Knowledge to Shape Your Future

Itron Enterprise Edition™

Task Web Services XML API Guide

v7.0 SP4

www.itron.com

Identification

Itron Enterprise Edition Task Web Services XML API Guide
7/22/2011
v7.0 SP4

Copyright

Confidentiality Notice

Trademark Notice

Suggestions

If you have comments or suggestions on how we may improve this documentation, send them to TechnicalCommunicationsManager@itron.com
If you have questions or comments about the software or hardware product, contact Itron Technical Support:

Contact

- Internet: www.itron.com
- E-mail: support@itron.com
- Phone: 1 877 487 6602

Contents

Before You Begin

This document describes an application programming interface (API) for the Itron Enterprise Edition (IEE) system. An API is a programmatic interface that a computer system provides to support interaction and communication with other computer systems.

IEE API guides typically include the following sections:

- An introduction to the API
- Definitions of the extensible markup language (XML) elements and attributes that are allowed by the API
- Definitions of the operations that are supported by the API
- Sample code that illustrates how to perform common business processes
- Schemas (in the form of one or more XSD files) and Web service descriptions (in the format of one or more WSDL files)

Audience

This guide is intended for experienced IEE implementation teams.

Related Documents

IEE provides associated API materials including XSDs, WSDLs, HTML versions of the schemas, and code samples. These materials are located on the IEE installation disc or installation package under \ReferenceGuides. You can also find them on your Itron customer portal.

Terminology

The IEE Web services and API guides use the following terminology.

- **API**. This guide uses the term API to refer to the complete application programming interface.
- **Element**. The IEE APIs use elements to classify data in the XML document. Elements are sometimes referred to as tags or XML tags. An IEE API element uses an opening tag and a closing tag to define where the element begins and ends. An element can contain attributes, values, and other elements.
- **Entity**. Each node in the IEE database is called an entity. For example, service points, customers, and meters are all entities. The XML APIs use elements and attributes to define the properties of each database entity.
- **Function call**. See Operation.
- **Method**. See Operation.
- **Operation**. An operation is a service that is provided by the API. The IEE APIs use operations to define the requested actions, such as adding a meter or assigning a rate. Operations are sometimes referred to as methods, services, or function calls in other programming languages.
- **Sample code**. Sample code illustrates the syntax and properties that are required to perform a typical business operation.
- **Service**. See Operation.
- **User-defined attribute (UDA)**. Administrators create UDAs and assign their values in the IEE rich client.
- **Value**. The value is the definition of an attribute or element. Values can be binary (such as t**rue/false** or **1/0**) or one of a group of definitions (such as **Active**, **Inactive**, or **Standby**). In the example <Meter Type="Centron" Active="True">, **CENTRON** is the value for the Type attribute, and **True** is the value for the Active attribute.

Document Conventions

IEE API guides use the following document conventions:

Schema diagrams. Display as annotated illustrations. Additional text descriptions and a list of elements follow the illustration, as applicable. Some schema illustrations do not provide additional text descriptions other than the annotations shown in the illustration.

Elements. Display in a bulleted list. You can also reference the corresponding XML schema for additional information and annotations.

Operation names. Display as a string with no spaces. The operation names are case-specific. For example, the billing export operation displays as: Service: BillingExportService, where BillingExportService is the operation name.

Operation diagrams. Display as annotated illustrations. A description follows the illustration, as applicable. Some operation illustrations do not provide additional text descriptions other than the annotations shown in the illustration.

Caution, Warning, Tip, and Note text boxes. See the following explanations.

Caution A caution note advises you that failure to take or avoid a specified action could result in a loss of data.

Warning A warning note advises you that failure to take or avoid a specified action could result in physical harm to the user or the hardware.

Tip A tip note helps you apply the techniques and procedures described in the text to your specific needs. A tip provides you with extra hints to make a task easier to perform or a concept easier to understand.

Note A note indicates neutral or positive information that stresses or supplements important points of the main text. A note supplies information that may apply only in special cases.

C H A P T E R 1

Task XML Adapter (ASMX Version)

This document introduces the IEE Task Web services and describes the available functionality in the API for the IEE Task XML Adapter. All IEE APIs require a request document that uses properly constructed XML, in the format that is described in this document and modeled in the XML schemas.

There are two versions of IEE Task APIs:

- The Active Server Methods Extension (ASMX) version is based on legacy Web service technology and can be hosted only in Internet Information Services (IIS).
- The Windows Communication Foundation (WCF) version is based on newer Web service technology and can be hosted in any managed application, including both IIS and the IEE application server.

You can view the XML schemas on the computer where the IEE rich client is installed at %Program Files%\Itron\Enterprise Edition\Documentation\XML Schemas.

Each API produces an XML-format results document. This results document contains any messages that IEE logged with the request along with the requested data.

