Oracle® Hyperion Financial Close Management

Integration Guide

Table of Contents

Introduction 1
Overview 1
Terminology 1
Integration Concepts 2
Integration Types 2
Integration Type Parameters 3
Parameter Attributes 3
Parameter Types 4
Dynamic List Parameter Type 5
Document Navigator Parameter Type 6
Task Execution Detail 7
Task Execution Endpoints 7
Task Security 8
Asynchronous Web Services 9
Task Response Handling 9
System Automated Task Response XSD 10
Dynamic List Parameter Type Response XSD 10
Document Navigator Parameter Type Response XSD 11
Creating Integrations 13
Creating Integrations using the Integration XML Document 13
Integration Type XML Schema 13
Integration XML Elements and Attributes 17
Loading the Integration Type XML 22
Creating Integrations with the Financial Close User Interface 22
Sample Integration XML 23

Introduction

Overview

The Financial Close product provides a flexible integration framework that allows an end user to leverage services from external applications as part of the close calendar. The integration framework is built around industry standards and supports web based interactive tasks and web service based automated tasks.

The Financial Close integration framework supports the following:

- Integration of third party end user tasks into the close calendar
- Integration of third party system automated tasks into the close calendar
- Interactive task parameter gathering and usage for integrated tasks
- Task parameter dependencies
- Runtime retrieval and listing for task parameter values from third party systems
- Asynchronous handling of system automated tasks
- Standardized Web Services security (WS-Security)
- Runtime creation and editing of task integrations

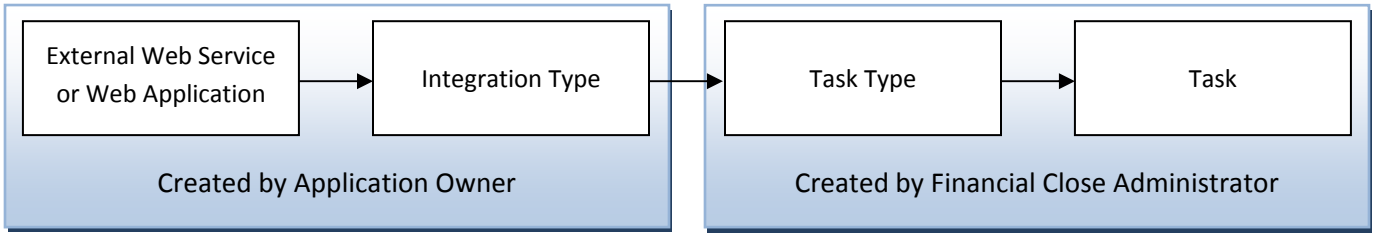
Terminology

Task	The Task is the core unit in the close calendar. For example, a task could be a data load, data entry, a meeting, or a simple notification.
Integration Type	An Integration Type is a reusable definition that describes an action and set of parameters that will be used to call an external application's service or interface. The user can create their own Integration Types that integrate with their own proprietary systems.
Task Type	Task Types are templates that tasks will be created from. Each Task Type has a name, description and optionally, an Integration Type associated with it. The user can specify instructions, questions, attributes, and parameter values that will be defaulted to any task created from this particular template. All tasks must be associated with a Task Type.

Integration Concepts

To integrate an external application service within the Financial Close close calendar, three basic steps are required:

1. Define the service to be integrated as an Integration Type in Financial Close.
2. Create a Task Type that wraps the integration for usage in the close calendar.
3. Add a Task based on the Task Type to the close calendar.



The owner of the application being integrated will be creating the Integration Types for the Financial Close integration. The Integration Type is the only place where knowledge of the external application service is required.

The Task Type and Tasks are not directly related to the external application service definition. The Task Type and Tasks are specific Financial Close concepts and those elements should be created by the Financial Close administrator (or another Financial Close user with the required permissions).

Integration Types

There are two types of tasks that can be integrated within the close calendar:

- **End User Tasks -** an end user task is a task that requires the user to interact with a user interface to complete the task. Examples of this type of a task would be data entry, journals updates, etc. To integrate an end user task with Financial Close, the task must be exposed as a directly callable web application URL endpoint.

- **System Automated Tasks** – a system automated task is a task that runs in the background without any user interaction required. To integrate a system automated task with Financial Close, the task must be exposed as an asynchronous web service. See the *Asynchronous Web Services* section for more detail.

For each end user or system automated task to be used in the Financial Close calendar, an Integration Type must be created within Financial Close. An Integration Type is the definition of the third party service and should include the following properties:

- **Execution Type** – defines the type of integration, end user or system automated.

- **Task Parameters** – defines the parameters needed to invoke the end user or system automated task. For example, if the integrated service will run a report, the user may need to specify which report to run. The values for these parameters may be defined in an associated Task Type or may be collected at run time when a Task is added to the close calendar.

