Oracle® Workforce Scheduling

Integration Technical Guide

Release 5.0.3 for Windows

Part No. E12663-01

June 2008

ORACLE®

Oracle® Workforce Scheduling Integration Technical Guide, Release 5.0.3 for Windows

Part No. 12663-01

Contents

Oracle Workforce Scheduling Interface Development Kit

Welcome to the OWS Integration Development Kit

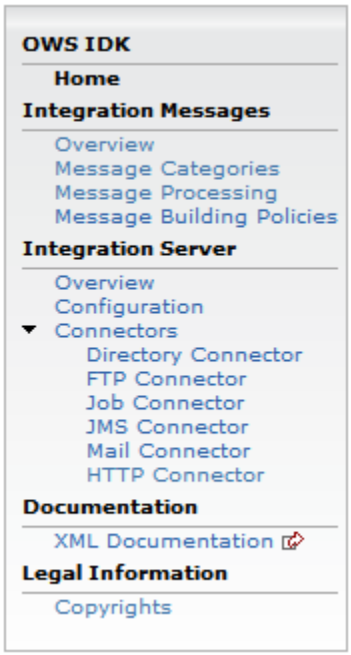
OWS IDK

The OWS Integration Development Kit is composed of several components:

- An integration server enabling the interaction between the OWS application and external application using asynchronous XML messages.
- An XML reference documentation describing the syntax and semantics of the business objects embodied by XML elements.

OWS IDK Contents

- The Integration Message Overview gives you an introduction to the structure of the business messages.
- The Integration Server Overview presents the main concepts underpinning the integration server.
- The XML Documentation provides a quick means to find information about the elements in the business messages.



Integration Message Overview

Message Content

The content of an OWS integration message is composed of:

- A set of properties (i.e. name/value pairs). These properties are either technical (e.g. the encoding) or related to message management (e.g. the message identifier, the correlation identifier).
- A body or payload. The body is business oriented. It contains an XML document associated with an OWS XML schema (via a namespace) defining its type.

XML Representation

The XML representation of an integration message can have two formats:

- A SOAP format:

  In that format, the integration message is a SOAP message and the root element is the SOAP Envelope. The SOAP header may contain an OWS element named **MessageHeader** defining the properties of the message. The message business payload is included in the SOAP body.

  The example below illustrates the SOAP format.

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance/">
  <soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:ows="http//www.oracle.com/ows/idk/Core">
    <ows:MessageHeader>
      <MessageId>1234567</MessageId>
      <Timestamp>2000-07-25T12:19:05</Timestamp>
    </ows:MessageHeader>
  </soapenv:Header>
  <soapenv:Body xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <msg:Organization xmlns:msg="http://www.oracle.com/ows/idk/Organization"
                      xmlns="http://www.oracle.com/ows/idk/Core">
      <Chart>...</Chart>
      <Variable>...</Variable>
      <Chart>...</Chart>
   </msg:Organization>
  </soapenv:Body>
</soapenv:Envelope>
```

- The raw (or stripped) format.

  In that format, there is no surrounding SOAP Envelope. The root element of the XML document is a business element. You should notice that this format does not allow you to define message properties. The example below shows the raw version of the previous SOAP message.

```
<msg:Organization xmlns:msg="http://www.oracle.com/ows/idk/Organization"
                  xmlns="http://www.oracle.com/ows/idk/Core">
  <Chart>...</Chart>
  <Variable>...</Variable>
  <Chart>...</Chart>
</msg:Organization>
```

Structure Of The Message Body

All OWS integration messages have the same body structure:

- An optional single default **Scope** element defining the global working context (e.g. the involved business unit).
- A sequence of top-level elements (named fragments). Each fragment may start with an optional **Scope** element (overriding the global one) followed by a sequence of inner elements containing the effective business data.

The OWS Integration Server

Introduction

An OWS integration server enables loosely coupled interactions between OWS and external applications by using asynchronous business oriented XML messages and connectors. The

syntax of the XML messages is defined by a set of schema. Connectors represent visible entry/exit points where external applications can write/read messages.

Connectors

A Connector is related to a specific transport protocol (e.g. JMS, HTTP, FTP) and presents two aspects:

- A "message source" aspect: from the OWS integration server perspective, incoming messages are read from message sources. From the point of view of an external application, its outgoing messages are written to an OWS message source.
- A "message sink" aspect: the OWS integration server writes message to those sinks. External applications read messages from these sinks.

Message Channels

A message channel is a pipeline composed of:

- A message source that provides the incoming messages.
- A message sink to which the messages are written.
- A series of nodes that can process (e.g. enrich/modify, route) the message between the message source and sink.
- An error message sink to which the processed message is routed whenever an error occurs.

A message source can belong to only one channel. In contrast, a message sink can be involved in several channels.

Nodes

A node acts as a valve in the message pipeline. Examples of nodes are:

- Validation nodes can reject (i.e. route the message to the error sink of the channel) the processed message. For instance, an XML validation node checks that an incoming/outgoing message is well formed and compliant with its associated schema.
- Transformation nodes can enrich or modify the message content. For example, an XSLT node can apply an XSLT style sheet to the processed message.
- Routing nodes can redirect the processed message to different paths (i.e. node series) depending on its type (i.e. XML namespace).

Message Processing

An integration server is configured by a broker defining a set of channels. Each channel has its own thread of control/execution and processes the message coming from its input connector independently.

Integration Server Configuration

An integration server is configured by a broker represented by an XML document. A broker defines a set of message sources, sinks, and channels. Each element of a broker has a unique identifier used by other elements to refer to it. The definition of processing node may omit the identifier. In that case a unique identifier is assigned to it.

The example below illustrates the structure of a broker document. It defines a channel linking a File message source to a File message sink. An XML validation using a catalog of schemas is used to reject invalid messages.

```
<?xml version="1.0" encoding="UTF-8"?>
<broker xmlns="http://www.oracle.com/ows/integration/server/broker"
        id="BrokerSamplet">
  <!-- ================================================================== -->
  <!-- Section: Sources -->
  <!-- ================================================================== -->
  <sources>
    <directorySource id="directorySource">
      <period>2000</period>
      <directory>d:/owsi/is/in</directory>
      <tmpDirectory>d:/owsi/is/in/tmp</tmpDirectory>
      <doneDirectory>d:/owsi/is/in/done</doneDirectory>
      <filter>xml</filter>
    </directorySource>
  </sources>

  <!-- ================================================================== -->
  <!-- Section: Sources -->
  <!-- ================================================================== -->
  <sinks>
    <directorySink id="stdErrorSink">
      <directory>d:/owsi/is/err</directory>
    </directorySink>
    <directorySink id="DirectorySink">
      <directory>d:/owsi/is/out</directory>
    </directorySink>
```

```
  </sinks>

  <!-- ==================================================================== -->
  <!-- Section: Channels -->
  <!-- ==================================================================== -->
  <channels>
    <channel id="FileToFile">
      <source id="directorySource"/>
      <flow>
        <XMLValidatorNode>
          <catalogPath>conf/catalogs/standard/catalog.xml</catalogPath>
        </XMLValidatorNode>
      </flow>
      <sink id="DirectorySink"/>
      <errorSink id="stdErrorSink"/>
    </channel>
  </channels>
</broker>
```

Directory Message Connectors

Introduction

Directory message connectors support a very simple point to point integration mechanism. An integration message is embodied by a file and a directory is used as a queue.

The following message properties are derived from the file representing the message:

Property	Value
Identifier	The file basename.
Mime Content Type	The content type is derived from the file extension. For instance, an xml extension leads to a "text/xml" content type.

A directory message source scans periodically its associated directory to discover new incoming messages. The new file is first moved to a temporary directory and then the message is processed by the channel associated with this source. If the message processing is successful, the message file is either deleted or moved to a "done" directory depending on the configuration. A timestamp can be added to the name of the message file.

NB: If you decide to keep message file in a "done" directory, pay attention to purge that directory.

Message Source Configuration

A directory message source is specified via a **directorySource** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSource	The identifier used to refer to the message source in the broker configuration.
description	element	String	0..1	none	The directory containing the incoming Organization messages	A short description used to comment the vocation of this broker element.
period	element	integer	0..1	3000	60000 (i.e. 1 minute)	The delay expressed in milliseconds between two directory scans.
directory	element	String	1..1	none	d:/ows/is/organization	The path of the directory to scan for incoming message files.
tmpDirectory	element	String	1..1	none	d:/ows/is/organization/tmp	Specify the path of the directory where the incoming message files are temporary

						moved.
doneDirectory	element	String	0..1	none	d:/ows/is/organization/done	Specify the path of the directory where the "done" message files are moved.
filter	element	String	0..1	none	xml	Specify a string used to match the extension of the incoming files. The matching is case sensitive.
doneTimestamp	element	Boolean	0..1	false	true	Specify that a timestamp should be added (before the extension) to the name of the files moved to the "done" directory.
deleteFile	element	Boolean	0..1	false	true	Specify that the incoming message file should be deleted once successfully processed.

Message Sink Configuration

A directory message sink is specified via a **directorySink** XML element. This element contains the following properties:

Name	Categor	Type	Cardinalit	Defaul	Example	Description

	y		y	t Value		
id	attribute	String	1..1	none	OrganizationMessageSink	The identifier used to refer to the message sink in the broker configuration.
description	element	String	0..1	none	The directory containing the outgoing Organization messages	A short description used to comment the vocation of this broker element.
directory	element	String	1..1	none	d:/ows/is/organization	The path of the directory where outgoing file messages are stored.

Broker Configuration Sample

The broker configuration defined below illustrates the use of directory connectors. The configuration defines a channel named "FileToFile" using:

- a directory message source named "directorySource" as input connector
- a directory message sink named "directorySink" as output connector
- a directory message sink named "stdErrorSink" as error output connector

```
<?xml version="1.0" encoding="UTF-8"?>
<broker xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.oracle.com/ows/integration/server/broker"
        id="fileConnectorSample">
  <!-- ================================================================= -->
  <!-- Section: Message Sources -->
  <!-- ================================================================= -->
  <sources>
    <directorySource id="directorySource">
      <period>2000</period>
      <directory>d:/ows/is/in</directory>
      <tmpDirectory>d:/ows/is/in/tmp</tmpDirectory>
```

```
      <doneDirectory>d:/ows/is/in/done</doneDirectory>
      <filter>xml</filter>
    </directorySource>
  </sources>

  <!-- ======================================================================= -->
  <!-- Section: Message Sinks -->
  <!-- ======================================================================= -->
  <sinks>
    <directorySink id="stdErrorSink">
      <directory>d:/ows/is/err</directory>
    </directorySink>
    <directorySink id="DirectorySink">
      <directory>d:/ows/is/out</directory>
    </directorySink>
  </sinks>