The IEE APIs are all independent, but can overlap. For example, to export readings, you can either use the Reading API's export method or use the Task API to run or schedule a Reading XML Export task.

Note All Itron Web services support Web services security through WS-Security credentials (part of WSE) and Secure Sockets Layer (SSL) when acting as Web service providers. For more information, see the *IEE Web Services User Guide*.

Task API

The Task API provides an interface to the Task Management system. It enables you to create a task and to either run it inline on the Web service or schedule it for the next available task runner.

When you create a request document for the Task API, you specify the IEE task template to use for the request. You also define the parameters for the task, and your defined parameter values override the default values in the task template. To view or modify task templates, use the System Operations workbench in the IEE rich client. For more information, see the *IEE Rich Client Help*.

The Task API calls the same task adapters that are used for manual imports and exports initiated through the IEE rich client.

Writing Client Programs to Call a Web Service

There are two options for calling the IEE Web services. You can either use a Web service testing tool or write your own code. For information on writing code to call IEE Web services by using one of the many available technologies, see Microsoft's documentation.

Creating Request Documents

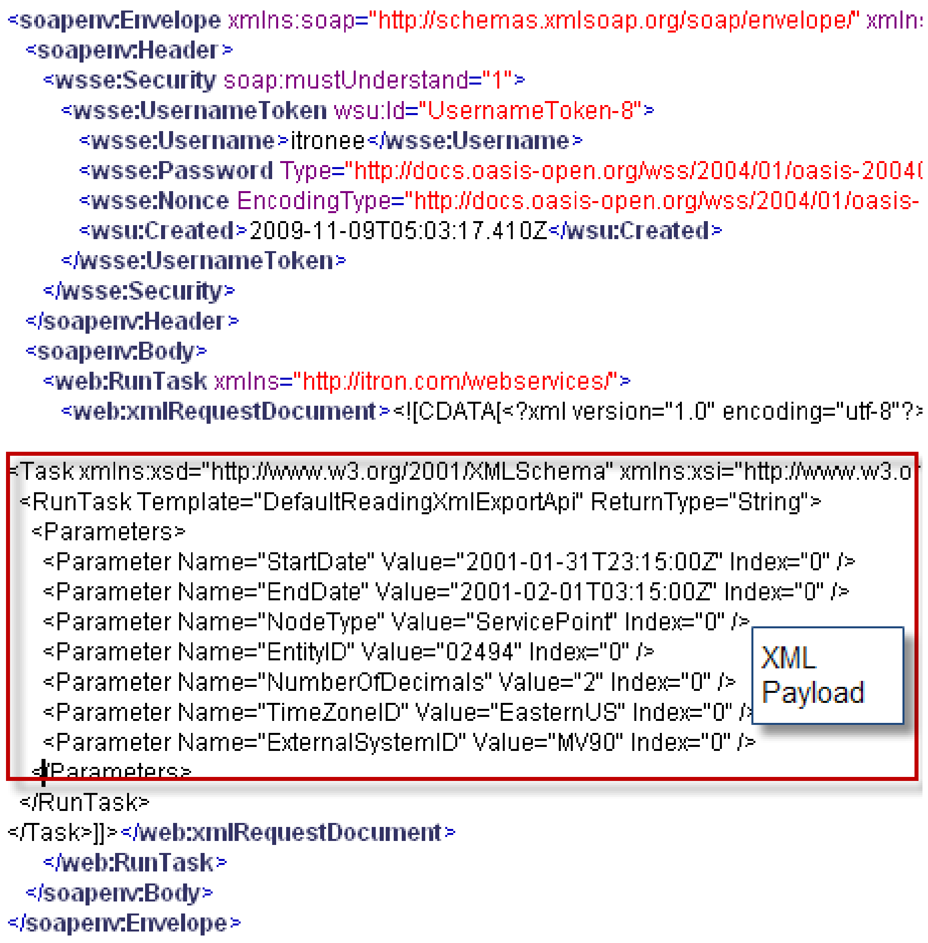
The request document that your program creates must use one of the following operations:

- **RunTask.** Use the RunTask operation to run the task immediately and to receive task output in the results document. This method is useful for infrequent tasks, but overuse of this method can slow down your Web server.
- **ScheduleTask.** Use the ScheduleTask operation to create a task that will be run by the next available task runner. This method is best for handling multiple tasks.