- **Execution Detail –** when a Task is actually run by the close calendar, Financial Close will use the execution information to invoke the task. For end user tasks, the execution detail would include the URL needed to invoke the third party web module. For system automated tasks, the execution detail would include the web service endpoint needed to invoke the service.

- **Response Detail –** when a system automated task is finished, the integrated service may respond with information indicating success or failure and detail on the task execution. The Integration Type should include the information needed to process the response.

Integration Type Parameters

The parameters of an Integration Type represent the values that must be sent to the service provider when Financial Close invokes the service. For system automated tasks, the parameter values will be sent as part of the SOAP message. For end user tasks, the parameter values will be sent as query parameters on the URL used to display the third party web module.

Parameter Attributes

Each parameter defined for an Integration Type will have the following attributes:

- **Parameter Name** – the display name of the parameter. The parameter name will be displayed next to the parameter when the Integration Type is used to create a Task in Financial Close. This value can be localized.

- **Parameter Code** – the internal name used for the parameter. This value will be used to generate the service request or URL when the task is processed. For example, the URL that is built for end user tasks will include a code=value pair for each parameter. The value for a parameter code must be alpha-numeric.

- **Required Flag –** this flag indicates whether or not a parameter value is required to execute the task.

- **Parameter Order –** a numeric value indicating the order to display the parameters in the Financial Close user interface.

- **Tooltip** – descriptive text about the parameter and its usage. This value will be displayed if the user hovers over the parameter in the Financial Close user interface. This value can be localized.

- **Dependencies** – a list of parameters that this parameter is dependent on. If a parameter is dependent on another parameter (its parent), the parameter will be disabled in the Financial Close user interface until the parent parameter has a value. The order attribute on a dependant parameter must be greater than the order of its parent.

Parameter Types

In addition to the above attributes, Financial Close needs to know the types of parameters being used. The following parameter types are understood by Financial Close:

- **Text** – this type is used for a free form text value. When creating a parameter of this type, the maximum field length and maximum number of text rows can be specified.

- **Number / Integer** – this type is used for a basic numeric value. When creating a parameter of this type, the lower and upper bounds of the number can be specified.

- **Date** – this type is used for a date value. When creating a parameter of this type, a date range (lower and upper bounds) can be specified. In the Financial Close user interface, the end user will have the ability to use a date picker control.

- **Checkbox** – this type is used for a boolean value. In the Financial Close user interface, this will be displayed as a checkbox control.

- **Static List** – this type is used for a pre-defined set of text values. The end user will be able to choose from the set of values in the Financial Close user interface.

- **Dynamic List** – this type is used for a dynamic set of text values. The list of valid values is determined at runtime by Financial Close. See the *Dynamic List Parameter Type* section for more information.

- **Options Group** – this type is used for a pre-defined set of values. The end user will be able to choose multiple values from the set in the Financial Close user interface.

- **Document Navigator** – this type is used for a hierarchical set of values (i.e. folders and documents). The hierarchy of valid values is determined at runtime by Financial Close. See the *Document Navigator Type* section for more information.

Dynamic List Parameter Type

In Financial Close, a dynamic parameter type is a parameter whose valid values can be determined at runtime. The Dynamic List Parameter type represents a flat set of values that has an associated web service on the integrated application to provide the valid list of values.

For example, suppose a task requires the name of a report and the list of valid reports can be retrieved from the integrated application by calling a web service. If the report name parameter is defined as a Dynamic List, Financial Close will use the application's web service to retrieve the list of values for the user to pick from. Instead of having to manually enter the name of a report, the end user will have a valid list of reports to select from.

To provide this functionality, Financial Close needs to know how to call the appropriate web service. The following attributes are required for each Dynamic List Parameter Type:

- **WSDL Location** – the URL for the web service's WSDL (i.e. http://rptsvr/ReportService?WSDL).

- **Namespace** – the namespace of the web service (i.e. http://reports.xyz.com).

- **Port** – the web service port type, as defined in the WSDL.

- **Name –** the web service name, as defined in the WSDL.

- **SOAP Action –** the SOAP action for the service.

- **Response Element –** the name of the SOAP response XML element that contains the parameter value.

- **Response Transform** – since the response may be returned in a form that isn't understood by Financial Close, the parameter definition can include an XSLT string to transform the response XML into a valid form. See the *Task Response Handling* section for more detail.

Document Navigator Parameter Type

The Document Navigator parameter type is similar to the Dynamic List parameter type in that the valid set of values can be retrieved from the integrated application at runtime. However, the Document Navigator represents a hierarchical set of values instead of a flat list. This parameter type can be used to represent documents and their folders.

The Document Navigator parameter type requires the same attributes as the Dynamic List to define the associated web service. In addition to those attributes there are some attributes specific to this type:

- **Parameter Path Code –** since the Document Navigator is used to list a hierarchical set of values, the integrated application may need Financial Close to send the full path to an item when setting the value of the parameter. For example, the integrated application may need "/FolderABC/FileXYZ" or it may only need "FileXYZ" as the parameter value.

