages depend on the type of service you generate and whether or not error conditions occur during invocation.

Data Export Services return data as a result of service invocation, but other services return a MessageResponse object. If a problem occurred, this object describes the severity of the problem and may contain one or more message objects that describe the problem.

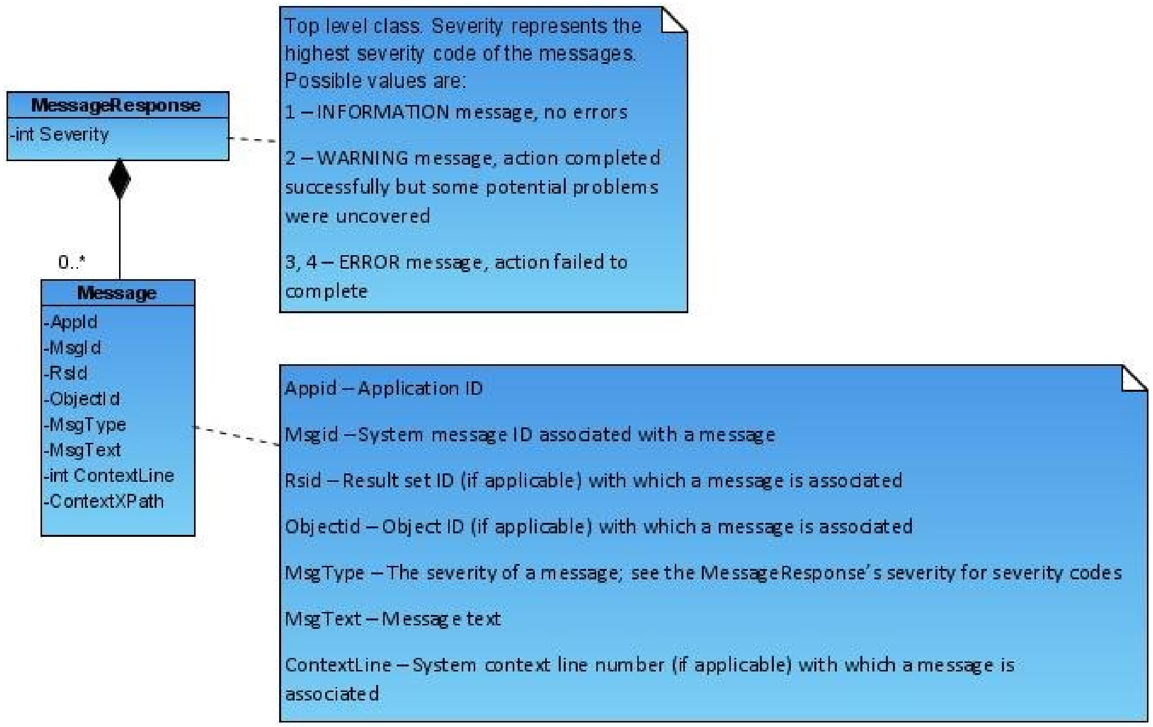
Whenever possible, you will receive a message describing the particular problem that occurred. When you invoke a Web service, control goes to the WebLogic security provider first. WebLogic tries to authenticate you to the system you are logged into, using your user name (concatenated with the system name) and password.

If authentication fails, WebLogic generates an exception. Since control never gets to the Web service runtime classes, the application cannot convert this exception into a MessageResponse object. These errors appear as exceptions in XML SOAP fault messages in the Web service.

If authentication completes successfully, control goes to the Web service runtime. From this point, runtime classes can catch all exceptions and convert them into MessageResponse objects. Since data export services do not return a MessageResponse object, the application will still have to generate an exception. Then the application will convert the MessageResponse object into an XML string and return it as an exception message. This way, you can get an exception message string, parse this xml string, and get a MessageResponse object that you can analyze.

Attention: See http://www.w3.org/TR/soap12-part1/#soapfault for more details about fault messages.

The diagram below shows you the structure of the service response message.



Top level class. Severity represents the highest severity code of the messages. Possible values are:

1 – INFORMATION message, no errors

2 – WARNING message, action completed successfully but some potential problems were uncovered

3, 4 – ERROR message, action failed to complete

Appid – Application ID

Msgid – System message ID associated with a message

Rsid – Result set ID (if applicable) with which a message is associated

Objectid – Object ID (if applicable) with which a message is associated

MsgType – The severity of a message; see the MessageResponse's severity for severity codes

MsgText – Message text

ContextLine – System context line number (if applicable) with which a message is associated

The structure of the MessageResponse object is very simple, but some attributes in the message object apply only to particular situations. Some attributes (such as MsgId or ContextLine) describe internal system entities that may be difficult for third party systems to interpret. For the internal ContextLine attribute, Deltek decided to create a corresponding ContextXPath attribute that can be used to map a particular message with a row in the document that you are trying to import into the system.

Report Services Return Messages

Special types of response objects are used for Web services that invoke reporting applications and return report output. For each type of report output, the Web service creates a new response object: HTMLReportResponse, PDFReportResponse, ExcelReportResponse, XMLReportResponse, or RTFReportResponse. These objects contain all the properties of the original MessageResponse object plus extra properties to return report data.

HTMLReportResponse has two extra properties :

- HtmlReportData of type string
- Array of bytes that represent used by report images

All other report return types have one extra parameter: byte array.

Integration-Specific Error Messages

If an error occurs during the service authentication process, an exception will be generated by WebLogic.

After successful authentication, control goes to the service runtime that analyzes service parameters and returns errors if the parameters are invalid.

If the parameters are correct, the service runtime sends these parameters to the regular Costpoint runtime classes. Costpoint runtime classes perform the same steps that would occur if a user performed the same activity using the regular Costpoint user interface. If an error occurs at this point, the Costpoint runtime classes return the same error messages that the user would see in the browser UI . The service runtime converts these errors into a MessageResponse object and returns to the client program.

Below are the integration-specific error messages that can occur at runtime:

Message ID	Message Text	Cause
CP_SYSSEC_WS_GNRL_FAILURE	Invalid login information provided. Web Services authentication failed.	This error can occur when the Integration runtime cannot read the user rights info from the system tables.
CP_SYSSEC_WS_GNRL_FAILURE_PARMS	Invalid login information provided. Web Services authentication failed. System: %1, User ID: %2, Timestamp: %3, Reason: %4.	This error occurs when the user does not have rights to access the system via Integration Services. By default, Costpoint users do not have rights to access integration. You have to grant the rights in the Manage Users application.
CP_SYS_NO_CHANGE_RIGHTS_PK	An attempt to persist a new value for the following Primary	This warning states that the runtime ignored the new value for the Primary Key.