  <!-- ======================================================================= -->
  <!-- Section: Channels -->
  <!-- ======================================================================= -->
  <channels>
    <channel id="FileToFile">
      <source id="directorySource"/>
      <sink id="directorySink"/>
      <errorSink id="stdErrorSink"/>
    </channel>
  </channels>
</broker>
```

Since there is no intermediate processing node between its input and output connectors, the "FileToFile" channel simply copies files having an "xml" extension from "d:/ows/is/in" to "d:/ows/is/out".

FTP Message Connectors

Introduction

FTP message connectors rely on the File Transfer Protocol to read/write messages. As for the directory connector, a message is embodied by a file. So consult the directory connector page to have the message properties derived from the file.

A FTP message source connects periodically to an FTP server and scans a given remote directory to discover new incoming messages. The new file transferred to a local working directory and then the message is processed by the channel associated with this source. If the message processing is successful, the message file is either deleted from the remote server or moved to a remote "done" directory depending on the configuration. A timestamp can be added to the name of the message file.

Message Source Configuration

A FTP message source is specified via an **ftpSource** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSource	The identifier used to refer to the message source in the broker configuration.
description	element	String	0..1	none	The ftp source in charge of reading incoming Organization messages	A short description used to comment the vocation of this broker element.
period	element	integer	0..1	3000	60000 (i.e. 1 minute)	The delay expressed in milliseconds between two directory scans.
directory	element	String	1..1	none	/home/jdoe/is/organization	The path of

						the remote directory to scan for incoming message files.
tmpDirectory	element	String	1..1	none	d:/tmp/ows/is/organization	Specify the path of the local directory where the remote incoming message files are temporary transferred.
doneDirectory	element	String	0..1	none	d:/ows/is/organization/done	Specify the path of the remote directory where the "done" message files are moved.
filter	element	String	0..1	none	xml	Specify a string used to match the extension of the incoming files. The matching is case sensitive.
doneTimestamp	element	Boolean	0..1	false	true	Specify that a timestamp should be added (before the extension) to the name of the files moved to the "done" directory.

deleteFile	element	Boolean	0..1	false	true	Specify that the incoming message file should be deleted once successfully processed.
host	element	String	1..1	none	ftpserver.mycompany.com	The hostname of the FTP server.
port	element	Integer	0..1	20	2345	The connection port of the FTP server.
user	element	String	1..1	none	john.doe@oracle.com	The user name to login.
password	element	String	1..1	none	2bOrNot2b	the password to login.

Message Sink Configuration

A FTP message sink is specified via an **ftpSink** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSink	The identifier used to refer to the message sink in the broker configuration.
description	element	String	0..1	none	The remote directory containing the outgoing Organization messages	A short description used to comment the vocation of this broker element.

directory	element	String	1..1	none	d:/ows/is/organization	The path of the remote directory where outgoing file messages are stored.
host	element	String	1..1	none	ftpserver.mycompany.com	The hostname of the FTP server.
port	element	Integer	0..1	20	2345	The connection port of the FTP server.
user	element	String	1..1	none	jdoe	The user name to login.
password	element	String	1..1	none	2bOrNot2b	the password to login.

Broker Configuration Sample

The broker configuration defined below illustrates the use of the FTP connectors. The configuration defines a channel named "FTPToFTP" using:

- a FTP message source named "ftpSource" as input connector
- a FTP message sink named "ftpSink" as output connector
- a directory message sink named "stdErrorSink" as error output connector

```
<broker xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.oracle.com/ows/integration/server/broker"
        id="FTPConnectorSample">

  <!-- ================================================================= -->
  <!-- Section: Sources -->
  <!-- ================================================================= -->
  <sources>
```

```
  <ftpSource id="FTPSource">
    <period>2000</period>
    <directory>/home/jdoe/ows/is/in</directory>
    <tmpDirectory>d:/tmp/ows/</tmpDirectory>
    <doneDirectory>/home/jdoe/ows/is/done</doneDirectory>
    <filter>xml</filter>
    <deleteFile>false</deleteFile>
    <host>myserverftp.mycompany.com</host>
    <user>jdoe</user>
    <password>2BOrNot2B</password>
  </ftpSource>
</sources>

<!-- ======================================================================== -->
<!-- Section: Sinks -->
<!-- ======================================================================== -->
<sinks>
  <directorySink id="stdErrorSink">
    <directory>d:/temp/is/err</directory>
  </directorySink>
  <ftpSink id="FTPSink">
    <directory>/home/jdoe/ows/is/out</directory>
    <host>myserverftp.mycompany.com</host>
    <user>jdoe</user>
    <password>2BOrNot2B</password>
  </ftpSink>
</sinks>

<!-- ======================================================================== -->
<!-- Section: Channels -->
<!-- ======================================================================== -->
<channels>
<channel id="FTPToFTP">
  <source id="FTPSource"/>
  <sink id="FTPSink"/>
  <errorSink id="stdErrorSink"/>
</channel>
```

```
  </channels>
</broker>
```

Since there is no intermediate processing node between its input and output connectors, the "FTPToFTP" channel simply copies files having an "xml" extension from the remote directory "/home/jdoe/ows/is/in" to "/home/jdoe/ows/is/out" both located on the server ftp.mycompany.com.

Job Message Connectors

Introduction

Job message connectors allow to:

- Create an OWS job from an incoming message. This is the role of a job message sink. The incoming message is passed to the created job as an input data (using an attachment). OWS imports are handled this way. A job message sink is usually configured to create a job whose associated procedure imports the business objects defined by the message.
- Create outgoing messages from job results (i.e. output attachments). This is role of a job message source. It scans job results (such as export jobs) and use output attachments as outgoing messages. Once a job result is processed by a job message source, it is either marked as visited or deleted.

Since the integration server posts job to do or reads job results via the job connectors, you need some asynchronous servers running in order to process the jobs and get their results.

Message Source Configuration

A job message source is specified via a **jobSource** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	ExportManager	The identifier used to refer to the message source in the broker configuration.

description	element	String	0..1	none	The job source in charge of exporting data	A short description used to comment the vocation of this broker element.
period	element	integer	0..1	3000	60000 (i.e. 1 minute)	The delay expressed in milliseconds between two job result scans.
queue	element	Integer	1..1	none	555	The queue to scan to discover new job results to process.
procedureId	element	Integer	0..1	none	2195406	The procedure identifier used to select only job results of this procedure.
procedureName	element	String	0..1	none	OutInterfaceProcedure	The procedure name used to select only job results of this procedure.
visitMode	element	Enumeration (mark or delete)	0..1	mark	mark	Specify if the job is only marked as processed or deleted once the outgoing message is successfully created.
fragmentFilter	element	JobFragmentFilterType	0..1	none	<fragmentFilter expression="^.*Export$" regexp="true"/>	Specify the filter used to select the fragment containing the XML integration message.

Job Output Attachment

A job output attachment is made of a main document and optionally several named subdocuments (alias fragments). If the JobMessageSource does not specify a fragment filter,

the integration message is extracted from the main document. Otherwise, the fragment filter is used to select the fragment whose name matches the filter expression.

A fragment filter defines the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
expression	attribute	String	1..1	none	^.*Export$	The expression used to select the fragment based on its name. It can be either a Java regular expression or a string matching exactly (so case sensitive) the fragment name.
regexp	attribute	Boolean	1..1	false	true	Specifies if the expression is a regular expression or a literal string.

Message Sink Configuration

A job message sink is specified via a **jobSink** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	ImportManager	The identifier used to refer to the message sink in the broker configuration.
description	element	String	0..1	none	The sink creating import jobs	A short description used to comment the vocation of this broker element.
queue	element	Integer	1..1	none	555	The queue to scan to

						discover new job results to process.
procedureId	element	Integer	0..1	none	942200	The procedure name identifying the procedure associated with the jobs to create.
procedureName	element	String	0..1	none	InInterfaceProcedure	The procedure name identifying the procedure associated with the jobs to create.
context	element	Context	0..1	none	see example below	The xml definition of a working context specifying the organization, activity nodes and date scope.
internalCrypt	element	Boolean	0..1	false	true	Specify in the incoming message should stored as a crypted attachment or not. This encryption is required for incoming login messages.

Broker Configuration Sample

This broker configuration defines three different channels:

- A "FileToJob" channel allowing to create jobs processing the incoming messages delivered in the "d:/ows/is/in" directory.
- A "FileToJobCrypt" channel allowing to create jobs processing the login import messages delivered in the d:/ows/is/inCrypt directory. The login messages are encrypted before being stored in the OWS datatbase.
- A "JobToFile" channel allowing to publish in the "d:/ows/is/export" directory, the outgoing messages produced by the OWS application (for instance the schedules).

```
<broker xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.oracle.com/ows/integration/server/broker"
        id="JobConnectorSample">
  <!-- ================================================================= -->
  <!-- Section: Message Sources -->
  <!-- ================================================================= -->
  <sources>
    <directorySource id="directorySource">
      <period>2000</period>
      <directory>d:/ows/is/in</directory>
      <tmpDirectory>d:/ows/is/in/tmp</tmpDirectory>
      <doneDirectory>d:/ows/is/in/done</doneDirectory>
      <filter>xml</filter>
    </directorySource>
    <directorySource id="directorySourceCrypt">
      <period>2000</period>
      <directory>d:/ows/is/inCrypt</directory>
      <tmpDirectory>d:/ows/is/inCrypt/tmp</tmpDirectory>
      <doneDirectory>d:/ows/is/inCrypt/done</doneDirectory>
      <filter>xml</filter>
    </directorySource>
    <jobQueueSource id="JobSource">
        <period>2000</period>
        <queue>666</queue>
        <procedureName>OutInterfaceProcedure</procedureName>
        <visitMode>mark</visitMode>
```

```
      <fragmentFilter expression="^.*Export$" regexp="true"/>
   </jobQueueSource>
 </sources>

<!-- ================================================================== -->
<!-- Section: Message Sinks -->
<!-- ================================================================== -->
<sinks>
  <directorySink id="stdErrorSink">
    <directory>d:/ows/is/err</directory>
  </directorySink>
  <directorySink id="DirectorySink">
    <directory>d:/ows/is/export</directory>
  </directorySink>
  <jobQueueSink id="jobSink">
    <queue>666</queue>
    <procedureName>InInterfaceProcedure</procedureName>
    <context>
       <resourceNode>40286</resourceNode>
       <organizationNode>40001</organizationNode>
       <HROrganizationNode>40002</HROrganizationNode>
       <beginDate>1800-01-01</beginDate>
       <endDate>9999-12-31</endDate>
    </context>
  </jobQueueSink>
  <jobQueueSink id="jobSinkCrypt">
    <queue>666</queue>
    <procedureName>InInterfaceProcedure</procedureName>
    <context>
       <resourceNode>40286</resourceNode>
       <organizationNode>40001</organizationNode>
       <HROrganizationNode>40002</HROrganizationNode>
       <beginDate>1800-01-01</beginDate>
       <endDate>9999-12-31</endDate>
    </context>
    <internalCrypt>true</internalCrypt>
  </jobQueueSink>
```