- **One Time Query Flag –** if the web service returns the entire hierarchy in a single call, set this flag to "Y". If the web service returns the hierarchy one level at a time, set this flag to "N". This allows Financial Close to use web services that support drilling into the hierarchy one level at a time.

- **Parent Folder Parameter –** if the web service is not a "one time query" service (One Time Query = Y), Financial Close will send the parent folder to the web service to provide the hierarchical context on each call. The Parent Folder parameter is the web service parameter where Financial Close should place the parent folder name.

- **Item Type Web Service –** if the items returned by the associated web service are segregated into types, Financial Close will support an additional web service call to list out the valid item types. For example, suppose the associated web service returns multiple document types (HTML, text, XML). If the integration needs only text documents, Financial Close can list the document types and allow the end user to filter the displayed hierarchy. The Item Type web service detail is the definition of the web service that will return the known item types (i.e. WSDL location, port type, etc).

- **Item Type Filter Parameter** – if the item types can be filtered, the item type to filter on filter needs to be sent to the Document Navigator parameter's web service. The Hierarchy Filter parameter is the parameter where Financial Close should place the item type to filter on.

Task Execution Detail

Task Execution Endpoints

- **End User Task Execution**

  For end user tasks, Financial Close needs a directly callable web application URL. The specified URL must be able to display the user interface associated end user task. The format of the URL should be:

  {protocol}://{server}:{port}/{context}/{page}?{param1}=$PARAM1$&{param2}=$PARAM2$...

  For example:

  http://rptsvr:80/reportingapp/viewReport.jsp?reportName=$PARAM1$

  The value for $PARAM1$, $PARAM$2, etc. will be filled in by Financial Close with the parameter codes that match based on the Integration Type definition. In the above example, there should be a "reportName" parameter code defined in the Integration Type. Financial Close will use the value of the "reportName" parameter in place of $PARAM1$.

- **System Automated Task Execution**

  For system automated tasks, Financial Close needs the endpoint information for the associated web service. The endpoint should be specified in the Integration Type's WSDL location element. The format should be:

  {protocol}://{server}:{port}/{context}/{service}?WSDL

  For example:

  http://rptsvr:80/reportingService/ReportService?WSDL

  In addition to the WSDL location, Financial Close needs the Integration Type to describe the web service detail. The following information is required:

  - Service Name – the name of the service being called. The service name is the name of the web service defined in the WSDL (i.e. <wsdl:definitions name="*Service Name*">).

- Service Namespace – the namespace associated with the web service. The namespace is defined in the WSDL (i.e. <wsdl:definitions name="*Service Name"* targetNamespace="*Service Namespace"*>).

- Service Operation (or SOAP Action) – the service operation is the name of the method to call on the web service. The service operation is defined in the WSDL (i.e. <wsdl:operation name="*Operation Name*">).

- Service Port Type – the service port type is the port definition for the web service. The port type is defined in the WSDL (i.e. <wsdl:portType name="*Service Port Type*">).

Task Security

- **System Automated Task Security**

  Financial Close relies on the WS-Security standard to establish the identity of the user when making web service calls. The services integrating with Financial Close will need to support WS-Security and Security Assertion Markup Language (SAML) based tokens for authentication.

  With SAML tokens, a trust relationship is established between the Financial Close client and external service provider by means of exchanging certificates. This allows Financial Close to invoke the system automated task using the task assignee's credentials.

  The Financial Close web service client uses Oracle's Web Service Manager (OWSM) to enforce the security policy. The policy in place is OWSM's "oracle/wss11_saml_token_with_message_protection_client_policy".

  This configuration requires that the Financial Close client's public key has been imported into the external service provider's keystore and that the external service provider's public key has been imported into the Financial Close client's keystore.

  For information on configuring the OWSM keystore and credential store, see the Configuring the Credential Store Using WLST in the Using Oracle Web Service Security Policies on the Oracle Technology Network.

- **End User Task Security**

  Financial Close will support single sign-on (SSO) with external web applications if that application is integrated with the Oracle EPM system's CSS SSO framework. When invoking an end user task, Financial Close will include the CSS SSO Token in the task URL if the task includes the SSO_TOKEN

replacement parameter. The external application can pick up the CSS SSO Token and use it to launch the end user task.

If the external web application does not support this integration, no user credentials will be provided when invoking the end user task URL. It would be up to the external application to prompt for credentials before displaying the end user task.

Asynchronous Web Services

Financial Close expects that all system automated tasks support asynchronous invocation.

To be usable by Financial Close, the web service must have two port types. Each port type performs a one-way operation. One port responds to the web service request and the second calls back into Financial Close with the response.

In addition to the two port WSDL, Financial Close requires that the web service accepts WS-Addressing based reply information. The WS-Addressing headers should be used by the web service to direct responses to the correct callback service.