Message ID	Message Text	Cause
	Key field was denied. Field: %1 Value: %2	
CP_SYS_NO_CHANGE_RIGHTS_NOEDIT	An attempt to persist a new value for the following non-editable field was denied. Field: %1 Value: %2	This warning states that the runtime ignored the new value for the non-editable field.
CP_SYS_NO_CHANGE_RIGHTS_LS	An attempt to persist a new value for the following field was denied based on labor suppression rules. Field: %1 Value: %2	This is a warning saying that runtime ignored a new value for a column that the user doesn't have rights to modify because of labor suppression rules.
CP_SYSSEC_WS_INVALID_SYSTEM	Provided to the integration method system %1 is invalid or different from the system %2 used to authenticate user.	This error is returned by the integration runtime when you pass an incorrect system name as a parameter to the service.
CP_SYS_WS_NO_OPERATION_RIGHTS	An attempt to %2 for the result set %1 in the integration method %3 was denied because the method doesn't support such an operation.	This error is returned by the integration runtime when you try to submit a result set row with an unsupported operation (for example, tranType="DELETE", but a particular result set only supports INSERT in the data method definition).
CP_SYS_NO_SAVE_RIGHTS	An attempt to %2 for the result set %1 was denied due to insufficient rights.	This error is returned by Costpoint runtime when it determines that you do not have rights to modify a particular row. This can happen when result set rights have been changed since you designed and/or deployed your service.
CP_SYS_WS_GNRL_INT_FAILURE	General Integration Framework Failure during integration method %1	This error is returned by the integration runtime when an unexpected error occurs during method invocation. Refer to the details

Message ID	Message Text	Cause
	invocation. Details: %2.	returned by the error message for more specific information.
CP_SYS_WS_GNRL_FRAMEWORK_FAILURE	General Framework Failure during integration method %1 invocation. Details: %2.	This error is returned by the Costpoint runtime when an unexpected error occurs during method invocation. Refer to the details returned by the error message for more specific information.
CP_SYS_WS_INVALID_DOC	Integration Framework failed to parse input document during integration method %1 execution. Please make sure document is formatted correctly. Details: %2.	This error is returned when the integration runtime cannot parse the input document or query condition in data services. Refer to the details returned by the error message for more specific information.

Example of MessageResponse Object

The Context Path below points to the second child of the first parent record in the document submitted to the service.

```
MethodResponse:
 Severity: 3
 Messages array:
  Message:
   AppId: BLMMNBIL
   MsgId: CP_OBJ_INVALID_COMP
   RsId: BLMMNBIL_MANUALBILLEDIT_CHLD
   ObjectId: ADDR_DC
   MsgType: 3
   MsgText: This |ADDR_DC| does not exist for this Company.
   ContextLine: -59999
   ContextXPath:
/BLMMNBIL_MANUALBILLEDIT_HDR[position()=1]/BLMMNBIL_MANUALBILLEDIT_CHLD[position()=2]
  Message:
   AppId: BLMMNBIL
```

Example of MessageResponse Object Generated as an Exception String When an Error Occurs in Data Export Services

```
<?xml version='1.0'?>

<MethodResponse>

<Severity>4</Severity>

<Message>

<AppId>null</AppId>

<MsgId>CP_SYSSEC_WS_INVALID_SYSTEM</MsgId>

<RsId>null</RsId>

<ObjectId>null</ObjectId>

<MsgType>0</MsgType>

<MsgSeverity>0</MsgSeverity>

<MsgText><![CDATA[

Provided to the integration method system C62RQM is invalid or different from the
system C62RADO used to authenticate user.

]]></MsgText>

<ContextLine>-60000</ContextLine>

<ContextXPath></ContextXPath>

</Message>

</MethodResponse>
```

Integration Security

This section provides a brief overview of the security associated with Costpoint Integration services.

Attention: For more information about Costpoint security, see the *Deltek Costpoint 8.0 Security* guide.

Web Services

To secure a Web service, you can specify at design time if you want to use the UsernameToken Profile or SAML Token Profile.

Attention: For more information, see the following WebLogic documentation:

http://docs.oracle.com/cd/E24329_01/web.1211/e24488/overview.htm

To invoke a Web Service that is using the UsernameToken Profile, the client program must provide security credentials (a username and password) for authentication and authorization purposes.

- **Username**: This is the regular Costpoint username, concatenated with the system name the user wants to connect to. The format is:

  CostpointUserName __ Costpoint

  Where:

  - CostpointWebUserName is the Costpoint user name used on the login screen to log into Costpoint.
  - CostpointWeb is the system name used on the login screen to log into Costpoint.
- **Password**: This is the regular Costpoint password that the user uses to log into Costpoint.

The user must be a valid Costpoint user who:

- Has all rights to the underlying applications and result sets to select/update data via the regular Costpoint UI
- Has the **Allow Application Access via Integration Services** check box selected on the Authentication tab in the Manage Users application

UsernameToken Profile is a standard that is defined in the OASIS WS-Security UsernameToken Profile specification.

Attention: For more information, see:

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

Warning: The UsernameToken Profile expects passwords to be passed as PasswordText, meaning that they are passed in clear text over the wire. Therefore, Deltek recommends that you use SSL to protect sensitive information if you invoke Web services in an unsecured environment. In the future, based on user feedback, the Integration application may include additional standard ways of passing security credentials, such as X.509.

To invoke a Web Service that is using SAMLToken Profile, the client program must obtain a valid SAML token and build a properly formatted Web services SOAP/XML request with SAML token included as defined in the OASIS WS-Security SAMLToken Profile specification.

Attention: For more information, see:

https://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SAMLTokenProfile.pdf

Use Default System

Deltek has found out that in some situations, it is technically difficult or impossible to concatenate the system name to the user name when you invoke a service from a third party. To deal with such situations, Deltek introduced a **Use Default System** option to invoke a service.

If you choose this option, you pass a username without the system name to the Web Service. The WebLogic authentication provider in such situations authenticates the user against the first system in the list of systems defined by the sysNames parameter in the enterprise.properties file. You still have to pass the system name as a parameter to the Web service, and the integration runtime checks to make sure that this system name is equal to the system that was used to authenticate the user.

This effectively means that you can use this mechanism only when you have one system. If you have a requirement to support many systems and cannot attach the system name to the user name, contact the Deltek Customer Care team for help.

Note: Currently, the Data Integration application does not support any kind of message security (Digital Signatures and Encryption) under the assumption that users will use SSL to secure messages. However, based on user feedback, support for message security may be added in the future.

Integration Console Specific Parameters in the WebLogic Server Startup Script

By default, WebLogic does not validate incoming Web service requests against the Web service WDSL. WebLogic tries to map parts of an incoming request with the Web service WSDL and silently discards all parts that are not formatted according to WSDL. This is done because validation adds some processing overhead.

While this behavior is acceptable in a production environment, where it is assumed that the client knows how to generate a correct request, it is very important to have validation in the test environment. This is especially true for object-based Web services, where submitted documents can have a very complex structure. In a test environment, it is very common that incoming requests have incorrect namespace declarations or misspelled result set columns. Without schema validation, WebLogic silently ignores result set columns or rows, creating problems that are very hard to troubleshoot.

For these reasons, you should modify the script that starts the WebLogic server in a test environment by adding an extra BEA parameter to the startup file of Weblogic server:

```
-DWebLogic.wsee.validate_request=true
```

Appendix A: Sample Functions to Archive Data with Java and C# Client Programs

These functions are copied from a generated test client application. You can use them to compress/decompress data when you are in the module definition screen and you choose to use the Pass Data as String and use Achieved Text String Data Format.