The RunTask and ScheduleTask methods use a loosely typed Web service model. The following sample shows the structure of a typical SOAP request. A CDATA section is required because the Web service takes a string as an argument. The contents of the document are encapsulated in a CDATA section, which enables them to be interpreted as character data. The string that is passed in would normally be the exact contents of the TaskAPI XML file in the file-based importer.

```
<soapenv:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://itron.com/webservices/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Header>
      <wsse:Security soap:mustUnderstand="1">
         <wsse:UsernameToken wsu:Id="UsernameToken-8">
            <wsse:Username>itronee</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">itronee</wsse:Password>
            <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary">mgSvRic12tNQzH4mBaNtig==</wsse:Nonce>
            <wsu:Created>2009-11-09T05:03:17.410Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
   </soapenv:Header>
   <soapenv:Body>
      <web:RunTask xmlns="http://itron.com/webservices/">
         <web:xmlRequestDocument><![CDATA<?xml version="1.0" encoding="utf-8"?>
<Task xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <RunTask Template="DefaultReadingXmlExportApi" ReturnType="String">
    <Parameters>
      <Parameter Name="StartDate" Value="2001-01-31T23:15:00Z" Index="0" />
      <Parameter Name="EndDate" Value="2001-02-01T03:15:00Z" Index="0" />
      <Parameter Name="NodeType" Value="ServicePoint" Index="0" />
      <Parameter Name="EntityID" Value="02494" Index="0" />
      <Parameter Name="NumberOfDecimals" Value="2" Index="0" />
      <Parameter Name="TimeZoneID" Value="EasternUS" Index="0" />
      <Parameter Name="ExternalSystemID" Value="MV90" Index="0" />
    </Parameters>
  </RunTask>
</Task>]]></web:xmlInputDocument>
      </web:RunTask>
   </soapenv:Body>
</soapenv:Envelope>
```

The following sample is a more graphical representation of how the XML payload is embedded into the CDATA structure:



```
<soapenv:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmln:
  <soapenv:Header>
    <wsse:Security soap:mustUnderstand="1">
      <wsse:UsernameToken wsu:Id="UsernameToken-8">
        <wsse:Username>itronee</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-2004(
        <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
        <wsu:Created>2009-11-09T05:03:17.410Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <web:RunTask xmlns="http://itron.com/webservices/">
      <web:xmlRequestDocument><![CDATA[<?xml version="1.0" encoding="utf-8"?>

<Task xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.o
 <RunTask Template="DefaultReadingXmlExportApi" ReturnType="String">
  <Parameters>
   <Parameter Name="StartDate" Value="2001-01-31T23:15:00Z" Index="0" />
   <Parameter Name="EndDate" Value="2001-02-01T03:15:00Z" Index="0" />
   <Parameter Name="NodeType" Value="ServicePoint" Index="0" />
   <Parameter Name="EntityID" Value="02494" Index="0" />
   <Parameter Name="NumberOfDecimals" Value="2" Index="0" />
   <Parameter Name="TimeZoneID" Value="EasternUS" Index="0" />
   <Parameter Name="ExternalSystemID" Value="MV90" Index="0" />
  </Parameters>
 </RunTask>
</Task>]]></web:xmlRequestDocument>
    </web:RunTask>
  </soapenv:Body>
</soapenv:Envelope>
```

The following samples illustrate only the XML payload. In a file-based import, the payload would be the contents of the XML file that is dropped into the import directory to schedule or run the task. In a Web service call, the payload would be the contents of the CDATA section of the SOAP message.

Sample: Readings Export Using Task API

This sample shows a basic Readings Export request. The RunTask operation is marked in bold. The document that is created by the Readings Export is returned in the results document.

Tip Use different templates for RunTask operations and ScheduleTask operations, to distinguish them in the IEE rich client Task Monitor. Too many API tasks might account for a slow Web server.

```
<?xml version="1.0" encoding="utf-8"?>
<Task xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <RunTask Template="DefaultReadingXmlExportApi" ReturnType="String">
    <Parameters>
      <Parameter Name="StartDate" Value="2001-01-31T23:15:00Z" Index="0" />
      <Parameter Name="EndDate" Value="2001-02-01T03:15:00Z" Index="0" />
      <Parameter Name="NodeType" Value="ServicePoint" Index="0" />
      <Parameter Name="EntityID" Value="02494" Index="0" />
      <Parameter Name="NumberOfDecimals" Value="2" Index="0" />
      <Parameter Name="TimeZoneID" Value="EasternUS" Index="0" />
      <Parameter Name="ExternalSystemID" Value="MV90" Index="0" />
    </Parameters>
  </RunTask>
</Task>
```

Sample: Readings Export for Confirmed Billing Data

This sample shows a Readings Export request file for requesting data that has been confirmed for billing. The RunTask operation is marked in bold. The sample uses some of the parameters on the Readings XML Export task template, such as FilterBySignificanceID and Confirmed. The document that is created by the Readings Export is returned in the results document.

```
<?xml version="1.0" encoding="utf-8"?>
<Task xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <RunTask Template="DefaultReadingXmlExportApi" ReturnType="String">
    <Parameters>
      <Parameter Name="StartDate" Value="2001-01-31T23:15:00Z" Index="0" />
      <Parameter Name="EndDate" Value="2001-02-01T03:15:00Z" Index="0" />
      <Parameter Name="EntityID" Value="02494" Index="0" />
      <Parameter Name="CorrelationID" Value="8675309" Index="0" />
      <Parameter Name="FilterBySignificanceID " Value="Billing" Index="0" />
      <Parameter Name="Confirmed " Value="true" Index="0" />
     </Parameters>
  </RunTask>
</Task>
```

Sample: Scheduling Readings Export Using Task API

This sample shows a basic Readings Export request that is scheduled to run on the next available task runner. The ScheduleTask operation is marked in bold.