```
  </sinks>

  <!-- ======================================================================== -->
  <!-- Section: Channels -->
  <!-- ======================================================================== -->
  <channels>
  <channel id="FileToJob">
    <source id="directorySource"/>
    <sink id="jobSink"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  <channel id="FileToJobCrypt">
    <source id="directorySourceCrypt"/>
    <sink id="jobSinkCrypt"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  <channel id="JobToFile">
    <source id="JobSource"/>
    <sink id="DirectorySink"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  </channels>
</broker>
```

JMS Message Connectors

Introduction

The JMS message connectors allow to use the Java Messaging Service to send/receive integration messages. The two JMS communication models are supported:

- The point to point model via queues.
- The publish/subscribe model via topics.

The JMS connectors use JNDI properties to configure the JMS connection properties.

Message Source Configuration

A JMS message source is specified via a **jmsSource** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSource	The identifier used to refer to the message source in the broker configuration.
description	element	String	0..1	none	The JMS source in charge of importing data	A short description used to comment the vocation of this broker element.
jndi	element	JNDI type	1..1	none	see example below	Specify the JNDI properties used to configure the connection and the access to JMS objects.
factory	element	String	1..1	none		The name of the factory property needed to create the JMS connections.
queue	element	String	0..1	none	OrganizationQueue	Specify the name of the JMS queue containing the message to read (only in queue mode).
topic	element	String	0..1	none	OrganizationTopic	The name of the topic to subscribe to (only in publish/subscribe mode).

Message Sink Configuration

A JMS message sink is specified via a **jmsSink** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSink	The identifier used to refer to the message sink in the broker configuration.
description	element	String	0..1	none	The queue/topic to write/publish messages.	A short description used to comment the vocation of this broker element.
jndi	element	JNDI type	1..1	none	see example below	Specify the JNDI properties used to configure the connection and the access to JMS objects.
factory	element	String	1..1	none		The name of the factory property needed to create the JMS connections.
queue	element	String	0..1	none	OrganizationQueue	Specify the name of the JMS queue where the messages have to be written.
topic	element	String	0..1	none	OrganizationTopic	The name of

						the topic used to publish the message.

Broker Configuration Sample

The broker configuration defined below illustrates the use of the JMS connectors in both the queue and publish/subscribe mode. It defines four channels:

- The JMSQueueToFile channel reads messages from the JMS queue named OrganizationQueue and writes them to the d:/ows/is/export directory.
- The FileToJMSQueue channel reads incoming message from the "d:/ows/is/in" directory and write them to the JMS OrganizationQueue.
- The JMSTopicToFile channel receives messages published to the topic name OrganizationTopic and writes them to the d:/ows/is/export1 directory
- The FileToJMSTopic channel reads messages coming from the d:/ows/is/in1 directory and publishes them to the JMS OrganizationTopic.

```
<broker xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.oracle.com/ows/integration/server/broker"
        id="JMSConnectorSample">


  <!-- ================================================================== -->
  <!-- Section: Message Sources -->
  <!-- ================================================================== -->
  <sources>
    <directorySource id="DirectorySource">
      <period>2000</period>
      <directory>d:/owsi/is/in</directory>
      <tmpDirectory>d:/owsi/is/in/tmp</tmpDirectory>
      <doneDirectory>d:/owsi/is/in/done</doneDirectory>
      <filter>xml</filter>
    </directorySource>
    <directorySource id="DirectorySource1">
      <period>2000</period>
      <directory>d:/owsi/is/in1</directory>
      <tmpDirectory>d:/owsi/is/in1/tmp</tmpDirectory>
      <doneDirectory>d:/owsi/is/in1/done</doneDirectory>
```

```
    <filter>xml</filter>
  </directorySource>
  <JMSSource id="JMSQueueSource">
     <jndi>
       <property>
         <name>java.naming.factory.initial</name>
         <value>org.apache.activemq.jndi.ActiveMQInitialContextFactory</value>
       </property>
       <property>
         <name>java.naming.provider.url</name>
         <value>tcp://myserver.mycompany.com:61616</value>
       </property>
     </jndi>
     <factory>QueueConnectionFactory</factory>
     <queue>testQueue</queue>
  </JMSSource>
  <JMSSource id="JMSTopicSource">
     <jndi>
       <property>
         <name>java.naming.factory.initial</name>
         <value>org.apache.activemq.jndi.ActiveMQInitialContextFactory</value>
       </property>
       <property>
         <name>java.naming.provider.url</name>
         <value>tcp://myserver.mycompany.com:61616</value>
       </property>
     </jndi>
     <factory>TopicConnectionFactory</factory>
     <topic>OrganizationTopic</topic>
  </JMSSource>
</sources>

<!-- ================================================================== -->
<!-- Section: Message Sinks -->
<!-- ================================================================== -->
<sinks>
  <directorySink id="stdErrorSink">
```

```
  <directory>d:/owsi/is/err</directory>
</directorySink>
<directorySink id="DirectorySink">
  <directory>d:/owsi/is/in/export</directory>
</directorySink>
<directorySink id="DirectorySink1">
  <directory>d:/owsi/is/in1/export</directory>
</directorySink>
<JMSSink id="JMSQueueSink">
   <jndi>
     <property>
       <name>java.naming.factory.initial</name>
       <value>org.apache.activemq.jndi.ActiveMQInitialContextFactory</value>
     </property>
     <property>
       <name>java.naming.provider.url</name>
       <value>tcp://wfsrh-pc.fr.oracle.com:61616</value>
     </property>
     <property>
       <name>queue.testQueue</name>
       <value>OWS.Queue</value>
     </property>
   </jndi>
   <factory>QueueConnectionFactory</factory>
   <queue>testQueue</queue>
</JMSSink>
<JMSSink id="JMSTopicSink">
   <jndi>
     <property>
       <name>java.naming.factory.initial</name>
       <value>org.apache.activemq.jndi.ActiveMQInitialContextFactory</value>
     </property>
     <property>
       <name>java.naming.provider.url</name>
       <value>tcp://wfsrh-pc.fr.oracle.com:61616</value>
     </property>
     <property>
```

```
          <name>topic.testTopic</name>
          <value>OWS.Topic</value>
        </property>
      </jndi>
      <factory>TopicConnectionFactory</factory>
      <topic>testTopic</topic>
    </JMSSink>
  </sinks>

  <!-- ================================================================== -->
  <!-- Section: Channels -->
  <!-- ================================================================== -->
  <channels>
  <channel id="JMSQueueToFile">
    <source id="JMSQueueSource"/>
    <sink id="DirectorySink"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  <channel id="FileToJMSQueue">
    <source id="DirectorySource"/>
    <sink id="JMSQueueSink"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  <channel id="JMSTopicToFile">
    <source id="JMSTopicSource"/>
    <sink id="DirectorySink1"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  <channel id="FileToJMSTopic">
    <source id="DirectorySource1"/>
    <sink id="JMSTopicSink"/>
    <errorSink id="stdErrorSink"/>
  </channel>

  </channels>
</broker>
```

Mail Message Connectors

Introduction

Mail message connectors are used to read/write integration messages from/to mailboxes. The integration message is either the body of the mail message or an attachment. A mail message source reads periodically a specified mailbox and can delete the read messages.

Message Source Configuration

A mail message source is specified via a **mailSource** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSource	The identifier used to refer to the message source in the broker configuration.
description	element	String	0..1	none	The mailbox acting as message source	A short description used to comment the vocation of this broker element.
period	element	integer	0..1	3000	60000 (i.e. 1 minute)	The delay expressed in milliseconds between two mailbox scans.
host	element	String	1..1	none	/home/jdoe/is/organization	The mail server hostname.
port	element	Integer	0..1	depends on the	3456	Specify connection

				protocol		port of the mail server.
protocol	element	String	1..1	none	IMAP	Specify the protocol to use.
user	element	String	1..1	none	john.doe@oracle.com	The connection user name.
password	element	String	1..1	none	2bOrNot2b	the connection password.
folder	element	String	1..1	none		The mailbox folder to read.
visitMode	element	Enumeration (markAsRead or delete)	0..1	none	markAsRead	delete the read message or not.
useBody	element	Boolean	0..1	false	true	use the body of the mail message as the integration or use the attachment.

Message Sink Configuration

A mail message sink is specified via a **mailSink** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSink	The identifier used to refer to the message sink in the broker configuration.

description	element	String	0..1	none	The mail reference acting as message sink	A short description used to comment the vocation of this broker element.
host	element	String	1..1	none	/home/jdoe/is/organization	The mail server hostname.
port	element	Integer	0..1	depends on the protocol	3456	Specify connection port of the mail server.
protocol	element	String	1..1	none	IMAP	Specify the protocol to use.
to	element	String	1..1	none	jdoe	The address used to send the mail.
subject	element	String	0..1	none	exported OWS schedules	The subject of the mail message.

Broker Configuration Sample

The broker configuration presented below illustrates the use of mail connectors. It defines two channels:

- FileToMail used to send incoming messages to a specific mail address.
- MailToFile used to store in a directory the messages contained in a given mailbox.