Java Functions

```
  private static final int BUFFER_LENGTH=65536;

  /**
   * Converts zipped data into clear text
   */
  public static String extractData(byte[] zippedBytes) throws
      Exception {
    // unzip data using standard java Inflater class
    java.util.zip.Inflater decompresser = new java.util.zip.Inflater(true);
    decompresser.setInput(zippedBytes, 0, zippedBytes.length);
    byte[] result = new byte[BUFFER_LENGTH];
    StringBuilder out=new StringBuilder();
    int resultLength = -1;
    while ((resultLength=decompresser.inflate(result))>0){
       out.append(new String(result, 0, resultLength, "UTF-8"));
    }
    decompresser.end();
    return out.toString();
  }

  /**
   * Converts clear text into hexed zipped data
   */
  public static byte[] archiveData(String sourceText) throws
      Exception {
    // Encode a String into bytes
    byte[] input = sourceText.getBytes("UTF-8");
    // Compress bytes
       java.io.ByteArrayOutputStream binaryData=new java.io.ByteArrayOutputStream();
    java.util.zip.Deflater compresser = new
java.util.zip.Deflater(java.util.zip.Deflater.DEFAULT_COMPRESSION,true);
    compresser.setInput(input);
    compresser.finish();
    int compressedDataLength = -1;
       byte[] output = new byte[BUFFER_LENGTH];
       while ( (compressedDataLength = compresser.deflate(output)) > 0) {
      binaryData.write(output,0,compressedDataLength);
    }
    return binaryData.toByteArray();
  }
```

.NET Functions

```
/*
*.Net code
*/
/*
  public byte[] archiveData(String sourceText){
               byte[] srcBytes =
System.Text.ASCIIEncoding.UTF8.GetBytes(textBox1.Text);
                System.IO.MemoryStream srcStream = new
System.IO.MemoryStream(srcBytes);
                System.IO.MemoryStream zippedStream = new System.IO.MemoryStream();
                using (System.IO.Compression.DeflateStream zipper = new
System.IO.Compression.DeflateStream(zippedStream,
System.IO.Compression.CompressionMode.Compress))
                {
                                  byte[] bytes = new byte[4096];
                                  int n;
                                  while ((n = srcStream.Read(bytes, 0, bytes.Length))
!= 0)
                                  {
                                         zipper.Write(bytes, 0, n);
                                  }
                           }
                return zippedStream.ToArray();
       }

  public String extractData(byte[] zippedBytes){
       System.IO.MemoryStream ms = new System.IO.MemoryStream(zippedBytes);
       using (System.IO.Compression.DeflateStream zip = new
System.IO.Compression.DeflateStream(ms,
System.IO.Compression.CompressionMode.Decompress)){
              byte[] bytes = new byte[4096];
              StringBuilder output = new StringBuilder();
              int n;
              while ((n = zip.Read(bytes, 0, bytes.Length)) != 0){
                     output.Append(ASCIIEncoding.UTF8.GetString(bytes, 0, n));
              }
              return output.ToString();
       }
  }
```

Appendix B: Accessing Integration Services from Costpoint Servers

Warning: This section only applies if you need to access CP Web Services through IIS server using SOAP protocol.

If web services are accessed ONLY through REST protocol, then the configuration discussed in this section can be skipped as REST access to Costpoint Web Services works with default Costpoint IIS configuration (that is, no additional setup is needed).

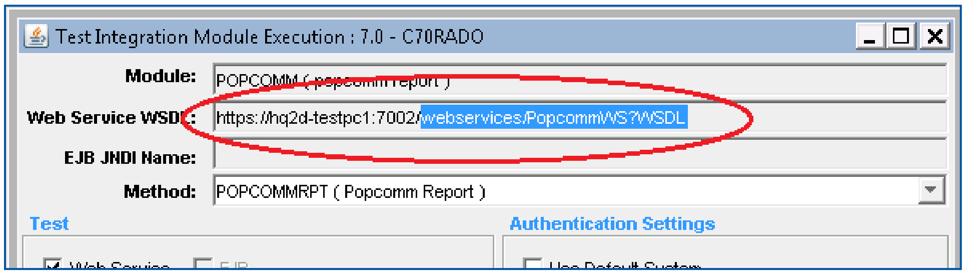
If Costpoint is deployed on premise but you intend to access some areas of Costpoint from public internet and want to restrict Web service access using Black/White list capability, then:

- Do NOT perform the configuration below on an externally facing IIS server. That is, do not enable SOAP access for public internet since Web services invoked using SOAP protocol will not honor the black/white list policy.
- You can invoke Costpoint Web services from a public internet using REST protocol with black/white lists being applied/honored.

All integration modules are deployed under the /webservices virtual folder.

The typical URI to access an integration module using SOAP protocol is http://<servername>:<port>/webservices/<servicename>.

To determine the correct URI to access an integration module, use the Test Integration Module screen in the Integration utility, as shown below.


Test Integration Module Execution : 7.0 - C70RADO

Module: POPCOMM (popcomm report)

Web Service WSDL: https://hq2d-testpc1:7002/webservices/PopcommWS?WSDL

EJB JNDI Name:

Method: POPCOMMRPT (Popcomm Report)

Access from WebLogic Server

All integration modules are deployed homogenously on all WebLogic servers including dedicated servers. In case of cluster, the admin server does not host any Web services. Web services can be accessed based on the URI of a specific server as discussed above or through some form of load balancing (for example, virtual IP or IIS).

When you need to expose integration modules to external users who do not have direct access to WebLogic servers, IIS can be used to proxy the requests to WebLogic. Based on the servers you decide to expose, you can either use the existing Costpoint Web Server (IIS) or create a new virtual Application. The standard installation of Costpoint Web server is configured to proxy UI requests to a single Weblogic server or all nodes of a cluster. It does not proxy to dedicated servers. By creating a new virtual Application in IIS, you can control and direct Web service calls to a single Weblogic server (which can be a node or a dedicated server) or a group of Dedicated Weblogic servers by providing a comma-separated list or by means of a virtual IP.

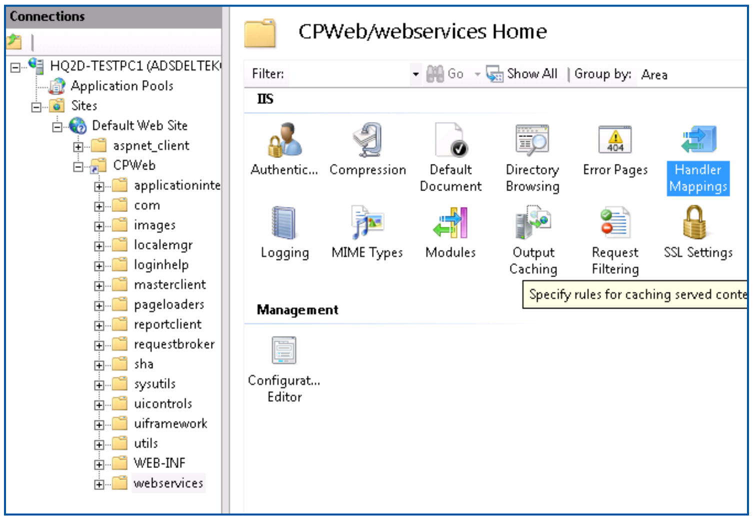
To configure IIS to proxy all Integration Modules:

1. Perform the following sub-steps if you want to expose Integration services on the entire Weblogic cluster (that is, you do not want to restrict the processing of Web services to specific WL nodes in your deployment).
   a. On the IIS server, locate the root folder of the DEWebApp application. By default, it is C:\deltek\costpoint\80\applications\DEWebApp.
   b. Create a new folder called **webservices** under C:\deltek\costpoint\80\applications\DEWebApp.
2. Perform the following sub-steps if you want to expose Integration services running on a specific Weblogic server or a group of nodes (for example, two dedicated servers).