The parameters specify the destination directory and file name for the export file. The results document indicates whether the task was successfully scheduled.

```
<?xml version="1.0" encoding="utf-8"?>
<Task xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ScheduleTask Template="DefaultReadingXmlExport" ReturnType="String">
    <Parameters>
      <Parameter Name="StartDate" Value="2001-01-31T23:15:00Z" Index="0" />
      <Parameter Name="EndDate" Value="2001-02-01T03:15:00Z" Index="0" />
      <Parameter Name="NodeType" Value="ServicePoint" Index="0" />
      <Parameter Name="EntityID" Value="02494" Index="0" />
      <Parameter Name="NumberOfDecimals" Value="2" Index="0" />
      <Parameter Name="TimeZoneID" Value="EasternUS" Index="0" />
      <Parameter Name="ExternalSystemID" Value="MV90" Index="0" />
      <Parameter Name="ExportDirectoryName" Value="c:\temp\exports" Index="0" />
      <Parameter Name="ExportFileName" Value="DefaultParametersByID.xml" Index="0" />
      <Parameter Name="OverwriteExistingFile" Value="Overwrite" Index="0" />
    </Parameters>
  </ScheduleTask>
</Task>
```

Sample: ScheduleTask Results Document

A successful ScheduleTask request returns a results document that is similar to the following sample.

```
<?xml version="1.0" encoding="utf-8"?>
<ResultDocument xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" TransactionID=""
   ResultCode="Success">
  <Messages>
    <Message Severity="Information" Text="Successfully created and scheduled
   DefaultReadingXmlExport task. TaskID is 272." ID="43005" />
  </Messages>
</ResultDocument>
```

Enabling Third-Party E-mail Notification

The Task API Web service interface enables a third-party system to use IEE to send e-mail notifications. This interface is typically used when e-mails must be digitally signed or encrypted. Requests that are submitted to the Web service are saved as tasks in the IEE task system. The requests are processed by the task service on an application server.

Technical Description: Third-Party E-mail Notification

The Task API e-mail system uses the following as default values for missing parameters in the submitted XML document:

- Existing Email Service configuration that is defined in the code tables.
- Optional configuration that is defined in the TaskApiEmail task template.

The values in the submitted XML document override the values in the TaskApiEmail template. The values in the TaskApiEmail template override the values in the code tables.

A typical scenario uses the following values and leaves all template parameters blank. The submitted XML document includes only the EmailTo, optional EmailCc, EmailBcc, EmailSubject, EmailBody, optional EmailDataFormat, optional Email, and optional EmailPriority parameters.

- The Service Address (SMTP server) and Sender Address (EmailFrom) that are defined in the code table
- The internal default SMTP server port value of 25

The preferred format for the XML document includes only the parameters that are needed. Parameters with blank values ("") override any values that are configured in the template. Blank values are permitted only for the EmailReplyTo, EmailCc, EmailBcc, EmailSubject, EmailBody, EmailAttach, Email, and EmailPriority parameters. An error occurs if the EmailFrom, EmailTo, EmailServer, or EmailPort parameters are included with blank values.

When the following special characters are included in submitted parameter values, they must be properly escaped for use in an XML file.

Character	Escaped
<	&lt;
>	&gt;
"	&quote;
'	&apos;
&	&amp;

Use the XmlTextWriter.WriteString() method to escape the characters automatically, or use the System.Security.SecurityElement.Escape() static function, which accepts a string and returns an escaped string.

Sample: Third-Party E-mail Notification

This sample file includes all possible parameters and is not intended to represent an actual submitted document:

```
<?xml version="1.0" encoding="utf-8"?>
<Task xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ScheduleTask Template="TaskApiEmail" ReturnType="String">
    <Parameters>
      <Parameter Name="EmailFrom" Value="iee@itron.com" />
      <Parameter Name="EmailReplyTo" Value="iеereply@itron.com" />
      <Parameter Name="EmailTo" Value="test1@itron.com" />
      <Parameter Name="EmailCc" Value="test2@itron.com" />
      <Parameter Name="EmailBcc" Value="test3@itron.com" />
      <Parameter Name="EmailSubject" Value="subject" />
      <Parameter Name="EmailBody" Value="body" />
      <Parameter Name="EmailDataFormat" Value="text" />
      <Parameter Name="EmailAttach" Value="C:\Test.txt" />
      <Parameter Name="EmailServer" Value="smtpServer" />
      <Parameter Name="EmailPort" Value="25" />
      <Parameter Name="EmailPriority" Value="high" />
    </Parameters>
  </ScheduleTask>
</Task>
```