```
<broker xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.oracle.com/ows/integration/server/broker"
        id="mailConnectorSample">

  <!-- ================================================================= -->
  <!-- Section: Sources -->
  <!-- ================================================================= -->
```

```
<sources>
  <directorySource id="DirectorySource">
    <period>2000</period>
    <directory>d:/ows/is/in</directory>
    <tmpDirectory>d:/ows/is/in/tmp</tmpDirectory>
    <doneDirectory>d:/ows/is/in/done</doneDirectory>
    <filter>xml</filter>
  </directorySource>
  <mailSource id="MailSource">
    <period>2000</period>
    <host>mymailserver.mycompany.com</host>
    <protocol>IMAP</protocol>
    <user>john.doe</user>
    <password>2BOrNot2B</password>
    <folder></folder>
  </mailSource>
</sources>

<!-- ======================================================================== -->
<!-- Section: Sinks -->
<!-- ======================================================================== -->
<sinks>
  <directorySink id="stdErrorSink">
    <directory>d:/ows/is/err</directory>
  </directorySink>
  <directorySink id="DirectorySink">
    <directory>d:/ows/is/out</directory>
  </directorySink>
  <mailSink id="MailSink">
    <host>mymailserver.mycompany.com</host>
    <protocol>IMAP</protocol>
    <to>john.doe@mycompany.com</to>
    <subject>OWS schedules</subject>
  </mailSink>
</sinks>

<!-- ======================================================================== -->
```

```
  <!-- Section: Channels -->
  <!-- ================================================================= -->
  <channels>
  <channel id="MailToFile">
    <source id="MailSource"/>
    <sink id="DirectorySink"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  <channel id="FileToMail">
    <source id="DirectorySource"/>
    <sink id="MailSink"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  </channels>
</broker>
```

HTTP Message Connectors

Introduction

These connectors allow to use the HTTP protocol to send/receive integration messages:

- The HTTP message source manages incoming message and can work in two modes:
    - A client mode. In this mode it relies on an HTTP client and periodically emits an HTTP request to get the incoming integration message as part of the body of the http response.
    - A server mode. In this mode, it receives HTTP requests emitted by external applications. The integration message is contained in the body of the http requests. This mode is supported by embedding an HTTP server in the OWS integration server.
- The HTTP message sink posts integration message as bodies of HTTP requests. Thus it acts as an HTTP client.

Message Source Configuration

An HTTP message source is specified via a **httpSource** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSource	The identifier used to refer to the message source in the broker configuration.
description	element	String	0..1	none	The job source in charge of importing data	A short description used to comment the vocation of this broker element.
period	element	integer	0..1	3000	60000 (i.e. 1 minute)	The delay expressed in milliseconds between emitting two http requests. Only used in "client" mode.
server	element	Boolean	0..1	false	true	Specify it the source acts relies on an HTTP client or server.
host	element	String	0..1	none	localhost	The http server used to send the request (client mode only).
port	element	Integer	0..1	80	8080	The connection port of the target http server (client mode only).

url	element	String	1..1	none	/owsi/export.owsi	In client mode, specify the requests. In server mode, specify the URL associated with this source.

Message Sink Configuration

A FTP message sink is specified via a **httpSink** XML element. This element contains the following properties:

Name	Category	Type	Cardinality	Default Value	Example	Description
id	attribute	String	1..1	none	OrganizationMessageSink	The identifier used to refer to the message sink in the broker configuration.
description	element	String	0..1	none	The service to invoke to post Organization messages	A short description used to comment the vocation of this broker element.
host	element	String	0..1	none	localhost	The http server used to send the request.
port	element	Integer	0..1	80	8080	The connection port of the target http server.

url	element	String	1..1	none	/owsi/export.owsi	The url to use to send the message.

Broker Configuration Sample

The broker configuration shown below defines three channels:

- The HTTPServerSourceToFile channel uses an http message source configured in server mode. The incoming messages are produced by external applications emitting http request containing the path "/owsi/put.owsi".
- The HTTPClientSourceToFile channel uses an http message source configured in client mode. Therefore it emits the "/owsi/getMessage" to the http server whose address is localhost:9090.
- The FileToHttpSink channel reads incoming messages delivered in the "d:/ows/in" directory and transmits them to the localhost:9090 http server.

```
<broker xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.oracle.com/ows/integration/server/broker"
        id="HTTPConnectorSample">


  <!-- ================================================================= -->
  <!-- Section: Sources -->
  <!-- ================================================================= -->
  <sources>
    <directorySource id="DirectorySource">
      <period>2000</period>
      <directory>d:/ows/is/in</directory>
      <tmpDirectory>d:/ows/is/in/tmp</tmpDirectory>
      <doneDirectory>d:/ows/is/in/done</doneDirectory>
      <filter>xml</filter>
    </directorySource>
    <httpSource id="HTTPServerSource">
       <server>true</server>
       <url>/owsi/put.owsi</url>
    </httpSource>
    <httpSource id="HTTPClientSource">
```

```
      <period>2000</period>
      <server>false</server>
      <host>localhost</host>
      <port>9090</port>
      <url>/owsi/getMessage</url>
  </httpSource>
</sources>

<!-- ======================================================================== -->
<!-- Section: Sinks -->
<!-- ======================================================================== -->
<sinks>
  <directorySink id="stdErrorSink">
    <directory>d:/ows/is/err</directory>
  </directorySink>
  <directorySink id="DirectorySink">
    <directory>d:/ows/is/export</directory>
  </directorySink>
  <httpSink id="HTTPClientSink">
     <host>localhost</host>
     <port>9090</port>
     <url>/owsi/putMessage</url>
  </httpSink>
</sinks>

<!-- ======================================================================== -->
<!-- Section: Channels -->
<!-- ======================================================================== -->
<channels>
<channel id="HTTPServerSourceToFile">
  <source id="HTTPServerSource"/>
  <sink id="DirectorySink"/>
  <errorSink id="stdErrorSink"/>
</channel>

<channel id="HTTPClientSourceToFile">
  <source id="HTTPClientSource"/>
```

```
    <sink id="DirectorySink"/>
    <errorSink id="stdErrorSink"/>
  </channel>