   You can specify the list of nodes or use a virtual IP for this group.
   a. Create a new Application Pool with settings similar to that of DefaultAppPool.
   b. Create a new virtual Application in IIS mapping to an empty folder on the local drive (for example, C:\services\). Assign it to the application pool created in previous step.
   c. Create a subfolder called **webservices** under the local C:\services folder.
   d. Copy **DEProxy** folder from Costpoint Web Server (by default located under C:\deltek\costpoint\80\applications\) to the local C:\services\webservices folder.
   e. Using Notepad, create a file named **iisproxy.ini** under C:\services\webservices\DEProxy with following content.

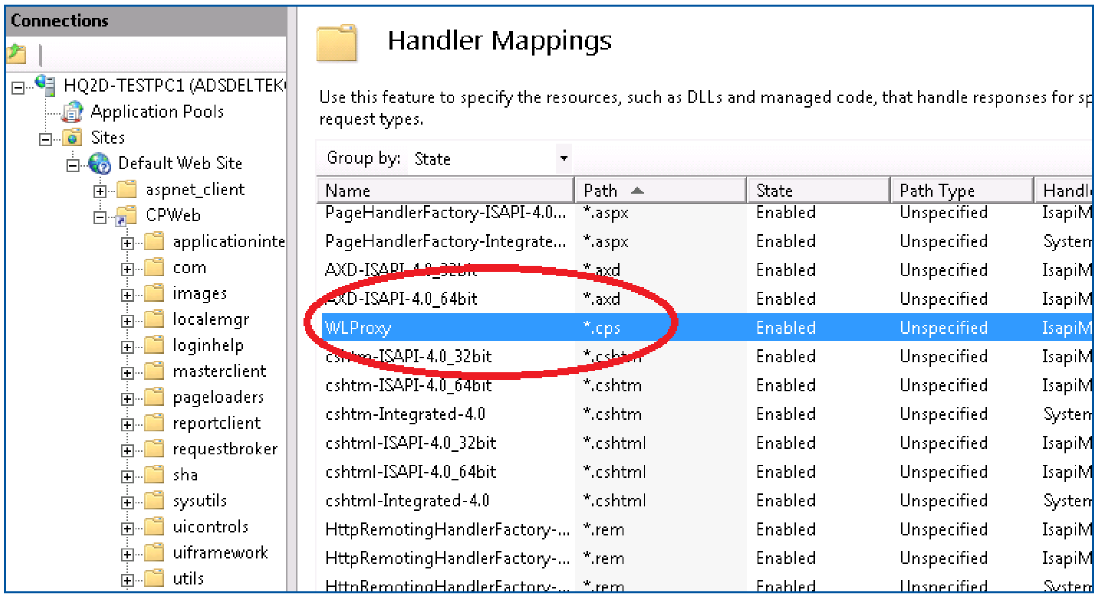
```
WLIOTimeoutSecs=3600
Idempotent=OFF
Debug=ALL
DebugConfigInfo=ON
WLLogFile=c:\temp\webservices_proxy.log
WLSocketTimeoutSecs=20
ConnectTimeoutSecs=80
ConnectRetrySecs=8
KeepAliveSecs=120
FileCaching=OFF
DynamicServerList=OFF
PathTrim=/CPWeb
```
(Replace **CPWeb** with the name of the new Virtual Directory in IIS.)
```
WebLogicCluster=<Dedicated-HostName1>:<Port1>,<Dedicated-HostName2>:<port2>
```
(Use a comma-separated list to include more than one dedicated server.)

3. Click **Control Panel » Administrative Tools » Internet Service Manager** to open the Internet Information Services (IIS) Console.
4. Drill down to access the newly created **webservices** folder.
5. Double-click the **Handler Mappings** icon.



6. Right-click the WLProxy mapping that was created by the Web tier installer, and click **Edit** on the shortcut menu.

   It should be mapping to *.cps.



Use this feature to specify the resources, such as DLLs and managed code, that handle responses for sp request types.

Group by: State

Name	Path	State	Path Type	Handl
PageHandlerFactory-ISAPI-4.0...	*.aspx	Enabled	Unspecified	IsapiM
PageHandlerFactory-Integrate...	*.aspx	Enabled	Unspecified	Systen
AXD-ISAPI-4.0_32bit	*.axd	Enabled	Unspecified	IsapiM
AXD-ISAPI-4.0_64bit	*.axd	Enabled	Unspecified	IsapiM
WLProxy	*.cps	Enabled	Unspecified	IsapiM
cshtm-ISAPI-4.0_32bit	*.cshtm	Enabled	Unspecified	IsapiM
cshtm-ISAPI-4.0_64bit	*.cshtm	Enabled	Unspecified	IsapiM
cshtm-Integrated-4.0	*.cshtm	Enabled	Unspecified	Systen
cshtml-ISAPI-4.0_32bit	*.cshtml	Enabled	Unspecified	IsapiM
cshtml-ISAPI-4.0_64bit	*.cshtml	Enabled	Unspecified	IsapiM
cshtml-Integrated-4.0	*.cshtml	Enabled	Unspecified	Systen
HttpRemotingHandlerFactory-...	*.rem	Enabled	Unspecified	IsapiM
HttpRemotingHandlerFactory-...	*.rem	Enabled	Unspecified	IsapiM
HttpRemotingHandlerFactory-...	*.rem	Enabled	Unspecified	Systen

7. Change the mapping to * and click **OK**.

   Do not modify any other fields on the dialog box.

8. Click **Yes** when prompted about allowing ISAPI extensions.

9. Restart IIS service for the changes to take effect.

   Verify that you can access webservices by querying the WSDL. A typical URI to access an integration module on IIS server is http://<servername>/CPWeb/webservices/<servicename>.

10. Test to ensure that you can access webservices through IIS.

    For example, if the module name is ldmtimews.jar, http://IISServer/CPWeb/webservices/LdmtimeWS?WSDL should display the WSDL of the Web service.

Note: This procedure will expose all integration modules via the IIS server. Alternatively, you can expose specific modules by creating subfolders under **webservices** mentioned in steps 1 and 2. The subfolder name should match the name of the Web service as per the URI to access it (for example, **webservice/<moduleName>**).