Parameter	Required?	Description	Value
EmailFrom	No. Including this parameter overrides both the From field in the task template and the Sender Address field in the Email Service code table.	The From address for the e-mail.	Enter one e-mail address. If Digital Signing is enabled, then this value must match an installed digital certificate on the application server that is sending the e-mails.
EmailReplyTo	No. Including this parameter overrides the Reply to field in the task template.	The address that will be used when e-mails are replied to.	Enter one e-mail address. This is an optional e-mail message header. If it is omitted, then the e-mail client uses the EmailFrom address.
EmailTo	Yes. At least one address is required for an e-mail to be sent.	One or more e-mail recipient addresses.	Enter e-mail addresses separated by semicolons.

Parameter	Required?	Description	Value
EmailCc	No. Including this parameter overrides the CC field in the task template.	One or more e-mail CC recipient addresses.	Enter e-mail addresses separated by semicolons.
EmailBcc	No. Including this parameter overrides the BCC field in the task template.	One or more e-mail BCC recipient addresses.	Enter e-mail addresses separated by semicolons.
EmailSubject	No. Including this parameter overrides the Subject field in the task template.	The subject of the e-mail.	Enter subject text. All special characters must be escaped for XML use.
EmailBody	No. Including this parameter overrides the Body field in the task template.	The body of the e-mail. The body format can be either plain text or HTML.	Enter body text. All special characters must be escaped for XML use. If the body format is HTML, then the Email parameter must be included and must be set to **html**.
EmailDataFormat	No. Including this parameter overrides the Message Type field in the task template.	The data format of the e-mail body.	Enter one of the following values: **html** or **text**. If this parameter is not included, then the default is plain text.
EmailAttach	No. Including this parameter overrides the Attachment field in the task template.	One or more paths to files that are added as attachments to the e-mail.	Enter full file paths separated by semicolons. The paths can be either local paths on the application server or Universal Naming Convention (UNC) paths to a file share system. All special characters must be escaped for XML use. The Windows user that the IEE task service is configured to run under must have sufficient security privileges to read the directory or file share where the files are located.
EmailServer	No. Including this parameter overrides both the Server field in the task template and the Server Address field in the Email Service code table.	Server name, URL, or Internet Protocol (IP) address of the SMTP server.	Enter the server name, URL, or IP address of the SMTP server.

Parameter	Required?	Description	Value
EmailPort	No. Including this parameter overrides the Port field in the task template.	The IP port number of the SMTP server.	Enter the port number of the SMTP server. If this parameter is not included, then the internal default is standard SMTP port 25.
EmailPriority	No. Including this parameter sets the priority of the e-mail.	The priority setting of the e-mail.	Enter one of the following values: **low**, **normal**, or **high**. If this parameter is not included, then the default is normal priority.

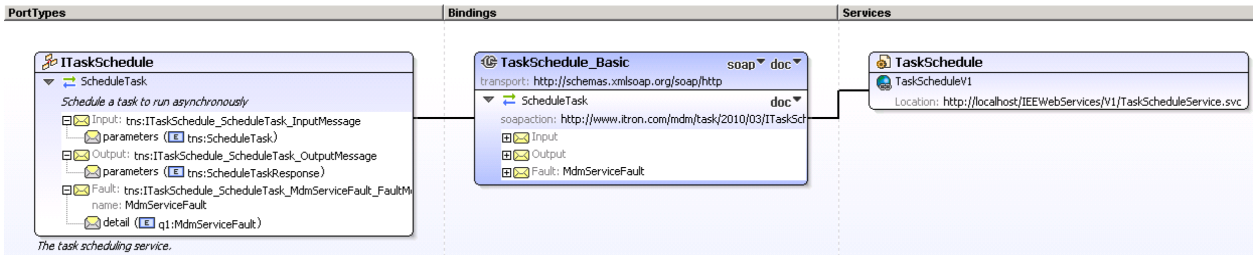
CHAPTER 2

Task XML Adapter (WCF Version)