To learn more about the asynchronous web services framework, see the Developing Asynchronous Web Services section in the Oracle Fusion Middleware Concepts Guide on the Oracle Technology Network.

Task Response Handling

There are three types of web services that Financial Close will be using from an external application:

1. The web service that executes a System Automated Task.
2. The web service that provides a list of values for the Dynamic List parameter type.
3. The web service that provides a hierarchical list of values for the Document Navigator parameter type.

When invoked, each of the services responds with an XML based SOAP message. The data types returned may be different based on the application being integrated. Financial Close provides a mechanism to translate the response XML into something understandable by Financial Close.

For the web services defined in an Integration Type, an XSLT string can be specified to take the actual web service response and translate it into the response expected by Financial Close.

The following sections describe the response XML required for each of the web service types.

System Automated Task Response XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:simpleType name="resultValue">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Success"/>
      <xs:enumeration value="Failure"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="TaskResult">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Result"/>
        <xs:element ref="LogLocation"/>
        <xs:element ref="Messages"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Result" type="resultValue"/>
  <xs:element name="LogLocation" type="xs:anyURI"/>
  <xs:element name="Messages">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Message"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Message" type="xs:string"/>
</xs:schema>
```

Example:

```
<TaskResult>
    <Result>Success</Result>
    <LogLocation>http://RptSvr1/ReportApp/ViewLog.jsp?LogID=1234</LogLocation>
    <Messages>
        <Message>The report ran successfully.</Message>
    </Messages>
</TaskResult>
```

Dynamic List Parameter Type Response XSD

```
<xs:element name="selectItem" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="code" type="xs:string" minOccurs=”1” maxOccurs=”1”/>
      <xs:element name="name" type="xs:string" minOccurs=”1” maxOccurs=”1”/>
```

```
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<selectItem>
  <code>SEL1</code>
  <name>Selection Item 1</name>
</selectItem>
<selectItem>
  <code>SEL2</code>
  <name>Selection Item 2</name>
</selectItem>
<selectItem>
  <code>SEL3</code>
  <name>Selection Item 3</name>
</selectItem>
```

Document Navigator Parameter Type Response XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="listing">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="folders"/>
        <xs:element ref="documents"/>
      </xs:sequence>
      <xs:attribute name="folder" use="optional" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="folders">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="folder"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="folder">
    <xs:complexType>
      <xs:attribute name="description" use="optional"/>
      <xs:attribute name="id" use="required" type="xs:NCName"/>
      <xs:attribute name="name" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="documents">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element maxOccurs="unbounded" ref="document"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="document">
    <xs:complexType>
      <xs:attribute name="description" use="optional"/>
      <xs:attribute name="id" use="required" type="xs:NCName"/>
      <xs:attribute name="name" use="required"/>
      <xs:attribute name="type" use="optional" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example:

```
<listing>
  <folders>
    <folder id=”folder1” name=”Folder 1” description=”Folder 1 Description”/>
    <folder id=”folder2” name=”Folder 2” description=”Folder 1 Description”/>
  </folders>
</listing>