To configure IIS to proxy only selective Integration Modules

The above procedure will expose all integration modules through IIS server. Alternatively, you can choose to expose only a select set of integration services. For example, you may want to expose only the Timesheet Module (ldmtimews.jar) and restrict access to all others. To do so, you first need to identify the URL of the module that needs to be exposed through IIS. You can use the Test Integration Module screen in the Integration utility to identify the URL of any module. For example, if an integration module ID is LDMTIME, its application server URL would be http://AppServer:7009/webservices/LdmtimeWS?WSDL and its IIS URL would be http://IISServer/CPWeb/webservices/LdmtimeWS?WSDL. On the IIS server, you need to create subfolders that match the module URL and change the handler mappings of only those modules that need to be exposed through IIS. For ldmtimews.jar, the subfolder should be named ldmtimews.

Procedure continued…

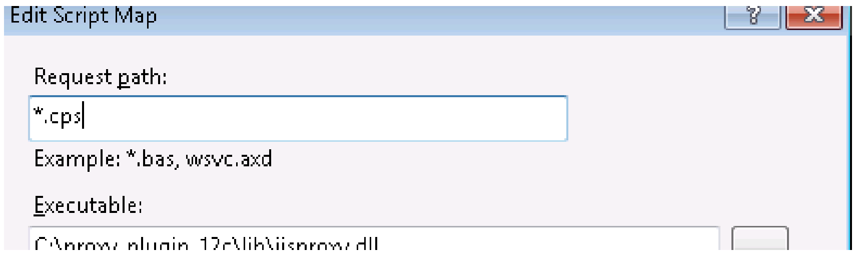
11. Open the webservices folder created in step 1 or 2 above and create a new subfolder under it.

    The name of the new folder should match the module name. For example if the module is ldmtimews.jar, the folder should be named LdmtimeWS.

12. Click **Control Panel » Administrative Tools » Internet Service Manager** to open the Internet Information Services (IIS) Console.

13. Navigate to previously created webservices folder and change the handler mapping WLProxy to proxy only the *.cps requests.

    Refer to Steps 3 through 8 above. This will stop proxying requests to all Integration modules.



Edit Script Map

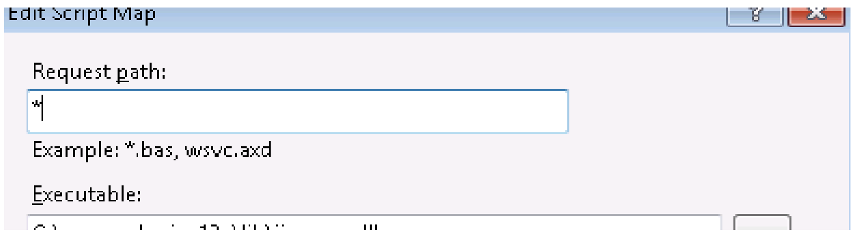
Request path: *.cps

Example: *.bas, wsvc.axd

Executable: C:\proxy_plugin_12c\lib\iisproxy.dll

14. Navigate to the newly created subfolder (LdmtimeWS) and change the handler mapping WLProxy to proxy * (all) requests.

    Refer to Steps 3 through 8 above. This will allow requests related to the module to be proxied.



Edit Script Map

Request path: *

Example: *.bas, wsvc.axd

Executable:

15. Repeat Steps 11-14 for all other integration modules to be exposed to IIS.

If you have a hardware or software load balancer in front of your IIS server(s) and you want to expose the integration modules through this load balancer, you will need to set up the Front End Host and Port(s) in Configuration Utility.

Attention: For more on this, please refer to the *Deltek Costpoint 8.0 Configuration Utility* guide.

SSL Communication

Costpoint integration modules support both one- and two-way SSL communication, but the servers they are deployed on must be SSL enabled. By default, the Costpoint application server is SSL-enabled with demo certificates. You need to replace the demo certificate with a valid verified certificate on the WebLogic server.

When the integration modules are exposed through IIS and you require SSL communication, you should configure the IIS server with its own SSL certificates and have the SSL port enabled on it.

Some companies require that SSL communication from the end user to IIS is proxied all the way to the WebLogic server.

Note: The IIS to WebLogic server proxy does not support two-way SSL communication. The WebLogic server should be accessed directly to participate in two-way SSL communication.

Appendix C: Invoking Generated Web Service from within Costpoint Extensibility Code

The following is an example of how you can invoke Web Services from within Costpoint itself. In this example, the WebService being built is for the report application. Because of this, the Java code receives a Costpoint-generated report in HTML format.

To invoke a generated Web Service from within Costpoint:

1. Build a Web Service that returns a report in the Integration Console.

   In this example, it is called **blrmbil**.

2. Copy the generated client jar (in this example, **blrmbilws_client.jar**) from **C:\deltek\costpoint\80\tmp\output\clientjar\blrmbilws\**, where it was created by the Integration Console, to **C:\deltek\costpoint\80\applications\enterprise\APP-INF\lib**, so it will be loaded by WebLogic as part of Costpoint application classpath.

3. Write your Java class.

   Note: See the Java code sample below. It uses InvokeWSFromExt class and registered as **After Save** Java plugin.

4. Compile it referencing the same client jar from step 2.

5. Assign this class as the Extensibility Java Plug-in event (in this example, it is After Save class) to the appropriate Costpoint Object in the Extensibility Console.