  <channel id="FileToHTTPSink">
    <source id="DirectorySource"/>
    <sink id="HTTPClientSink"/>
    <errorSink id="stdErrorSink"/>
  </channel>
  </channels>
</broker>
```

Integration Message Categories

Incoming and outgoing messages

The OWS application deals with two kinds of integration messages:

- **Incoming** (or IN) messages published by external applications and consumed by the OWS application.

  The following table illustrates some use cases of incoming messages.

Publisher	Use Case
POS	Publishing charts of the number of customers at POS by store/department every 15 minutes
HR	Publishing employee hiring, contracts, assignments
Payroll	Publishing dollars paid per employee
T&A	Publishing hours paid per employee
Store	Publishing dated properties of stores and departments (square footage, etc.)
Merchandising	Publishing charts of number of pallets delivered by trucks, every 60 minutes

- **Outgoing** (or OUT messages) published by the OWS application and consumed by external applications.

  The following table illustrates some use cases of outgoing messages.

Publisher	Use Case
Clocking Devices	Subscribing to forecast schedules per employee
HR	Subscribing to hiring requests entered in advance in Oracle Workforce Scheduling for better scheduling
Payroll	Subscribing to forecast schedules per employee
T&A	Subscribing to forecast schedules per employee
Store	Subscribing to KPIs for stores and departments
Data warehouse	Subscribing to KPIs for stores, departments and forecast employees and schedules for employees

Integration Message Types

The OWS application supports several integration message types. A message type is embodied by an XML schema associated with a specific namespace.

Organization Message

Organization messages are incoming/outgoing messages and can contain **Chart**, **Variable**, and **TimeWindow Cycle**, **Event** and **EventAssignment** fragments, for organization units, such as districts, stores, departments, etc.

Outgoing Organization messages mainly contains **Charts** for exporting forecast, actuals and KPI.

Employee Message

Employee messages contain fragments on employees (contracts) or employee files:

- PersonIdentification: This is a variable fragment for key personal information, such as first name, last name, social security number, etc., which is valid at all times.
- Address, Contact: These are variable fragments relating to the information of a person, which may vary over time.
- Hiring: This concerns the hire date of an employee; customized fields can be added.
- Contract: This concerns all dated fields related to an employee contract; customized fields can be added.

- Assignment: This is a variable describing the assignment history in the organization for an employee contract.
- Chart: This provides the actual hours completed and the dollars paid for an employee contract period.
- Schedule: This fragment gives the absence forecast for an employee contract.
- Skill: This fragment gives the ordered list of skills for an employee contract.
- Availability: This fragment gives the fixed work hours and the preferences for an employee contract.

Schedules

Schedules in the HRSchedule outgoing message denote the optimized schedules computed by OWS.

Integration Message Processing

Introduction

The OWS integration layer deals with message fragments (i.e. toplevel elements) in a transactional way: a fragment is either completely consumed/produced or not at all.

Several kinds of error may occur in the various steps of message processing. They are handled and reported differently.

Technical Errors

Technical errors that prevent the message to be consumed / produced are reported as SOAP Faults. These errors are related to the acquisition of the needed resources to process (i.e. consume/produce) the message:

- The message itself due to some communication errors.
- A connection to the OWS database or computing resource (e.g. asynchronous job execution context).

For incoming messages, the SOAP fault is part of the body of the reply message (if required) and is related to the incoming message using the correlation identifier. For outgoing messages, the SOAP fault replaces the business body of the message.

XML Validation Errors

Syntax errors are caused when the message content is not well formed from an XML point of view (e.g. an opening tag without a matching closing tag) or does not respect the rules defined

by the associated schema (e, the element sequence is not compliant with the schema (e.g. a date is not in the date format).

- Non-compliance with XML syntax or the overall structure will interrupt message parsing. Therefore, the end of message is rejected since the last fragment was not completely parsed and processed.
- Non-compliance with a text element or attribute syntax value causes rejection of that particular fragment, but subsequent fragments will be processed.

Functional Errors

An example of a functional error is the arrival of a **Hiring** fragment related to a non-existent employee (not yet having received a **PersonIdentification** fragment for that employee). The fragment is rejected but subsequent fragments are processed normally. A fragment update is not possible. Errors are reported in a functional **Trace** element. This element also contains:

- Warnings The fragment is processed and updates are done. But, the integration layer detected a particular situation that is potentially a mistake. Typically, an address fragment implying that an employee no longer has an address is a warning. This may be possible, but is often a mistake.
- Information

  The fragment is processed. Information data provides details on what has been done, and a trace is processed for auditing purposes.

Message Building Policies

The OWS integration message schemas have been designed to be flexible in order to cover different used cased both in terms of granularity or element grouping. This section illustrates some use cases for **Organization** and **Employee** messages.

Organization Messages

The grouping policy of organization message elements can be:

- Grouping by organization unit: A message can be related to only one unit or have fragments for several units.
- Grouping by data type: If, for example, a specific sub-system manages all transactions and sales charts, a weekly grouping by type of chart can be considered.
- Non-grouping of some messages: Typically, some messages can be generated as soon as a variable changes for a store, and can be sent on a "real-time" basis.

Employee Messages

Here are some examples of grouping policy for employee messages:

- In a real-time HR sub-system, you may group three or more fragments together, such as **PersonalIdentification**, **Address**, and **Contact** by employee. The HR system triggers a message on an employee change and as a result messages are sent on a near real-time basis.

- In a non-real-time HR sub-system, you may use a different policy. Every day, the system detects differences between the current day and the previous day. A single daily message groups all contract and variable changes, concerning all employees in a single daily transmission. The system sorts all fragments by employee. They do not reflect detailed changes concerning employees, but only the overall difference between two consecutive days.

- Charts on actual and accrual counters are issued from a completely separate payroll system. One message is issued each week for all employees and for each type of chart.

- Generate actual and accrual charts per store, as this data is managed per store in several payroll systems. Generate one message on a weekly basis for each store with data of all employees assigned to the store and paid by that store.

- Absence forecasts are managed in an in-house T&A sub-system. Therefore, every night, newly created or changed absences are detected and sent in a single message for all employees.

- The designer uses a trigger feature on the real time T&A sub-system. As soon as an employee makes a change concerning a forecasted absence, a message is generated for that employee that includes all forecasted absences ranging from a month earlier through the following year. One message per employee is sent on a near real-time basis.

OWS IDK 5.0.3 Schemas Overview

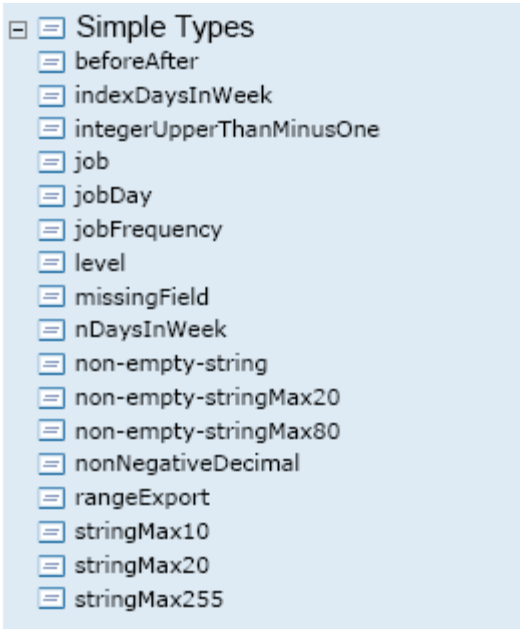
Content Summary

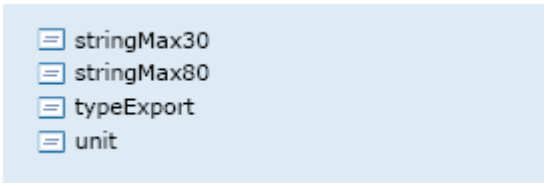
All Definitions

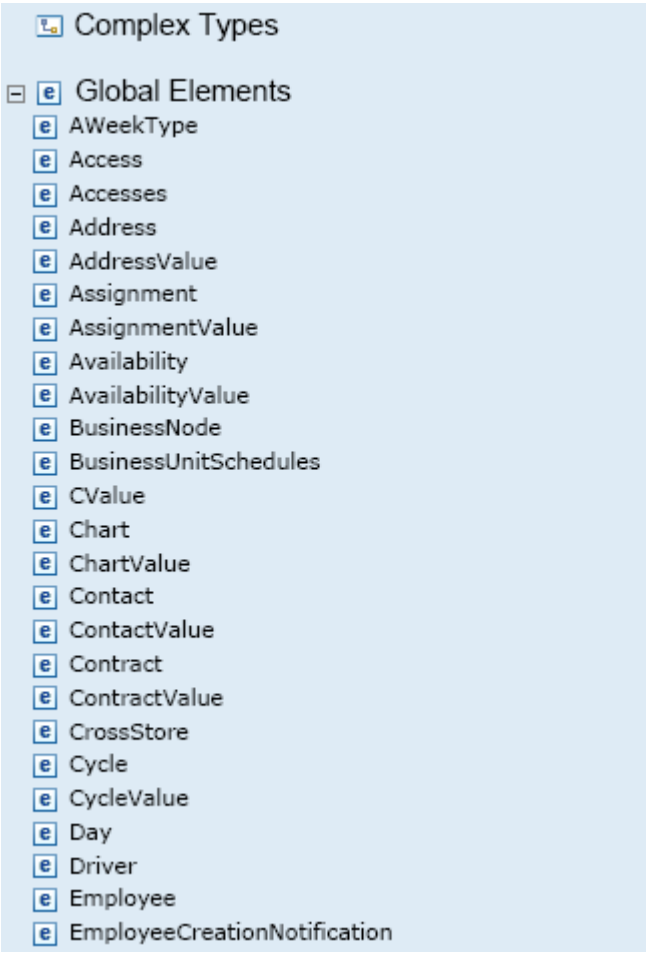
Schemas

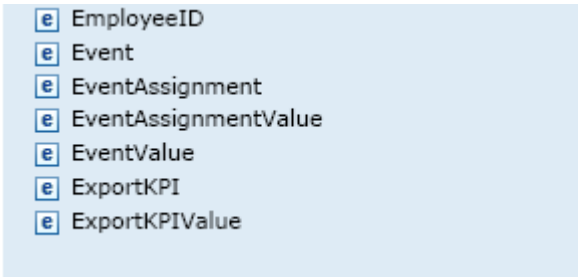
- Core [http://www.oracle.com/ows/idk/Core]
- Login [http://www.oracle.com/ows/idk/Login]
- Employee [http://www.oracle.com/ows/idk/Employee]
- EmployeeCreationNotification [http://www.oracle.com/ows/idk/EmployeeCreationNotification]
- Organization [http://www.oracle.com/ows/idk/Organization]
- BusinessUnitSchedule [http://www.oracle.com/ows/idk/BusinessUnitSchedule]
- HRSchedule [http://www.oracle.com/ows/idk/HRSchedule]
- KPI [http://www.oracle.com/ows/idk/KPI]

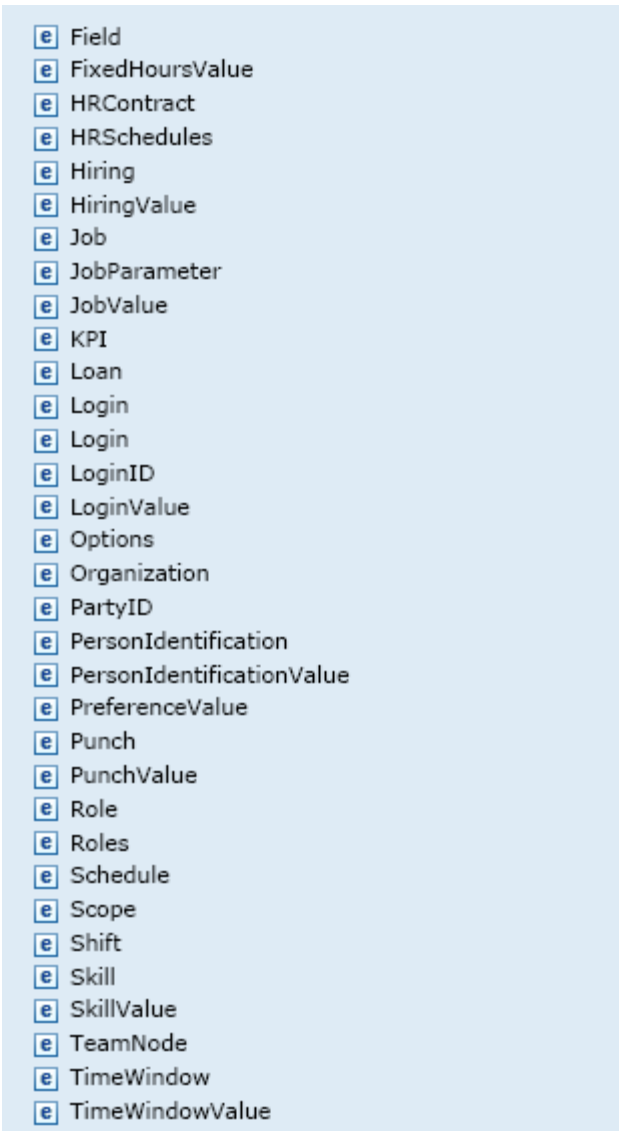
Schema List	
Name	**Description**
Core http://www.oracle.com/ows/idk/Core	Defines the common items used in the OWS IDK messages.
Login http://www.oracle.com/ows/idk/Login	Provides the definitions used in the "login" messages.
Employee http://www.oracle.com/ows/idk/Employee	Provides the definitions used in the employee messages.
EmployeeCreationNotification http://www.oracle.com/ows/idk/EmployeeCreationNotification	Provides the definitions used in the employee creation notification messages.
Organization http://www.oracle.com/ows/idk/Organization	Provides the definitions used in the organization messages.
BusinessUnitSchedule http://www.oracle.com/ows/idk/BusinessUnitSchedule	Provides the definitions used in the schedule messages denoting the schedules computed by the OWS application.
HRSchedule http://www.oracle.com/ows/idk/HRSchedule	Provides the definitions used in the HRSchedule messages.
KPI http://www.oracle.com/ows/idk/KPI	Provides the definitions used in the "KPI" messages.

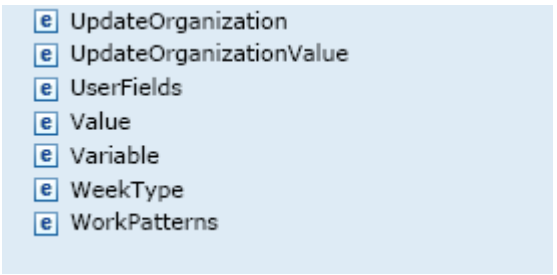


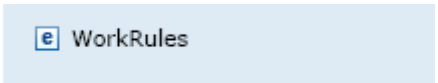

stringMax30
stringMax80
typeExport
unit


Complex Types
Global Elements
AWeekType
Access
Accesses
Address
AddressValue
Assignment
AssignmentValue
Availability
AvailabilityValue
BusinessNode
BusinessUnitSchedules
CValue
Chart
ChartValue
Contact
ContactValue
Contract
ContractValue
CrossStore
Cycle
CycleValue
Day
Driver
Employee
EmployeeCreationNotification


EmployeeID
Event
EventAssignment
EventAssignmentValue
EventValue
ExportKPI
ExportKPIValue


Field
FixedHoursValue
HRContract
HRSchedules
Hiring
HiringValue
Job
JobParameter
JobValue
KPI
Loan
Login
Login
LoginID
LoginValue
Options
Organization
PartyID
PersonIdentification
PersonIdentificationValue
PreferenceValue
Punch
PunchValue
Role
Roles
Schedule
Scope
Shift
Skill
SkillValue
TeamNode
TimeWindow
TimeWindowValue


UpdateOrganization
UpdateOrganizationValue
UserFields
Value
Variable
WeekType
WorkPatterns


WorkRules

Core Schema Overview

Content Summary

Simple Type List	
Name	**Description**
beforeAfter	
indexDaysInWeek	This type is used to identify a day in a week. The range starts from the first day of week as specified in the CCD.
integerUpperThanMinusOne	This type denotes integer values >= -1.
job	
jobDay	
jobFrequency	
level	The level of detail associated with a shift.
missingField	
nDaysInWeek	This type is used to denote the number of days in a week.
non-empty-string	This type denotes string containing at least one character.
non-empty-stringMax20	This type is equivalent to char[20] with at least a character
non-empty-stringMax80	This type is equivalent to char[80] with at least a character.
nonNegativeDecimal	This type denotes decimal values >= 0.
rangeExport	
stringMax10	This type is equivalent to char[10].
stringMax20	This type is equivalent to char[20].
stringMax255	This type is equivalent to char[255].

stringMax30	This type is equivalent to char[30].
stringMax80	This type is equivalent to char[80].
typeExport	
unit	

Complex Type List

Name	Description
There is no complex type.	

Global Element List

Name	Description
AWeekType	
Access	
Accesses	
Address	The Address element is related to an employee and gives the history of the employee's addresses over time.
AddressValue	The AddressValue element defines the address of an employee.
Assignment	This record is related to an employee's file (work period) and gives the history of the employee's successive assignments for each business unit (team hierarchy).
AssignmentValue	
Availability	The Availability element provides the available, preferred and fixed work hours of an employee, for each day in the defined temporal scope.
AvailabilityValue	
BusinessNode	
CValue	
Chart	

ChartValue	
Contact	
ContactValue	The ContactValue element is related to an employee and gives the history of the employee's contact information over time.
Contract	
ContractValue	
CrossStore	
Cycle	
CycleValue	
Day	
Driver	
EmployeeID	
Event	
EventAssignment	
EventAssignmentValue	
EventValue	
ExportKPI	
ExportKPIValue	
Field	
FixedHoursValue	
HRContract	
Hiring	
HiringValue	The HiringValue element is related to an employee and gives the history of the employee's hiring events.
Job	
JobParameter	

JobValue	
Loan	
Login	
LoginID	
LoginValue	
Options	
PartyID	
PersonIdentification	
PersonIdentificationValue	
PreferenceValue	
Punch	
PunchValue	
Role	
Roles	
Schedule	
Scope	
Shift	
Skill	The Sill element is related to an employee's file (work period) and gives the sequence of the employee's list of skills for that file.
SkillValue	
TeamNode	
TimeWindow	The TimeWindow element describes the time slots for each day of a typical week.
TimeWindowValue	
UpdateOrganization	
UpdateOrganizationValue	

UserFields	
Value	
Variable	
WeekType	
WorkPatterns	
WorkRules	

Login Schema Overview

Content Summary

Simple Type List	
Name	**Description**
There is no simple type.	

Complex Type List	
Name	**Description**
There is no complex type.	

Global Element List	
Name	**Description**
Login	

Employee Schema Overview

Content Summary

Simple Type List	
Name	**Description**
There is no simple type.	

Complex Type List	
Name	**Description**
There is no complex type.	

Global Element List	
Name	**Description**
Employee	

EmployeeCreationNotification Schema Overview

Content Summary

Simple Type List	
Name	**Description**
There is no simple type.	

Complex Type List	
Name	**Description**
There is no complex type.	

Global Element List	
Name	**Description**
EmployeeCreationNotification	

Organization Schema Overview

Content Summary

Simple Type List	
Name	**Description**
There is no simple type.	

Complex Type List	
Name	**Description**
There is no complex type.	

Global Element List	
Name	**Description**
Organization	

BusinessUnitSchedule Schema Overview

Content Summary

Simple Type List	
Name	**Description**
There is no simple type.	

Complex Type List	
Name	**Description**
There is no complex type.	

Global Element List	
Name	**Description**
BusinessUnitSchedules	

HRSchedule Schema Overview

Content Summary

Simple Type List	
Name	**Description**
There is no simple type.	

Complex Type List	
Name	**Description**
There is no complex type.	

Global Element List	
Name	**Description**
HRSchedules	

KPI Schema Overview

Content Summary

Simple Type List	
Name	**Description**
There is no simple type.	

Complex Type List	
Name	**Description**
There is no complex type.	

Global Element List	
Name	**Description**
KPI	

Core Simple Types

Name	Description	Definition
beforeAfter		<xs:simpleType name="beforeAfter"> <xs:restriction base="xs:string"> <xs:enumeration value="Before"/> <xs:enumeration value="After"/> </xs:restriction> </xs:simpleType>
indexDaysInWeek	This type is used to identify a day in a week. The range starts from the first day of week as specified in the CCD.	<xs:simpleType name="indexDaysInWeek"> <xs:restriction base="xs:integer"> <xs:minInclusive value="0"/> <xs:maxInclusive value="6"/> </xs:restriction> </xs:simpleType>
integerUpperThanMinusOne	This type denotes integer values >= -1.	<xs:simpleType name="integerUpperThanMinusOne"> <xs:restriction base="xs:integer"> <xs:minInclusive value="-1"/> </xs:restriction> </xs:simpleType>
job		<xs:simpleType name="job"> <xs:restriction base="xs:string"> <xs:enumeration value="KpiExport"/> <xs:enumeration value="ScheduleExport"/> <xs:enumeration value="Fire"/>

		<xs:enumeration value="Demand"/> <xs:enumeration value="Check"/> <xs:enumeration value="Forecast"/> <xs:enumeration value="EarnedHours"/> </xs:restriction> </xs:simpleType>
jobDay		<xs:simpleType name="jobDay"> <xs:restriction base="xs:integer"> <xs:minInclusive value="-1"/> <xs:maxInclusive value="31"/> </xs:restriction> </xs:simpleType>
jobFrequency		<xs:simpleType name="jobFrequency"> <xs:restriction base="xs:string"> <xs:enumeration value="Daily"/> <xs:enumeration value="Weekly"/> <xs:enumeration value="Monthly"/> </xs:restriction> </xs:simpleType>
level	The level of detail associated with a shift.	<xs:simpleType name="level"> <xs:restriction base="xs:string"> <xs:enumeration value="detail"/> <xs:enumeration value="day"/> <xs:enumeration value="week"/> </xs:restriction>

		</xs:simpleType>
missingField		<xs:simpleType name="missingField"> <xs:restriction base="xs:string"> <xs:enumeration value="null"/> <xs:enumeration value="default"/> <xs:enumeration value="nothing"/> </xs:restriction> </xs:simpleType>
nDaysInWeek	This type is used to denote the number of days in a week.	<xs:simpleType name="nDaysInWeek"> <xs:restriction base="xs:integer"> <xs:minInclusive value="0"/> <xs:maxInclusive value="7"/> </xs:restriction> </xs:simpleType>
non-empty-string	This type denotes string containing at least one character.	<xs:simpleType name="non-empty-string"> <xs:restriction base="xs:string"> <xs:minLength value="1"/> </xs:restriction> </xs:simpleType>
non-empty-stringMax20	This type is equivalent to char[20] with at least a character	<xs:simpleType name="non-empty-stringMax20"> <xs:restriction base="non-empty-string"> <xs:maxLength value="20"/> </xs:restriction> </xs:simpleType>
non-empty-stringMax80	This type is equivalent to char[80] with at least a character.	<xs:simpleType name="non-empty-stringMax80"> <xs:restriction base="non-empty-string"> <xs:maxLength value="80"/>

		</xs:restriction> </xs:simpleType>
nonNegativeDecimal	This type denotes decimal values >= 0.	<xs:simpleType name="nonNegativeDecimal"> <xs:restriction base="xs:decimal"> <xs:minInclusive value="0"/> </xs:restriction> </xs:simpleType>
rangeExport		<xs:simpleType name="rangeExport"> <xs:restriction base="xs:string"> <xs:enumeration value="Week"/> <xs:enumeration value="Day"/> </xs:restriction> </xs:simpleType>
stringMax10	This type is equivalent to char[10].	<xs:simpleType name="stringMax10"> <xs:restriction base="xs:string"> <xs:maxLength value="10"/> </xs:restriction> </xs:simpleType>
stringMax20	This type is equivalent to char[20].	<xs:simpleType name="stringMax20"> <xs:restriction base="xs:string"> <xs:maxLength value="20"/> </xs:restriction> </xs:simpleType>
stringMax255	This type is equivalent to char[255].	<xs:simpleType name="stringMax255"> <xs:restriction base="xs:string"> <xs:maxLength value="255"/> </xs:restriction> </xs:simpleType>
stringMax30	This type is equivalent to	<xs:simpleType name="stringMax30">

	char[30].	<xs:restriction base="xs:string"> <xs:maxLength value="30"/> </xs:restriction> </xs:simpleType>
stringMax80	This type is equivalent to char[80].	<xs:simpleType name="stringMax80"> <xs:restriction base="xs:string"> <xs:maxLength value="80"/> </xs:restriction> </xs:simpleType>
typeExport		<xs:simpleType name="typeExport"> <xs:restriction base="xs:string"> <xs:enumeration value="Summary"/> <xs:enumeration value="Detail"/> <xs:enumeration value="Distribution"/> </xs:restriction> </xs:simpleType>
unit		<xs:simpleType name="unit"> <xs:restriction base="xs:string"> <xs:enumeration value="Day"/> <xs:enumeration value="Week"/> <xs:enumeration value="Month"/> </xs:restriction> </xs:simpleType>

Availability Element

Description

The **Availability** fragment is attached to an employee's file. Its purpose is to define preferred and fixed work hours for every day, in the temporal scope. Preferred and fixed hours are given on a weekly basis. The rule is that you need to give a sequence of weeks (for example, 4 weeks) and then to repeat that sequence for the overall scope. For example, if you have defined the duration of the scope as 12 weeks, and the initial sequence has 4 weeks, then that sequence is repeated thrice.

Each week defined in the initial sequence has a name, for example:

- EveningWeek which specifies the preferred and fixed hours, when the employee works after 8:00PM in that week.
- MorningWeek which specifies the preferred and fixed hours, when the employee works before 10:00AM in that week.
- DayWeek which specifies the preferred and fixed hours, when the employee works after 8:00AM in that week.

However, a same week may be repeated more than once in the sequence. For example, the week sequence could be EveningWeek DayWeek DayWeek MorningWeek. In that case the description of the DayWeek is not to be given twice:

- The first appearance in the sequence gives the full description
- The second appearance gives an empty description (already known).