<listing folder=”folder1”>
  <folders>
    <folder id=”folder3” name=”Folder 3” description=”Folder 3 Description”/>
  </folders>
  <documents>
    <document id=”doc1” type=”1” name=”Document 1” description=”Doc 1 description”/>
    <document id=”doc2” type=”5” name=”Document 2”/>
    <document id=”doc3” type=”5” name=”Document 3”/>
  </documents>
</listing>
```

Creating Integrations

There are two methods for creating Integration Types within Financial Close.

If there are a small number of services to integrate into Financial Close, the user can add and edit these integrations directly in the Financial Close user interface. Alternatively, Financial Close also allows the end user to import the service definitions in the form of an integration XML document.

Creating Integrations using the Integration XML Document

To integrate a large number of services or to define the integrations without using the Financial Close user interface, the end user can create the Integration Type XML document. This document should contain all of the Integration Types associated with the third party application being integrated.

Integration Type XML Schema

The following XML schema represents the structure and content for the Integration Type XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xs:element name="integrationTypes">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="application" maxOccurs="1" minOccurs="0">
     <xs:complexType>
      <xs:attribute name="applicationName" use="required" type="xs:string"/>
      <xs:attribute name="webAppName" use="required" type="xs:string"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="integrationType" maxOccurs="unbounded" minOccurs="1">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="responseXSLTemplate" type="xs:string" maxOccurs="1"
                   minOccurs="0"/>
       <xs:element name="responseNamespace" type="xs:anyURI" maxOccurs="1"
                   minOccurs="0"/>
       <xs:element name="name" maxOccurs="unbounded" minOccurs="1">
        <xs:complexType mixed="true">
         <xs:attribute name="language" use="required" type="xs:string"/>
        </xs:complexType>
```

```
</xs:element>
<xs:element name="description" maxOccurs="unbounded" minOccurs="1">
 <xs:complexType mixed="true">
  <xs:attribute name="language" use="required" type="xs:string"/>
 </xs:complexType>
</xs:element>
<xs:element name="parameter" maxOccurs="unbounded" minOccurs="0">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="xsltTransformation" type="xs:string" maxOccurs="1"
               minOccurs="0"/>
   <xs:element name="dimSelectorSelection" maxOccurs="1" minOccurs="0">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Parameters">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="ParameterNames">
          <xs:complexType>
           <xs:sequence>
            <xs:element name="ParameterName" maxOccurs="unbounded">
             <xs:complexType>
              <xs:sequence>
               <xs:element name="Name" type="xs:string" maxOccurs="1"
                           minOccurs="1"/>
               <xs:element name="Value" type="xs:string" maxOccurs="1"
                           minOccurs="1"/>
              </xs:sequence>
             </xs:complexType>
            </xs:element>
           </xs:sequence>
          </xs:complexType>
         </xs:element>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="dependency" maxOccurs="unbounded" minOccurs="0">
    <xs:complexType>
     <xs:attribute name="parameterCode" use="required" type="xs:string"/>
     <xs:attribute name="parameterName" use="required" type="xs:string"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="name" maxOccurs="1" minOccurs="1">
    <xs:complexType mixed="true">
     <xs:attribute name="language" use="required" type="xs:string"/>
```

```
  </xs:complexType>
 </xs:element>
 <xs:element name="selectItem" maxOccurs="unbounded" minOccurs="0">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="name" maxOccurs="unbounded" minOccurs="1">
     <xs:complexType mixed="true">
      <xs:attribute name="language" use="required" type="xs:string"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="order" use="required" type="xs:string"/>
   <xs:attribute name="selectItemCode" use="required"
                 type="xs:string"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="docNavDocTypeXsltTransformation" type="xs:string"
             maxOccurs="1" minOccurs="0"/>
</xs:sequence>
<xs:attribute name="SOAPAction" type="xs:string"/>
<xs:attribute name="dimSelectorURL" type="xs:anyURI"/>
<xs:attribute name="docNavDocTypeName" type="xs:string"/>
<xs:attribute name="docNavDocTypeParameter" type="xs:string"/>
<xs:attribute name="docNavDocTypePort" type="xs:string"/>
<xs:attribute name="docNavDocTypeRespElement" type="xs:NMTOKEN"/>
<xs:attribute name="docNavDocTypeSOAPAction" type="xs:string"/>
<xs:attribute name="docNavDocTypeURI" type="xs:anyURI"/>
<xs:attribute name="docNavDocTypeWSDLLoc" type="xs:anyURI"/>
<xs:attribute name="docNavFolderParameter" type="xs:string"/>
<xs:attribute name="docNavOneTimeQuery" type="xs:string"/>
<xs:attribute name="multiselect">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="Y"/>
   <xs:enumeration value="N"/>
  </xs:restriction>
 </xs:simpleType>
</xs:attribute>
<xs:attribute name="order" use="required" type="xs:integer"/>
<xs:attribute name="parameterCode" use="required" type="xs:string"/>
<xs:attribute name="parameterPathCode" type="xs:string"/>
<xs:attribute name="parameterType" use="required">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="TEXT"/>
   <xs:enumeration value="INTEGER"/>
   <xs:enumeration value="NUMBER"/>
   <xs:enumeration value="DATE"/>
```

```
      <xs:enumeration value="BOOLEAN"/>
      <xs:enumeration value="LOV"/>
      <xs:enumeration value="DYNLOV"/>
      <xs:enumeration value="RADIO"/>
      <xs:enumeration value="USER"/>
      <xs:enumeration value="POV"/>
      <xs:enumeration value="DOCNAV"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="required" use="required">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="Y"/>