6. Deploy the Extensibility unit and assign it to the appropriate User.

Example of Java Code to Invoke Regular (Legacy) Web Service

```
package com.deltek.enterprise.extensions.xt_222;

import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import javax.xml.rpc.Stub;
//security stuff
import weblogic.xml.crypto.wss.provider.CredentialProvider;
import weblogic.xml.crypto.wss.WSSecurityContext;
import weblogic.wsee.security.unt.ClientUNTCredentialProvider;
import java.util.*;
import java.security.*;
import java.sql.SQLException;

//client jar
import com.deltek.enterprise.integration.ws.blrmbil.client.*;
import com.deltek.enterprise.system.applicationinterface.DEException;
import com.deltek.enterprise.system.applicationinterface.ResultSetInterface;
import com.deltek.enterprise.system.applicationinterface.RowSetInterface;
import com.deltek.enterprise.system.applicationinterface.AfterRSSave;
import com.deltek.enterprise.system.applicationinterface.RSIterator;
import com.deltek.enterprise.system.applicationinterface.SqlManager;
```

```
public class InvokeWSFromExt {

  public short afterRSSave(ResultSetInterface rsI) throws DEException,SQLException {

      try {

com.deltek.enterprise.system.serverapi.remoteapi.integration.HTMLReportResponse result
=

(com.deltek.enterprise.system.serverapi.remoteapi.integration.HTMLReportResponse) new
InvokeWSFromExt().invoke( );
         printResult( result );
         System.out.println( "Ok " );
      }
      catch (Exception e) {
        e.printStackTrace();
        return 2;
      }
      return 0;
  }

  public Object invoke(  ) throws Exception {
    try {
         //connection information to WL server and system
      String ip = "http://server1:7001";
      String methodName = "blrmbil";
      String systemName = "<system>";
      String userName = "<username>";
      String userPassword = "<password>";
      String companyId = "1";
      String useDefaultSystem = "N";
      BlrmbilService service = new BlrmbilService_Impl( ip +
"/webservices/BlrmbilWS?WSDL" );
      Blrmbilws port = service.getBlrmbilwsSoapPort();

      // create credential provider and set it to the Stub
      List credProviders = new ArrayList();
      // client side UsernameToken credential provider
      String systemUsername = userName + "__" + systemName;
      if (useDefaultSystem.equals( "Y" )) {
        systemUsername = userName;
      }
      CredentialProvider cp = new ClientUNTCredentialProvider( systemUsername,
userPassword );
      credProviders.add( cp );
      Stub stub = (Stub) port;
      stub._setProperty( WSSecurityContext.CREDENTIAL_PROVIDER_LIST, credProviders );
      String cdata="";
         //in my saved parameter id is EDS
      cdata = "<blrmbil xmlns='http://www.deltek.com/enterprise/integration/blrmbil'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'> " +
              "<BLRMBIL_WFUNCPARMCATLG_HDR > " +
              "<SAVED_PARM_ID>EDS</SAVED_PARM_ID> " +
              "</BLRMBIL_WFUNCPARMCATLG_HDR> " +
              "</blrmbil>";

      if (methodName.equals( "blrmbil" )) {
        return port.blrmbil( systemName, companyId, cdata );
      }
      System.err.println( "Method not found: " + methodName );
      return null;
    }
    catch (Exception ex) {
      ex.printStackTrace();
      if (ex.getCause() != null
              && ex.getCause().getClass().getName().equals(
"javax.xml.rpc.soap.SOAPFaultException" )) {
        throw (Exception) ex.getCause();
      }
      throw ex;
    }

  }

  private static void printResult( Object result ) {
    if (result instanceof String) {
      String strResult = (String) result;
      System.out.println( "Data: " + strResult );
    }
    else if (result.getClass().isArray()) {
      byte[] binaryResult = (byte[]) result;
      String strResult = null;
      if (false) {
        try {
          // System.out.println("Archived data: " + strResult);
          strResult = extractData( binaryResult );
        }
        catch (Exception e) {
          e.printStackTrace();
        }
      }
      System.out.println( "Data: " + strResult );
    }
    else if (result == null) {
      System.out.println( "Data: " + result );
    }
    else {
      try {
        System.out.println( "Data: " );
        System.out.println( "<!--object represented as XML string-->" );
        print( result );
      }
      catch (Exception e) {
        e.printStackTrace();
      }
    }
  }

  private static final int BUFFER_LENGTH = 65536;

  /**
   * Converts zipped data into clear text
   */
  public static String extractData( byte[] zippedBytes ) throws Exception {
    // unzip data using standard java Inflater class
    java.util.zip.Inflater decompresser = new java.util.zip.Inflater( true );
    decompresser.setInput( zippedBytes, 0, zippedBytes.length );
    byte[] result = new byte[BUFFER_LENGTH];
    StringBuilder out = new StringBuilder();
    int resultLength = -1;
    while ((resultLength = decompresser.inflate( result )) > 0) {
      out.append( new String( result, 0, resultLength, "UTF-8" ) );
    }
    decompresser.end();
    return out.toString();
  }

  /**
   * Converts clear text into hexed zipped data
   */
  public static byte[] archiveData( String sourceText ) throws Exception {
    // Encode a String into bytes
    byte[] input = sourceText.getBytes( "UTF-8" );
    // Compress bytes
    java.io.ByteArrayOutputStream binaryData = new java.io.ByteArrayOutputStream();
    java.util.zip.Deflater compresser =
            new java.util.zip.Deflater(
                    java.util.zip.Deflater.DEFAULT_COMPRESSION, true );
    compresser.setInput( input );
    compresser.finish();
    int compressedDataLength = -1;
    byte[] output = new byte[BUFFER_LENGTH];
    while ((compressedDataLength = compresser.deflate( output )) > 0) {
      binaryData.write( output, 0, compressedDataLength );
    }
    return binaryData.toByteArray();
  }

  /**
   * Converts date in the jdbc string format into the instance of Calendar class
   */
  public static java.util.Calendar getDate( String strDate ) throws Exception {
    Calendar cal = null;
    if (strDate == null) {
      return null;
    }
    cal = java.util.Calendar.getInstance();
    cal.setLenient( false );
    try {
      cal.setTime( java.sql.Timestamp.valueOf( strDate ) );
    }
    catch (Exception e) {
      System.err.println( "Invalid date :" + strDate );
      throw e;
    }
    return cal;
  }

  // prints data into console in stream
  public static void print( Object obj ) throws Exception {
    // printField(obj,classToFieldName(obj.getClass().getName()));
    String name = classToFieldName( obj.getClass().getName() );
    l( "<" + name + ">" );
    if (obj == null) {
      return;
    }
    java.lang.reflect.Method[] methods = obj.getClass().getMethods();
    // print fields
    for (java.lang.reflect.Method m : methods) {
      if (m.getName().startsWith( "get" ) && !m.getName().equals( "getClass" )) {

        if (!m.getReturnType().isArray()) {
          printField( m.invoke( obj ), getFieldName( m.getName() ) );
        }
      }
    }
    // print arrays
    for (java.lang.reflect.Method m : methods) {
      if (m.getName().startsWith( "get" ) && !m.getName().equals( "getClass" )) {

        if (m.getReturnType().isArray()) {
          printChildren( m.invoke( obj ), getFieldName( m.getName() ) );
        }
      }
    }
    l( "</" + name + ">" );
  }

  private static String classToFieldName( String className ) {
    int i = className.lastIndexOf( '.' );
    className = className.substring( i + 1 );
    i = className.lastIndexOf( '@' );
    if (i != -1) {
      return className.substring( 0, i );
    }
    else {
      return className;
    }
  }

  private static void printField( Object field, String fieldName ) throws Exception {
    if (field != null) {
      p( "<" + fieldName + ">" );
      if (field instanceof java.lang.String) {
        p( field + "" );
      }
      else if (field instanceof java.math.BigDecimal) {
        p( javaValueToString( field ) );
      }
      else if (field instanceof java.util.Calendar) {
        p( javaValueToString( field ) );
      }
      else {
        p( field + "" );
      }
      l( "</" + fieldName + ">" );
    }
  }

  private static String getFieldName( String getMethod ) {
    return getMethod.substring( 3 );
  }

  private static void printChildren( Object array, String fieldName ) throws Exception
{
    if (array == null) {
      return;
    }
    // l("<"+fieldName+">");

    int length = java.lang.reflect.Array.getLength( array );
    for (int i = 0; i < length; i++) {
      Object obj = java.lang.reflect.Array.get( array, i );
      if (obj.getClass().isArray()) {
        printChildren( obj, classToFieldName( obj.getClass().getName() ) );
      }
      else {
        print( obj );
      }
    }
    // l("</"+fieldName+">");

  }

  private static void l( String text ) {
    p( text + "\r\n" );
  }

  private static void p( String text ) {
    System.out.print( text );
  }

  private static String javaValueToString( Object value ) throws Exception {
    if (value instanceof String) {
      return (String) value;
    }
    else if (value instanceof java.math.BigDecimal) {
      return zeroKiller( ((java.math.BigDecimal) value).toString() );
    }
    else if (value instanceof java.util.Calendar) {
      return (new java.sql.Timestamp( ((java.util.Calendar) value).getTime().getTime()
)).toString().replace(
              " ", "T" );
    }
    // some kind of combobox probably ...
    else {
      java.lang.reflect.Method m = value.getClass().getMethod( "getValue" );
      if (m != null) {
        Object strValue = m.invoke( value );
        return (String) strValue;
      }
      // out of guesses :(
      else {
        return value + "";
      }
    }

  }

  private static java.text.NumberFormat numFormatter =
java.text.NumberFormat.getInstance();

  private static String zeroKiller( String s ) {
    if (s == null) {
      return s;
    }
    int k = 0;
    int m = 0;
    k = s.indexOf( '.' );
    if (k != -1) {
      m = s.length();
      StringBuffer convert = new StringBuffer( s );
      while (s.charAt( --m ) == '0') {
        convert.deleteCharAt( m );
      }
```

```
        if (m == k) {
          convert.deleteCharAt( m );
        }
        return convert.toString();
      }
      else {
        if (s.indexOf( 'E' ) != -1) {
          try {
            return numFormatter.parse( s ).toString();
          }
          catch (java.text.ParseException e) {

            return s;
          }
        }
        else {
          return s;
        }
      }
    }

    public static String readFile( String filename ) {
      // for backward compatibility in case user passes real data
      if (!filename.endsWith( ".xml" ))
        return filename;
      StringBuffer Str = new StringBuffer();
      String s = null;
      java.io.BufferedReader file;
      try {
        file = new java.io.BufferedReader( new java.io.FileReader( filename ) );
        s = file.readLine();
        while (s != null) {
          Str.append( s + "\r\n" );
          s = file.readLine();
        }
        file.close();
      }
      catch (Exception e) {
        System.out.print( e );
      }
      return Str.toString();
    }
  }
```