```
<Availability name="Rota1" >
  <Scope>
    <EmployeeID IDType="HRID" fileID="1" >002145</EmployeeID>
    <StartDate>2004-01-05</StartDate>
    <EndDate>2004-03-28</EndDate> <!-- 12 Weeks -->
  </Scope>
  <AWeekType name="EveningWeek">
    <AvailibiltyValue> <!-- Day 2 -->
        ...
```

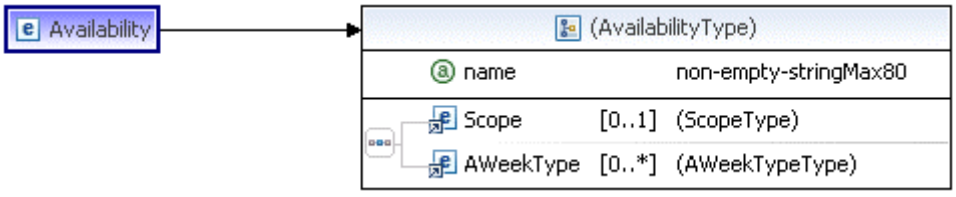
```
    </AvailabilityValue>
    <AvailibiltyValue> <!-- Day 3 -->
        ...
    </AvailabilityValue>
    <AvailibiltyValue> <!-- Day 4 -->
        ...
    </AvailabilityValue>
  </AWeekType>
  <AWeekType name="DayWeek">
    <AvailibiltyValue> <!-- Day 0 -->
        ...
    </AvailabilityValue>
    <AvailibiltyValue> <!-- Day 6 -->
        ...
    </AvailabilityValue>
  </AWeekType>
  <AWeekType name="DayWeek" /> <!-Already described -->
  <AWeekType name="MorningWeek">
    <AvailibiltyValue> <!-- Day 0 -->
        ...
    </AvailabilityValue>
    <AvailibiltyValue> <!-- Day 1 -->
        ...
    </AvailabilityValue>
    <AvailibiltyValue> <!-- Day 5 -->
        ...
    </AvailabilityValue>
  </AWeekType>
</Availability>
```

AvailabilityValue element
Each element describes the preferred and fixed hours for one day of the week. The index of the day is 0 to 6. Since an employee does not usually work every day in a week, only some of the 7 indexes are present.