      <xs:enumeration value="N"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="serviceName" type="xs:string"/>
   <xs:attribute name="serviceNamespace" type="xs:anyURI"/>
   <xs:attribute name="servicePort" type="xs:string"/>
   <xs:attribute name="serviceResponseElement" type="xs:NMTOKEN"/>
   <xs:attribute name="serviceWSDLLoc" type="xs:anyURI"/>
  </xs:complexType>
 </xs:element>
</xs:sequence>
<xs:attribute name="applicationCode" use="required" type="xs:string"/>
<xs:attribute name="callbackOperation" type="xs:string"/>
<xs:attribute name="callbackPortType" type="xs:string"/>
<xs:attribute name="endUserURL" type="xs:anyURI"/>
<xs:attribute name="executionType" use="required">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="S"/>
   <xs:enumeration value="U"/>
  </xs:restriction>
 </xs:simpleType>
</xs:attribute>
<xs:attribute name="integrationTypeCode" use="required" type="xs:string"/>
<xs:attribute name="serviceName" type="xs:string"/>
<xs:attribute name="serviceNamespace" type="xs:anyURI"/>
<xs:attribute name="serviceOperation" type="xs:string"/>
<xs:attribute name="servicePortType" type="xs:string"/>
<xs:attribute name="serviceWSDLLoc" type="xs:anyURI"/>
<xs:attribute name="userCreated" use="required">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="Y"/>
```

```
          <xs:enumeration value="N"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:attribute>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

Integration XML Elements and Attributes

- **integrationType** Element - the integrationType element is the root element of the XML tree. It describes one Integration Type in Financial Close.

    - **integrationTypeCode** - a code to uniquely identify this Integration Type. This is used to ensure that multiple uploads of the XML can determine if a new Integration Type should be created or if an existing one is being updated. It is also passed as a parameter when the task executes.

    - **applicationCode** – the application code for this Integration Type. The application code is used to segregate Integration Types within Financial Close.

    - **userCreated** – a flag indicating whether an end user created this Integration Type or the Integration Type was seeded by Financial Close.

    - **executionType** – a flag indicating if the task is a System Automated or End User task. Use "S" for System Automated and "U" for End User task.

    - **endUserURL** – the web application URL that can be called to display the end user task module. This attribute is only required for end user tasks.

    - **serviceWSDLLoc** – the location of the web service WSDL that describes the service used to invoke the system automated task. This attribute is only required for system automated tasks.

    - **serviceNamespace** – the web service namespace as defined in the service's WSDL. This attribute is only required for system automated tasks.

    - **servicePortType** – the web service port type as defined in the service's WSDL. This attribute is only required for system automated tasks.

    - **serviceName** – the web service name as defined in the service's WSDL. This attribute is only required for system automated tasks.

- **serviceOperation** – the name of the method to invoke on the defined web service. This attribute is only required for system automated tasks.

- **callbackPortType** – the web service port type for the asynchronous callback. This attribute is only required for system automated tasks.

- **callbackOperation** – the name of the method that will be invoked on the callback web service. This attribute is only required for system automated tasks.

- **responseXSLTTemplate** – an XSL transformation that will be applied to the task's response to transform it into a format recognized by Financial Close. Refer to the *System Automated Task Response XSD* section for more detail.

- **responseNamespace** – if the XSL defined in the responseXSLTemplate element contains additional namespaces, those namespaces need to be listed here as semi-colon delimited list of name=value pairs.

- **name** Element – the name element is used to store the translated name of each Integration Type. There can be a localized name element for each language supported by Financial Close.

  - **language** – the language of this name's translation. The languages currently supported by Financial Close are:

Language Attribute Value	Description
en	English
fr	French
de	German
ja	Japanese
es	Spanish
ru	Russian
ko	Korean
zh	Chinese
sv	Swedish
tr	Turkish
lt	Lithuanian
da	Danish
pt	Portuguese
fi	Finnish
nl	Dutch

- **description** Element – the description element is used to store the translated description of each Integration Type. There can be a localized description element for each language supported by Financial Close.

    - **language** – the language of this description's translation.

- **parameter** Element – the parameter element describes one parameter for the Integration Type. The parameters are sent to integrated application when the web service or end user task URL is invoked. See the *Integration Type Parameters* section for more detail.

    - **parameterCode** – a unique name for the parameter. This attribute must contain only letters and numbers, and must start with a letter.

    - **parameterType** – the type of parameter. This value will affect how the Financial Close user interface handles the display and validation of the parameter. See the *Integration Type Parameters* section for a description of the parameter types.

    - **required** – a flag indicating whether the parameter is required or not.

    - **order** – a number representing the display order for the parameters in the Financial Close user interface.

    - **multiselect** – a flag indicating that the user can select multiple values (default="N"). This attribute is only valid for parameter types LOV, DYNLOV, and RADIO. If the parameter type is RADIO and the multiselect attribute = "Y", Financial Close will display the parameter as a checkbox group.

    - **inputSize** – the maximum length of the string allowed in input box. This attribute is only valid for TEXT parameter types.