Example of Java Code to Invoke Generic Web Service

Here is another example of how to invoke Generic Web Service from the Extensibility code

```
package com.deltek.enterprise.extensions.xt_222;

import java.io.StringReader;
import java.io.UnsupportedEncodingException;

import javax.xml.namespace.QName;
import javax.xml.transform.Source;
import javax.xml.transform.stream.StreamSource;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Service;
import com.deltek.enterprise.system.utils.IntegrationUtils;
import com.deltek.enterprise.system.utils.IntegrationWSHandlerResolver;
import com.deltek.enterprise.system.utils.IntegrationWSProps;

public class GenericModuleTester {

  public GenericModuleTester() {
  }

  public static void main( String[] args ) throws Exception {
    new GenericModuleTester().invoke( args );
  }

  public void invoke( String[] args ) throws Exception {
    try {
      String ip = args[0];
      String methodName = args[1];
      String systemName = args[2];
      String userName = args[3];
      String userPassword = args[4];
      String companyId = args[5];
      String useDefaultSystem = args[6];
      String systemUsername = userName + "__" + systemName;
      if (useDefaultSystem.equals( "Y" )) {
        systemUsername = userName;
      }
      String serviceName = (ip.startsWith( "https" ) ? "cpwwsgenericmodulesec" :
"cpwwsgenericmodule");
      Dispatch<Source> dispatcher = getWSDispatcher( ip + "/webservices/" +
serviceName + "WS?WSDL",
              serviceName, systemUsername, userPassword );

      StringBuilder b = new StringBuilder();
      b.append( "<cps:" + methodName + "
xmlns:cps='http://www.deltek.com/enterprise/integration/ws/"
              + serviceName.toLowerCase() + "'><cps:systemName>" + systemName
              + "</cps:systemName><cps:companyId>" + companyId +
"</cps:companyId><cps:"
              + (IntegrationUtils.GENERIC_WS_METHOD_EXPORT.equals( methodName ) ?
"queryCondition"
                      : "document")
              + ">" );

      b.append( IntegrationUtils.toXMLString( readFile( args[7] ) ) + "</cps:"
              + (IntegrationUtils.GENERIC_WS_METHOD_EXPORT.equals( methodName ) ?
"queryCondition"
                      : "document")
              + "></cps:" + methodName + ">" );

      Source request = new StreamSource( new StringReader( b.toString() ) );
      Source response = dispatcher.invoke( request );
      String result = IntegrationUtils.sourceToXMLString( response );

      printResult( result );
    }
    catch (Exception ex) {
      if (ex.getCause() != null
              && ex.getCause().getClass().getName().equals(
"javax.xml.rpc.soap.SOAPFaultException" )) {
        throw (Exception) ex.getCause();
      }
      throw ex;
    }

  }

  private Dispatch<Source> getWSDispatcher( String url, String serviceName, String
user, String password ) {
    QName portQName = null;
    Service service = null;

    portQName = new QName( "http://www.deltek.com/enterprise/integration/ws/" +
serviceName.toLowerCase(),
            serviceName + "ServiceSoapBinding" );
    service = Service.create(
            new QName( "http://www.deltek.com/enterprise/integration/ws/" +
serviceName.toLowerCase(),
                    serviceName + "Service" ) );
    service.addPort( portQName, javax.xml.ws.soap.SOAPBinding.SOAP11HTTP_BINDING, url
);
    IntegrationWSHandlerResolver handlerResolver = null;
    try {
      IntegrationWSProps props = new IntegrationWSProps();
      props.setPass( password == null ? new byte[0] : password.getBytes( "UTF-8" ) );
      props.setUser( user );
      props.setExpireMinutes( 60 * 24 * 7 );// one week
      props.setServiceName( serviceName + "Service" );
      props.setClearText( true );
      handlerResolver = new IntegrationWSHandlerResolver( props );
    }
    catch (UnsupportedEncodingException e) {
    }
    service.setHandlerResolver( handlerResolver );
    return service.createDispatch( portQName, Source.class, Service.Mode.PAYLOAD );
  }

  private static void printResult( Object result ) {
    if (result instanceof String) {
      String strResult = (String) result;
      System.out.println( "Data: " + IntegrationUtils.toJavaString( strResult
).replace( "><", ">\n<" ) );
    }
    else if (result == null) {
      System.out.println( "Data: " + result );
    }
  }

  public static String readFile( String filename ) {
    // for backward compatibility in case user passes real data
    if (!filename.endsWith( ".xml" ))
      return filename;
    StringBuffer Str = new StringBuffer();
    String s = null;
    java.io.BufferedReader file;
    try {
      file = new java.io.BufferedReader( new java.io.FileReader( filename ) );
      s = file.readLine();
      while (s != null) {
        Str.append( s );
        s = file.readLine();
      }
      file.close();
    }
    catch (Exception e) {
      System.out.print( e );
    }
    return Str.toString();
  }

}
```

Appendix D: Troubleshooting Web Services

This appendix discusses common types of issues you may encounter while testing Integration Web services.