```
<AvailabilityValue>
  <Index>0</Index>
  <Interval>0</Interval>
  <Start>08:00:00</Start>
```

```
  <End>12:00:00</End>
</AvailabilityValue>
<AvailabilityValue>
  <Index>0</Index>
  <Interval>1</Interval>
  <Start>14:00:00</Start>
  <End>20:00:00</End>
  <PreferenceValue>
    <Start>15:00:00</Start>
    <End>17:00:00</End>
  </PreferenceValue>
</AvailabilityValue>
<AvailabilityValue>
  <Index>1</Index>
  <Start>02:02:01</Start>
  <End>12:02:01</End>
  <PreferenceValue>
  <Start>02:02:01</Start>
  <End>12:02:01</End>
  </PreferenceValue>
    <FixedHoursValue>
    <Start>02:02:01</Start>
    <End>12:02:01</End>
  </FixedHoursValue>
</AvailabilityValue>
```

Definition




```
<xs:element name="Availability">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
```

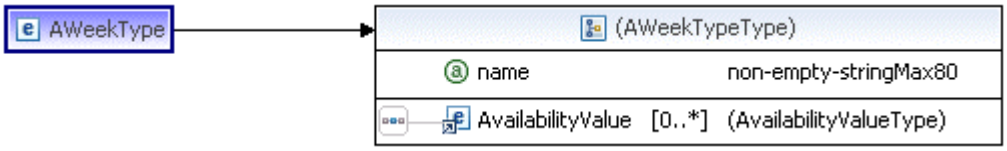
```
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="AWeekType"/>
      </xs:sequence>
      <xs:attribute name="name" type="non-empty-stringMax80" use="required"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
name	attribute	non-empty-stringMax80	1	
Scope	element	Scope	0..1	
AWeekType	element	AWeekType	0..*	

AWeekType Element

AWeekType Definition




```
<xs:element name="AWeekType">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded"
ref="AvailabilityValue"/>
      </xs:sequence>
      <xs:attribute name="name" type="non-empty-stringMax80" use="required"/>
   </xs:complexType>
</xs:element>
```

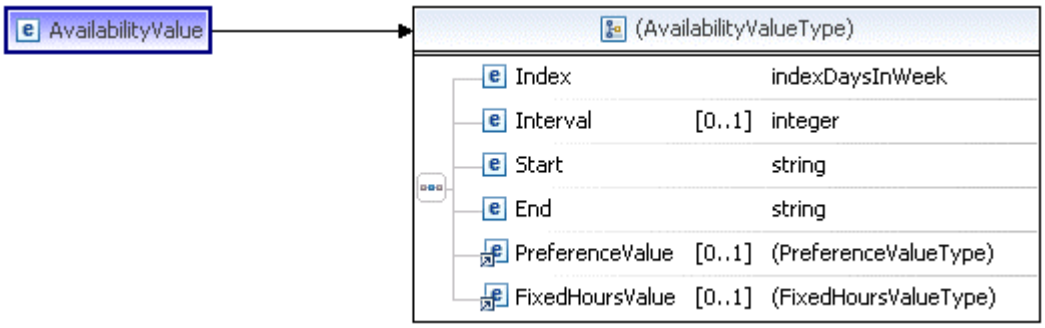
AWeekType Content Description

Name	Category	Type	Cardinality	Description

name	attribute	non-empty-stringMax80	1	
AavailabilityValue	element	AvailabilityValue	0..*	

AvailabilityValue Element

AvailabilityValue Definition




```
<xs:element name="AvailabilityValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Index" type="indexDaysInWeek"/>
         <xs:element name="Interval" type="xs:integer" minOccurs="0"/>
         <xs:element name="Start" type="xs:string"/>
         <xs:element name="End" type="xs:string"/>
         <xs:element minOccurs="0" ref="PreferenceValue"/>
         <xs:element minOccurs="0" ref="FixedHoursValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

AvailabilityValue Content Description

Name	Category	Type	Cardinality	Description
Index	element	indexDaysInWeek	1..1	
Interval	element	xs:integer	0..1	

Start	element	xs:string	1..1	The start time of the availability.
End	element	xs:string	1..1	The end time of the availability.
PreferenceValue	element	PreferenceValue	0..1	The preferred hours associated with the availability.
FixedHoursValue	element	FixedHoursValue	0..1	The fixed hours associated with the availability.

Accesses Element

Definition

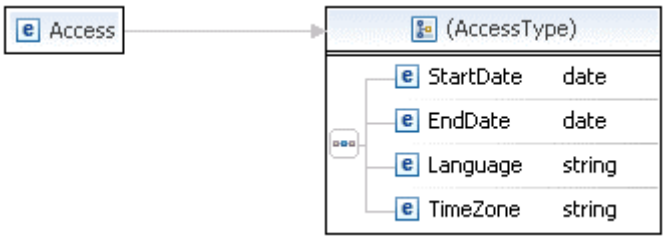

```
<xs:element name="Accesses">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Access"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Access	element	Access	0..*	

Access Element

Access Definition




```
<xs:element name="Access">
   <xs:complexType>
```

```
      <xs:sequence>
         <xs:element name="StartDate" type="xs:date"/>
         <xs:element name="EndDate" type="xs:date"/>
         <xs:element name="Language" type="xs:string"/>
         <xs:element name="TimeZone" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Access Content Description

Name	Category	Type	Cardinality	Description
StartDate	element	xs:date	1..1	
EndDate	element	xs:date	1..1	
Language	element	xs:string	1..1	
TimeZone	element	xs:string	1..1	

Address Element

Description

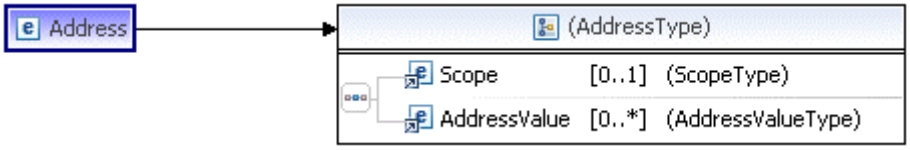
The **Address** fragment is attached to an employee, and gives a history (via the **AddressValue** element) of all the fields attached to an address.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg:Employee xmlns:msg="http://www.oracle.com/ows/idk/Employee"
                 xmlns="http://www.oracle.com/ows/idk/Core" >

<Address>
  <Scope>
    <EmployeeID IDType="HRID">049996</EmployeeID>
    <StartDate>2004-01-01</StartDate>
    <EndDate>2004-12-31</EndDate>
  </Scope>
  <AddressValue>
    <Date>2004-01-04</Date>
    <Street>3 ChampsElysees</Street>
        ...
    </AddressValue>
    <AddressValue>
      <Date>2004-01-04</Date>
      <Street>3 rue de La Paix</Street>
        ...
    </AddressValue>
  </Address>
</msg:Employee>
```

NB: The **Date** field is mandatory.

Definition



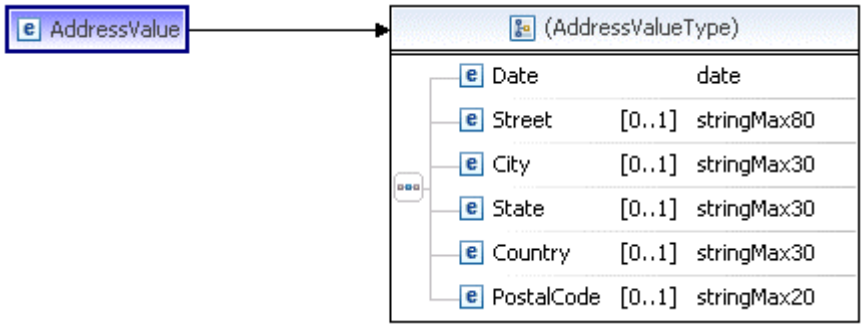

```
<xs:element name="Address">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="AddressValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	0..1	
AddressValue	element	AddressValue	0..*	

AddressValue Element

AddressValue Definition




```
<xs:element name="AddressValue">
   <xs:complexType>
```

```
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="Street" type="stringMax80" minOccurs="0"/>
         <xs:element name="City" type="stringMax30" minOccurs="0"/>
         <xs:element name="State" type="stringMax30" minOccurs="0"/>
         <xs:element name="Country" type="stringMax30" minOccurs="0"/>
         <xs:element name="PostalCode" type="stringMax20" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

AddressValue Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	1..1	Effective date of the value.
Street	element	stringMax80	0..1	The street where the employee resides.
City	element	stringMax30	0..1	The city where the employee resides.
State	element	stringMax30	0..1	The state or province where the employee resides.
Country	element	stringMax30	0..1	The employee's country of residence.
PostalCode	element	stringMax20	0..1	The postal code relevant for the country.

Assignment Element

Description

Assignment fragments refer to all assignments (via the **AssignmentValue** element) assigned to an employee within certain organization units, during a specific time period. **Assignment** fragments appear in IN messages **Employee** with an employee file as target.