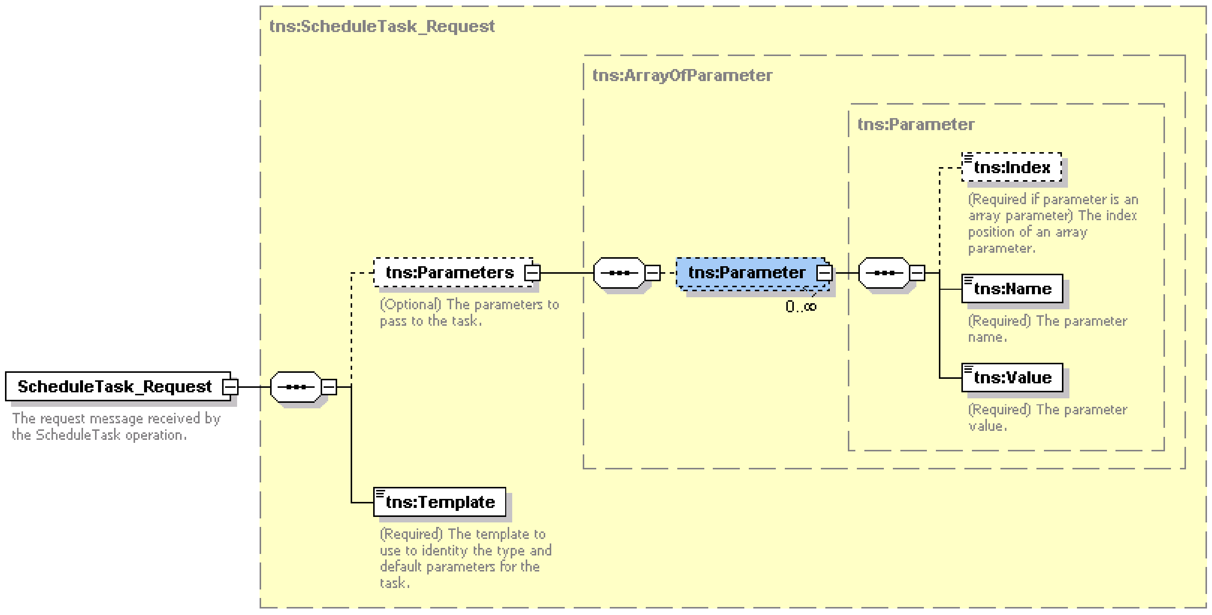
This section describes the WCF version of the IEE Task API. The WCF version provides similar functionality to the ASMX version, but enables the Web service to be hosted both on the IEE application server and on IIS. Unlike the ASMX version, the WCF version supports only asynchronous task execution. It does not support inline, synchronous task execution.

The WCF version provides one Web service, TaskSchedule in TaskScheduleService.svc.

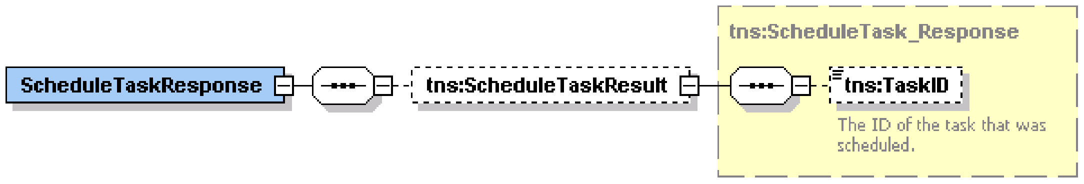
The TaskSchedule Web service provides an asynchronous interface to the IEE task system. A call to this service's TaskSchedule operation saves a task in IEE, and the task runs whenever the task system picks it up. The IEE task architecture operates in a hungry consumer mode. When a TaskSchedule call saves a task, it is put into the queue with all the other tasks that have been scheduled through the user interface. The returned task ID enables you to investigate the task's status using the IEE rich client Task Monitor.


PortTypes
Bindings
Services
ITaskSchedule
ScheduleTask
Schedule a task to run asynchronously
Input: tns:ITaskSchedule_ScheduleTask_InputMessage
parameters (E tns:ScheduleTask)
Output: tns:ITaskSchedule_ScheduleTask_OutputMessage
parameters (E tns:ScheduleTaskResponse)
Fault: tns:ITaskSchedule_ScheduleTask_MdmServiceFault_FaultM
name: MdmServiceFault
detail (E q1:MdmServiceFault)
The task scheduling service.
TaskSchedule_Basic
soap doc
transport: http://schemas.xmlsoap.org/soap/http
ScheduleTask
doc
soapaction: http://www.itron.com/mdm/task/2010/03/ITaskSch
Input
Output
Fault: MdmServiceFault
TaskSchedule
TaskScheduleV1
Location: http://localhost/IEEWebServices/V1/TaskScheduleService.svc