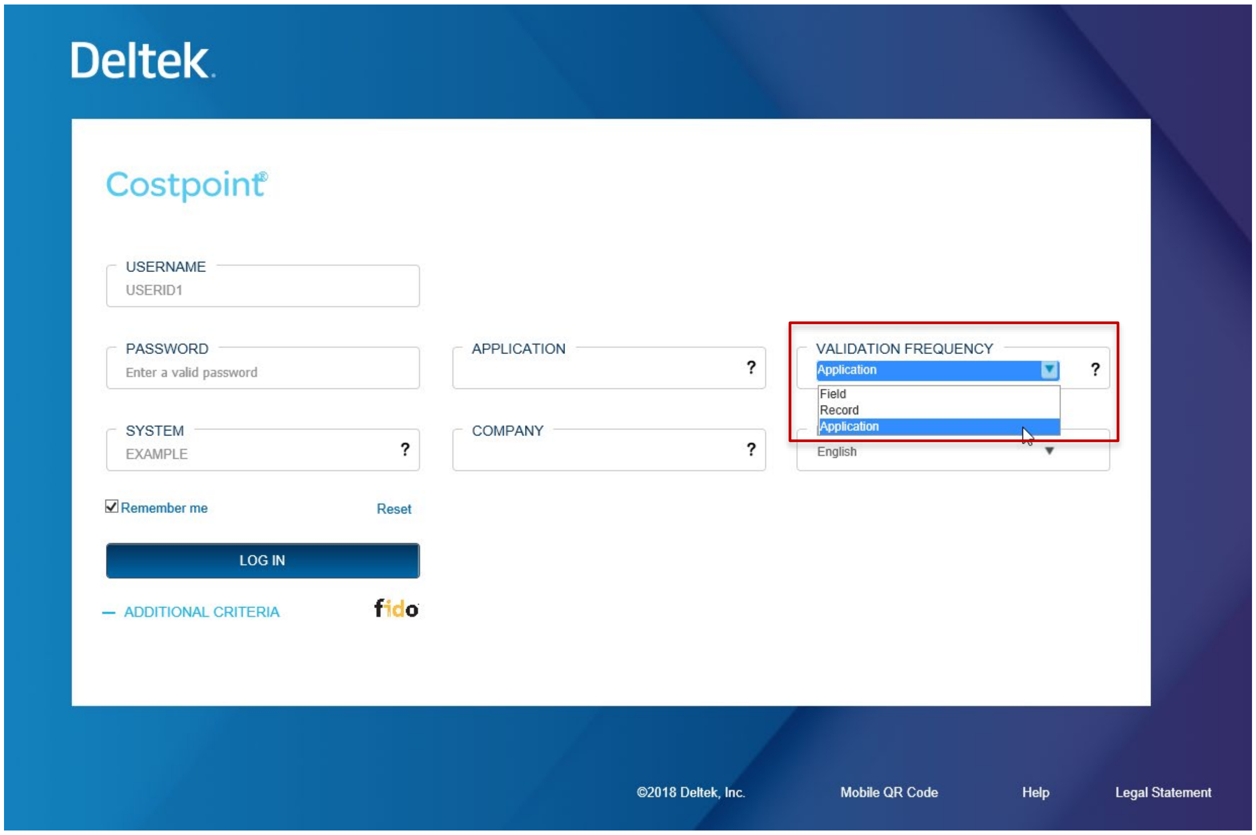
Problem: "I can successfully do what I need through browser end-user UI, but the same operation with same data returns a validation error through Web service."

This is usually caused by one of two reasons:

- There is an oversight in the data provided to Web service (that is, the data provided between browser UI and web-service are not the same).
- There is a defect in the application related to the timing and order of when different validations are performed. When you provide data through the browser UI, the order of the provided fields and performed validations may differ from the order used in Web-service mode when the XML file with all the data is provided at once. There could be defects which manifest themselves only with the order is used by Web service

Regardless of the reason, you should use following process to narrow down the issue and expedite processing the case through Deltek Customer Care if you ultimately decide to open a case.

1. Log into Costpoint using the **Application Frequency** mode.



2. Open the application used in Web service.

3. Copy/paste the data from the XML file into the corresponding fields in the Costpoint application.
   - Make sure that you do not enter or change any other fields which are not listed in the XML file.
   - Make sure that you do not click any action buttons within an application, such as **Load**, **Autopopulate**, **Default**, **Recompute**, **Allocate**, and so on.

   The only buttons you can use are from the standard screen menu:
     - **Query**: Click this button if you are updating/deleting existing an record in the XML file.
     - **New**: Click this button if you are passing new rows in the XML file.
     - **Delete**: Click this button if you are deleting rows in the XML file.
4. Click the **Save** button to save your changes

In the majority of cases, you will be able to replicate the same validation error that you received through Web service. At this point, you can further analyze if the error message is correct based on your setup and data provided, or you believe it's a defect. In case of a defect, you can submit a case based on using the browser UI instead of Web Service, which will greatly simplify and accelerate the replication process for the Deltek Customer Care and Engineering teams.

To further troubleshoot issues with Web Services that import data, you should turn on **Log Debug Information for WS Calls** in the Configuration Utility. Be sure to restart Weblogic when done. When submitting a case to Deltek Customer Care, include:

- This log (described above) if it is importing WS
- Exported Web service definition (jar)
- The exact XML file you are submitting to Web Service
- The user name (that is invoking Web Service)
- The system name
- The exact date/time when WS was invoked
- The exact XML file that was received in response to the WS invocation
- The newly generated diagnostic image (please generate it after WS invocation was finished)

Problem: "I can't find a way to do through Web services what I can do through the browser UI."

The most likely scenario in this case is that you are using an application-specific "convenience" action—such as **Load**, **Autopopulate**, **Default**, **Recompute**, **Allocate**, and so on. In the browser UI, you can click **New** and manually enter all the values in the fields, or you can use a combination of manually entered data and data entered via a convenience button such as **Load**.

Such user-convenience functions as **Load** cannot be used in Web services. The Web services XML file should contain the data of all the fields as if you manually clicked **New** and then entered all the appropriate values without ever clicking **Load**.

If an application allows you to insert all rows manually, then you should follow the same instructions as provided in the previous section for testing the use-case in **Application Frequency** mode. If it works, then the same data and operations need to be passed through the XML file as well.

There could be cases when applications do not provide you the ability to perform a necessary operation without the use of the user-convenience buttons. This is an oversight in application design. Such cases should be considered defects and need to be submitted to the Deltek Customer Care team.

Problem: "My Web service worked well in the past, but after I upgraded to 711 and/or deployed new fixes it no longer works."

The first thing to do is to go to the Integration Console, open your Web service, and check if there are any new columns in the screens (result sets) involved in the failing Web service. The most likely explanation is that new fields were added as part of an enhancement or fix to the application in question, and some of the fields are either required or critical to the business logic of the application. When using the application through the browser UI, the fields are likely properly defaulted, but Web service is missing to pass the proper values, which could be causing the error.

Again, if ever in doubt why Web service does work, use the same troubleshooting technique as previously discussed by using **Application Frequency** mode.

About Deltek

Better software means better projects. Deltek delivers software and information solutions that enable superior levels of project intelligence, management and collaboration. Our industry-focused expertise makes your projects successful and helps you achieve performance that maximizes productivity and revenue. www.deltek.com