    - **rows** – the number of rows to display for the input box (default = 1). This attribute is only valid for TEXT parameter types.

    - **numLowerBound** – the lowest number allowed. This attribute is only valid for NUMBER and INTEGER parameter types.

    - **numUpperBound** – the highest number allowed. This attribute is only valid for NUMBER and INTEGER parameter types.

    - **dateLowerBound** – the earliest date allowed. This attribute is only valid for DATE parameter types.

    - **dateUpperBound** – the latest date allowed. This attribute is only valid for DATE parameter types.

- **dimSelectorURL** – internal use only.

- **serviceWSDLLoc** – the location of the web service that will enumerate the values for the dynamic parameter types. See the *Parameter Type* section for more detail. This attribute is only valid for DYNLOV and DOCNAV parameter types.

- **serviceNamespace** – the namespace for the web service that will enumerate the values for the dynamic parameter types. See the *Parameter Type* section for more detail. This attribute is only valid for DYNLOV and DOCNAV parameter types.

- **servicePort** – the port for the web service that will enumerate the values for the dynamic parameter types. See the *Parameter Type* section for more detail. This attribute is only valid for DYNLOV and DOCNAV parameter types.

- **serviceName** – the service name for the web service that will enumerate the values for the dynamic parameter types. See the *Dynamic List Parameter Type* section for more detail. This attribute is only valid for DYNLOV and DOCNAV parameter types.

- **SOAPAction** – the name of the method to invoke on the web service that will enumerate the values for the dynamic parameter types. See the *Parameter Type* section for more detail. This attribute is only valid for DYNLOV and DOCNAV parameter types.

- **serviceResponseElement** – the name of the XML element within the web service response that contains the XML that Financial Close expects for the parameter.

- **xsltTransformation** – the XSLT transformation string that will transform each of the children of the serviceResponseElement to the format expected by Financial Close. This attribute is only valid for the DYNLOV and DOCNAV parameter types. This attribute is not needed if the web service returns the format identical to what Financial Close expects.

- **parameterPathCode** – if the Integration Type requires that DOCNAV parameter types contain the full document path, this attribute should contain the name of the enumeration service's response element that contains the path. This attribute is valid only for the DOCNAV parameter type.

- **docNavOneTimeQuery** – a flag to indicate if the web service for the document navigator parameter type returns an iterative result. See the *Document Navigator Parameter Type* section for more detail. This attribute is valid only for the DOCNAV parameter type.

- **docNavFolderParameter** – if the docNavOneTimeQuery flag is set to "Y", this attribute should contain the parameter that Financial Close needs to set the folder ID being drilled on. This attribute is valid only for the DOCNAV parameter type.

- **docNavFileTypeWSDLLoc** – the WSDL location of the web service that can enumerate the file types available for the Document Navigator parameter type. See the *Document Navigator Parameter Type* section for more detail. This attribute is valid only for the DOCNAV parameter type.

- **docNavFileTypeURI** – the namespace for the web service that will enumerate the file types available for the Document Navigator parameter type. This attribute is valid only for the DOCNAV parameter type.

- **docNavFileTypePort** – the port for the web service that will enumerate the file types available for the Document Navigator parameter type. This attribute is valid only for the DOCNAV parameter type.

- **docNavFileTypeName** – the service name for the web service that will enumerate the file types available for the Document Navigator parameter type. This attribute is valid only for the DOCNAV parameter type.

- **docNavFileTypeSOAPAction** – the name of the method to invoke on the web service that will enumerate the file types available for the Document Navigator parameter type. This attribute is valid only for the DOCNAV parameter type.

- **docNavFileTypeRespElement** – the name of the XML element within the web service response that contains the XML that Financial Close expects for the parameter. This attribute is valid only for the DOCNAV parameter type.

- **docNavFileTypeXsltTransformation** – the XSLT transformation to be applied to the docNavFileTypeResponseElement to transform the response into the format expected by Financial Close. This transformation is used against each child of the docNavFileTypeRespElement. This attribute is valid only for the DOCNAV parameter type.

- **docNavFileTypeParameter** – the name of the parameter passed to the Document Navigator webservice to indicate which file type the user is filtering on. This attribute is valid only for the DOCNAV parameter type.

- **selectItem** Element – the selectItem describes one element in the selectable list for the LOV and RADIO type parameters.

    - **selectItemCode** – the code that Financial Close will use as the value of the LOV or RADIO parameter if this item is selected by the user. The attribute must contain only letters and numbers and must start with a letter.

    - **order** – an integer value indicating which order to display the item in the LOV or RADIO group in the Financial Close user interface.

- **dependency** Element – the dependency element indicates that one parameter is dependent on another parameter. The parameters that are listed as dependencies are considered parents of the current parameter. In the Financial Close user interface, the parent parameters will be processed before the child parameters.

- **parameterCode** – the code of the parameter that this parameter is dependent on.

- **parameterName** – the name of the parameter used to pass this value to the parent parameter's service.

Loading the Integration Type XML