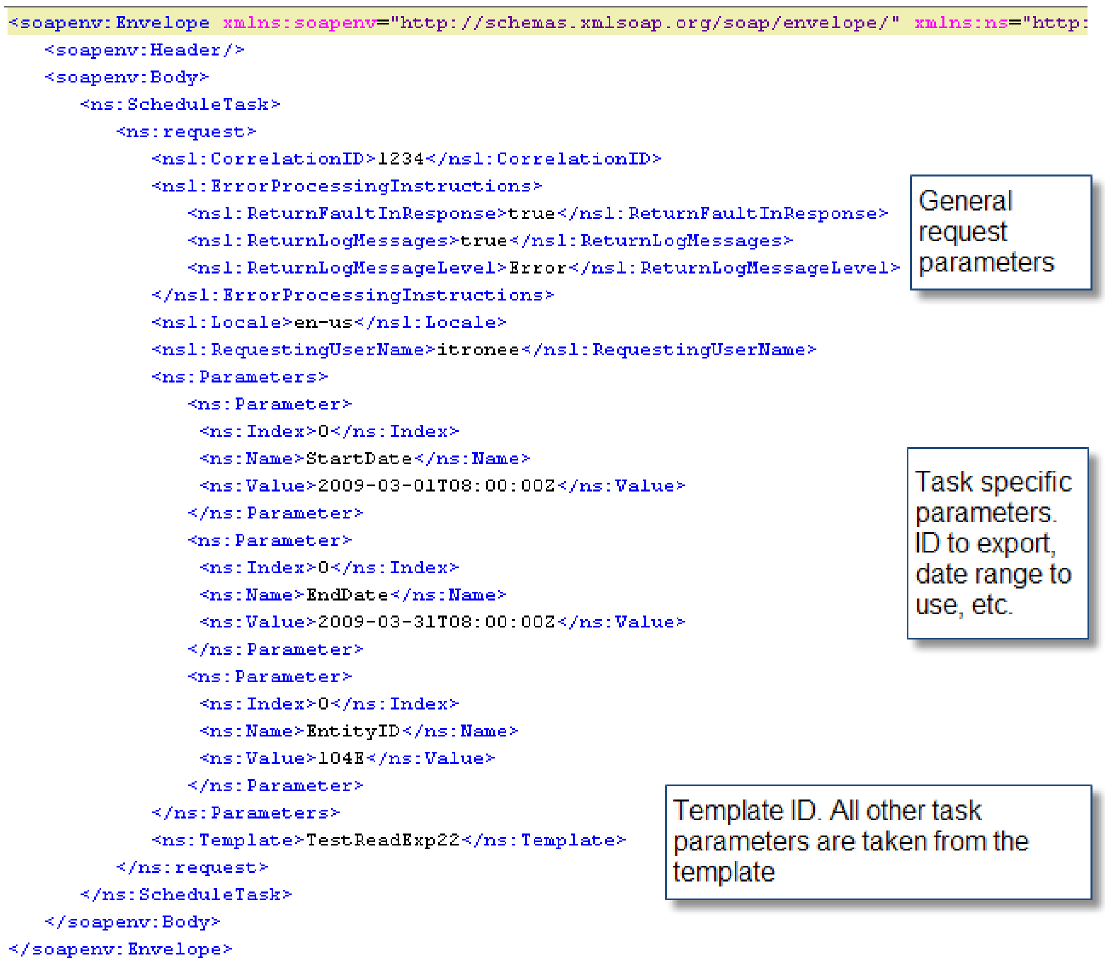
The service operation takes a request as input. The following illustration shows the schema for the request. The namespace is www.itron.com.mdm.task.2010.03. Use this namespace to specify a template ID, which is the task template in IEE that will be used to run the task. The remainder of the request object consists of name and value pairs that specify the task parameters to be used when the task is executed.


tns:ScheduleTask_Request
tns:ArrayOfParameter
tns:Parameter
tns:Index
(Required if parameter is an array parameter) The index position of an array parameter.
tns:Parameters
(Optional) The parameters to pass to the task.
tns:Parameter
0..∞
tns:Name
(Required) The parameter name.
tns:Value
(Required) The parameter value.
ScheduleTask_Request
The request message received by the ScheduleTask operation.
tns:Template
(Required) The template to use to identity the type and default parameters for the task.

The service operation returns a response message. The following illustration shows the schema for this response message. The response message provides the TaskID for the scheduled task in IEE. You can use the IEE rich client Task Monitor to check the task's status and to view any associated log messages.


tns:ScheduleTask_Response
ScheduleTaskResponse
tns:ScheduleTaskResult
tns:TaskID
The ID of the task that was scheduled.

The following is an example of a typical Task API call to schedule an export of readings data. This call assumes that the task template has been configured in IEE and that most of the task parameters are taken from the task template. The Task API request needs to provide only the required task-specific parameters for the task instance.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="http:
   <soapenv:Header/>
   <soapenv:Body>
      <ns:ScheduleTask>
         <ns:request>
            <ns1:CorrelationID>1234</ns1:CorrelationID>
            <ns1:ErrorProcessingInstructions>
               <ns1:ReturnFaultInResponse>true</ns1:ReturnFaultInResponse>
               <ns1:ReturnLogMessages>true</ns1:ReturnLogMessages>
               <ns1:ReturnLogMessageLevel>Error</ns1:ReturnLogMessageLevel>
            </ns1:ErrorProcessingInstructions>
            <ns1:Locale>en-us</ns1:Locale>
            <ns1:RequestingUserName>itronee</ns1:RequestingUserName>
            <ns:Parameters>
               <ns:Parameter>
                <ns:Index>0</ns:Index>
                <ns:Name>StartDate</ns:Name>
                <ns:Value>2009-03-01T08:00:00Z</ns:Value>
               </ns:Parameter>
               <ns:Parameter>
                <ns:Index>0</ns:Index>
                <ns:Name>EndDate</ns:Name>
                <ns:Value>2009-03-31T08:00:00Z</ns:Value>
               </ns:Parameter>
               <ns:Parameter>
                <ns:Index>0</ns:Index>
                <ns:Name>EntityID</ns:Name>
                <ns:Value>104E</ns:Value>
               </ns:Parameter>
            </ns:Parameters>
            <ns:Template>TestReadExp22</ns:Template>
         </ns:request>
      </ns:ScheduleTask>
   </soapenv:Body>
</soapenv:Envelope>
```

General request parameters

Task specific parameters. ID to export, date range to use, etc.

Template ID. All other task parameters are taken from the template

The following is the XML for the request:

```
<ns:ScheduleTask>

   <ns:request>

      <ns1:CorrelationID>1234</ns1:CorrelationID>

      <ns1:ErrorProcessingInstructions>

         <ns1:ReturnFaultInResponse>true</ns1:ReturnFaultInResponse>

         <ns1:ReturnLogMessages>true</ns1:ReturnLogMessages>

         <ns1:ReturnLogMessageLevel>Error</ns1:ReturnLogMessageLevel>

      </ns1:ErrorProcessingInstructions>

      <ns1:Locale>en-us</ns1:Locale>

      <ns1:RequestingUserName>itronee</ns1:RequestingUserName>

      <ns:Parameters>

         <ns:Parameter>

          <ns:Index>0</ns:Index>

          <ns:Name>StartDate</ns:Name>

          <ns:Value>2009-03-01T08:00:00Z</ns:Value>

        </ns:Parameter>

  <ns:Parameter>

          <ns:Index>0</ns:Index>

          <ns:Name>EndDate</ns:Name>

          <ns:Value>2009-03-31T08:00:00Z</ns:Value>

        </ns:Parameter>

         <ns:Parameter>

          <ns:Index>0</ns:Index>

          <ns:Name>EntityID</ns:Name>

          <ns:Value>104E</ns:Value>

          </ns:Parameter>

       </ns:Parameters>
```

```
          <ns:Template>TestReadExp22</ns:Template>

       </ns:request>

    </ns:ScheduleTask>
```

The following is the SOAP XML response from this call. Note that the TaskID is the task that was scheduled in IEE. You can check the task's status by using the IEE rich client Task Monitor.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">

 <s:Body>

    <ScheduleTaskResponse xmlns="http://www.itron.com/mdm/task/2010/03">

       <ScheduleTaskResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">

          <CorrelationID
xmlns="http://www.itron.com/mdm/common/2008/04">1234</CorrelationID>

          <LogMessages xmlns="http://www.itron.com/mdm/common/2008/04">

             <ServiceLogMessage>

                <Category>TaskApiMessages</Category>

                <ID>TaskApiMessages.SUCCESSFULLY_SCHEDULED_TASK</ID>

                <Message>Successfully created and scheduled TestReadExp22 task. TaskID is
1816.</Message>

                <Severity>Information</Severity>

                <Source>Itron.EE.Facade.Service</Source>

             </ServiceLogMessage>

          </LogMessages>

          <TaskID>1816</TaskID>

       </ScheduleTaskResult>

    </ScheduleTaskResponse>

 </s:Body>

</s:Envelope>
```

APPENDIX A

Middleware

This appendix describes the logic for invoking the IEE APIs from specific middleware tools.

BEA Aqualogic

The following procedures describe how to configure the system to use BEA Aqualogic.

To use BEA Aqualogic to invoke IEE Web services

1. Disable the replay detection on the application server's IIS.
2. Use a custom SOAP header to connect the BEA Aqualogic Service Bus (ALSB) to the IEE Web service.

To disable replay detection

Go to Itron\Enterprise Edition\WebService\web.config and edit the security parameters to match the following:

```
 <security>

      <securityTokenManager type="Itron.EE.Facade.API.Common.ItronEETokenManager,
Itron.EE.Facade.API.Common"

           xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd" qname="wsse:UsernameToken">

     <replayDetection enabled="false" />

      </securityTokenManager>

    </security>
```

To connect the BEA ALSB to the IEE Web service

Use the following code to authorize Aqualogic to the IEE Web service:

```
<soap:Header xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">

    <wsse:Security soap:mustUnderstand="0">

          <wsse:UsernameToken wsu:Id="SecurityToken-cdd5e395-853b-4930-bbd7-80f045d00a12">

          <wsse:Username>itronee</wsse:Username>

          <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">itronee</wsse:Password>

        </wsse:UsernameToken>

      </wsse:Security>

</soap:Header>
```