After the Integration Type XML has been created, the XML must be loaded into Financial Close to create or update the integrations in the system. For the detailed procedure on loading integrations into Financial Close Management, see the *Oracle Hyperion Financial Close Management Administrator's Guide.*

Creating Integrations with the Financial Close User Interface

Integration Types can also be created by using the Financial Close Manage Integration Types module. From within the Financial Close user interface, select Manage and then Manage Integration Types. For details on creating and managing Integration Types, see the *Oracle Hyperion Financial Close Management Administrator's Guide.*

Sample Integration XML

The following integration XML will create two integrations types in Financial Close.

The first integration type, RUNREPORT, points to a web service for running an automated report on the http://rptsvr:80/reportingService/ReportingService web service. The web service method runReport takes a single parameter reportName that identifies the report to run. The available reports will be listed by the Close Calendar UI by calling the http://rptsvr:80/reportingService/ReportListingService web service method getReports as defined in the integration type.

The second integration type, VIEWREPORT, points to an end user UI module at the URL http://rptsvr1:80/reportingApp/ViewReport .jsp. When called in a browser, the URL displays the report for the end user. The ViewReport URL also takes a single parameter REPORTNAME on the URL. The actual REPORTNAME value will be replaced when the URL is launched. The available reports will be listed by the Close Calendar UI by calling the getReports web service method as defined in the integration type.

The results from the runReport and getReports methods are not in the format expected by Financial Close so the associated XSL templates are included in the integration type definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<integrationTypes xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                  xmlns:fcmns1="http://reports.xyz.com">
  <application applicationName="Sample"/>
  <integrationType integrationTypeCode="RUNREPORT" applicationCode="Sample"
                executionType="S" userCreated="N"
                serviceNamespace="http://reports.xyz.com"
                serviceWSDLLoc="http://rptsvr:80/reportingService/ReportService?WSDL"
                serviceName="ReportService" servicePortType="ReportService"
                callbackPortType="ReportServiceResponse" serviceOperation="runReport"
                callbackOperation="RunReportResponse">
    <responseXSLTemplate>
      <![CDATA[
      <xsl:template match="return">
        <fcmns1:TaskResult>
          <fcmns1:Result>
            <xsl:choose>
              <xsl:when test="resultCode = 'Success'">Success</xsl:when>
              <xsl:otherwise>Failure</xsl:otherwise>
            </xsl:choose>
          </fcmns1:Result>
          <fcmns1:LogLocation>
            <xsl:value-of select="resultLogPath"/>
```

```
        </fcmns1:LogLocation>
        <fcmns1:Messages>
          <fcmns1:Message>
            <fcmns1:Value>
              <xsl:value-of select="resultMessage"/>
            </fcmns1:Value>
          </fcmns1:Message>
        </fcmns1:Messages>
      </fcmns1:TaskResult>
    </xsl:template>
    ]]>
  </responseXSLTemplate>
  <name language="en" countryCode="US">Run Report</name>
  <description language="en" countryCode="US">
    Run Report Automated Task
  </description>
  <parameter parameterCode="reportName" required="Y"
       parameterType="DYNLOV" order="1"
       serviceWSDLLoc="http://rptsvr:80/reportingService/ReportListingService?WSDL"
       serviceNamespace="http://reports.xyz.com" serviceName="ReportListingService"
       servicePort="ReportListingServicePortType"
       serviceResponseElement="ns0:getReportsResponse" SOAPAction="getReports">
    <name language="en" countryCode="US">Report Name</name>
    <xsltTransformation>
      <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                      xmlns:ns0="http://reports.xyz.com" version="1.0">
        <xsl:template match="ns0:getReportsResponse">
          <selectItems>
            <xsl:for-each select="return">
              <selectItem>
                <code>
                  <xsl:value-of select="."/>
                </code>
                <name>
                  <xsl:value-of select="."/>
                </name>
              </selectItem>
            </xsl:for-each>
          </selectItems>
        </xsl:template>
      </xsl:stylesheet>
    </xsltTransformation>
  </parameter>
</integrationType>
<integrationType integrationTypeCode="VIEWREPORT" applicationCode="Sample"
 executionType="U" userCreated="N"
 endUserURL="http://rptsvr:80/reportingApp/ViewReport.jsp?reportName=$REPORTNAME$">
  <name language="en" countryCode="US">Manage Documents</name>
```

```
    <description language="en" countryCode="US">View Report UI Task</description>
    <parameter parameterCode="REPORTNAME" required="Y"
         parameterType="DYNLOV" order="1"
         serviceWSDLLoc="http://rptsvr:80/reportingService/ReportListingService?WSDL"
         serviceNamespace="http://reports.xyz.com" serviceName="ReportListingService"
         servicePort="ReportListingServiceReportType"
         serviceResponseElement="ns0:getReportsResponse" SOAPAction="getReports">
      <name language="en" countryCode="US">Report Name</name>
      <xsltTransformation>
        <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                        xmlns:ns0="http://reports.xyz.com" version="1.0">
          <xsl:template match="ns0:getReportsResponse">
            <selectItems>
              <xsl:for-each select="return">
                <selectItem>
                  <code>
                    <xsl:value-of select="."/>
                  </code>
                  <name>
                    <xsl:value-of select="."/>
                  </name>
                </selectItem>
              </xsl:for-each>
            </selectItems>
          </xsl:template>
        </xsl:stylesheet>
      </xsltTransformation>
    </parameter>
  </integrationType>
</integrationTypes>
```