This example below illustrates the content of an **Assignment** fragment.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg:Employee xmlns:msg="http://www.oracle.com/ows/idk/Employee"
                 xmlns="http://www.oracle.com/ows/idk/Core" >

  <Assignment>
    <Scope>
      <EmployeeID IDType="HRID" fileID="1">049996</EmployeeID>
      <StartDate>2004-01-01</StartDate>
      <EndDate>2004-12-31</EndDate>
    </Scope>
    <AssignmentValue>
      <Date>2004-01-04</Date>
      <PartyID IDType="StoreID">0464</PartyID>
    </AssignmentValue>
    <AssignmentValue>
      <Date>2004-04-01</Date>
      <PartyID IDType="StoreID">0468</PartyID>
    </AssignmentValue>
  </Assignment>
</msg:Employee>
```

NB: The **Date** property of the AssignmentValue element is mandatory.

The previous example gives a portion of history of an employee's assignment history.

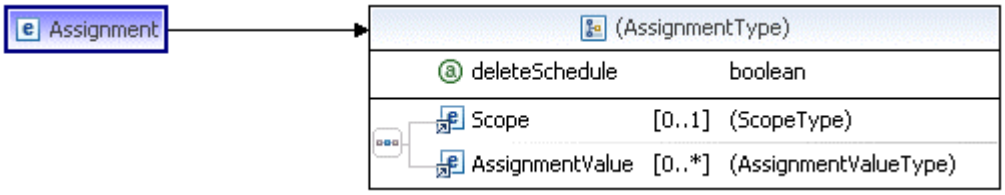
- Assignment is unknown between the start date of the scope (January 1) and the first value given (January 4). If an assignment existed in that time frame (from January 1 to January 3) it is reset to unknown.
- Two assignments are given, one from January 4 until March 31, and the second one from April 1 until December 31 (the end date in the scope).
- Assignments after December 31 and before January 1 are left unchanged.

An update or creation of an assignment occurs, only if:

1. An employee exists with an HRID 049996.
2. It has only one contract at the start date. However, if the contract management specifies the existence of a single contract at any time, a default contract can be created, if it does not exist at the start date.
3. Two organization units of class Store must exist with the codes 0464 and 0468.

However, if there is a unique unit class supporting assignments, the IDType attribute can be omitted.

Definition



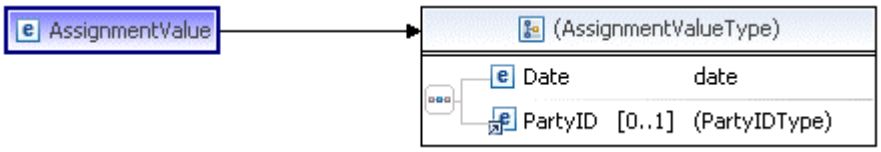

```
<xs:element name="Assignment">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="AssignmentValue"/>
      </xs:sequence>
      <xs:attribute default="false" name="deleteSchedule" type="xs:boolean"
use="optional"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
deleteSchedule	attribute	xs:boolean	0..1	
Scope	element	Scope	0..1	
AssignmentValue	element	AssignmentValue	0..*	

AssignmentValue Element

AssignmentValue Definition



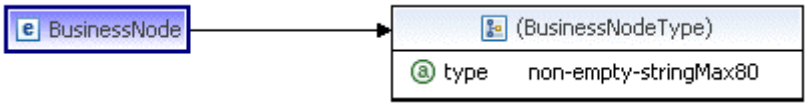

```
<xs:element name="AssignmentValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element minOccurs="0" ref="PartyID"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

AssignmentValue Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	1..1	Effective date of the value
PartyID	element	PartyID	0..1	Identification of the "team" business unit to which the employee is assigned.

BusinessNode Element

Definition



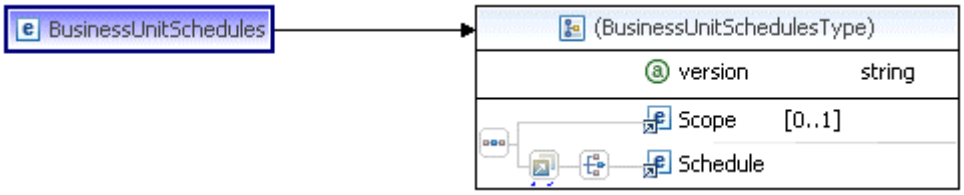

```
<xs:element name="BusinessNode">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="non-empty-stringMax80">
            <xs:attribute name="type" type="non-empty-stringMax80"
use="required"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
type	attribute	non-empty-stringMax80	1	

BusinessUnitSchedules Element

Definition



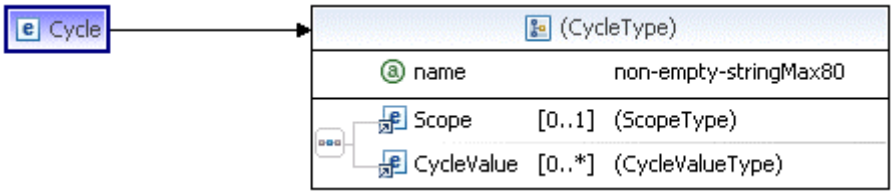

```
<xs:element name="BusinessUnitSchedules">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="1" ref="ows:Scope"/>
         <xs:group maxOccurs="unbounded" ref="Block"/>
      </xs:sequence>
      <xs:attribute default="1.0" name="version" type="xs:string"
use="optional"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
version	attribute	xs:string	0..1	
Scope	element	Scope	0..1	
Schedule	element	Schedule	0..*	

Cycle Element

Definition



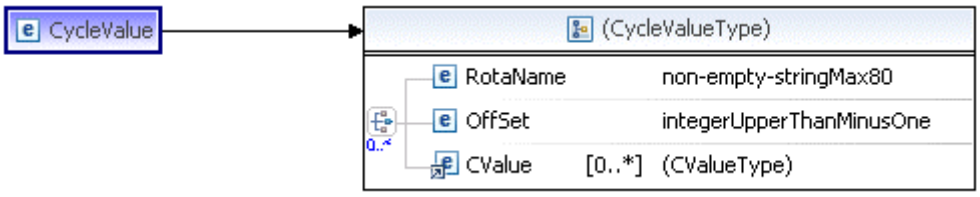

```
<xs:element name="Cycle">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="CycleValue"/>
      </xs:sequence>
      <xs:attribute name="name" type="non-empty-stringMax80" use="required"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
name	attribute	non-empty-stringMax80	1	
Scope	element	Scope	0..1	
CycleValue	element	CycleValue	0..*	

CycleValue Element

CycleValue Definition




```
<xs:element name="CycleValue">
   <xs:complexType>
      <xs:choice maxOccurs="unbounded" minOccurs="0">
         <xs:element name="RotaName" type="non-empty-stringMax80"/>
         <xs:element name="OffSet" type="integerUpperThanMinusOne"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="CValue"/>
      </xs:choice>
   </xs:complexType>
</xs:element>
```

CycleValue Content Description

Name	Category	Type	Cardinality	Description
RotaName	element	non-empty-stringMax80	0..*	The identifier of the rotation defined by the element.
OffSet	element	integerUpperThanMinusOne	0..*	The starting week in the list defined by the CValue element.
CValue	element	CValue	0..*	A code list giving the ordered list of week types for that cycle

CValue Element

CValue Definition

```
<xs:element name="CValue">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="non-empty-stringMax80">
            <xs:attribute name="index" type="xs:unsignedShort"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

CValue Content Description

Name	Category	Type	Cardinality	Description
index	attribute	xs:unsignedShort	0..1	

Chart Element

Description

A **Chart** fragment may have an organization unit or an employee as a target. A **Chart** fragment describes values of a chart for a time period. The attribute name of a **Chart** fragment indicates the type of chart and its business meaning. Names are listed in the CCD (Customer's Configuration Document) and are either predefined or specific to the customer (the majority). It also defines:

- The units (called the precision) of the chart: week, day, 15 minutes
- When the unit is a day, the chart can be defined:
    - For each day: One value is given for every day, even when several successive days have values.
    - By a range of days: One value is given for the day on which the value changes. Then, this value is assumed to be the same for all successive days until a new value is specified for a later date.
- The unit of quantity and its number type (integer or decimal)

```
<Chart name='nbOfCustomers'>
  <Scope>
    <PartyID IDType='StoreID'>0464</PartyID>
    <StartDate>2004-01-04</StartDate>
    <EndDate>2003-01-10</EndDate>
  </Scope>
  <Day>
    <Date>2003-12-08</Date>
    <Quantity index='0'>5</Quantity>
    <Quantity index='1'>7</Quantity>
  </Day>
  <Day>
    <Date>2003-12-09</Date>
    <Quantity index='0'>15</Quantity>
    <Quantity index='1'>25</Quantity>
  </Day>
```

```
...
</Chart>
```

Precision of a Day or Week

A **<Date>** element specifies the first day of the precision period. For example, for a **week** precision, all given dates are the first days of weeks. Only one **<Quantity>** element is given inside each **<Day>** without an index attribute. < a Inside Precision Minutes>A **<Date>** element specifies the day. Then, a vector of quantities is given, with one element for each time slot. For example, for a 15-minute increment chart, a vector of 96 values is defined, one for each quarter of an hour within the day. The **index=** attribute starts at **0**. You can observe that for days having more than 24 hours (midnight crossing), the index value is over 95. The index value of 0 is the day start time (not always midnight).
Note: Dates must be in the ascending order in a chart. A chart in Oracle Workforce Scheduling has values from the first to the last Oracle Workforce Scheduling day: all days between the start and end dates in the scope are replaced. The effect of a **Chart** fragment is to completely replace a slot of an existing chart or to create a chart having values in that slot only. Except for a day precision chart given by range, values not given are left as unknown. The meaning of unknown is not zero, but unknown.

Cyclic Charts

The CCD can specify a daily cyclic profile (15 minutes) named for example, as TuesdayNormalWeek (Tuesday in a week type, Normal). For importing the corresponding data, a fragment has to only provide one single day with an array of 96 values; this day will be cyclically repeated for all Tuesdays inside the scope. The CCD can specify a weekly cyclic profile (daily values) named for example as PeakWeek (sales profile for a peak week). For importing the corresponding data, a fragment has to only provide one single day with an array of 7 values; those days will be cyclically repeated for all weeks inside the scope. Mapping and Processing: Organization Unit Level At the level of organization units, a name of a **Chart** is given in the CCD. For example, for a store, the chart **nbOfCustomers**, is specified with:

- The time precision of the chart (day) and the units on the x-axis. For a day precision, the option of providing values by range can be specified.
- The internal representation of the quantity (integer or decimal)

The processing involves storing the chart for the store. The **Chart** fragments are presented in:

- IN messages of **Organization** type for incoming history of customer flows, transactions, truck arrivals, etc. and **Employee** messages for actual and accrual charts
- OUT messages of KPI type for stating KPI computed by Oracle Workforce Scheduling (or any importable chart).

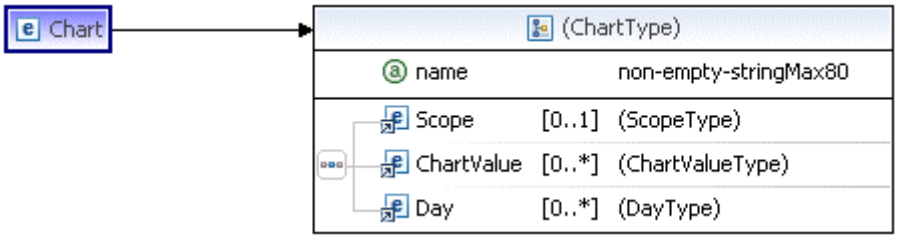
Mapping and Processing: Employee level

At the employee level, chart names are restrictively defined at application setup:

- **hoursPaid**: the weekly amount paid to each employee
- **hoursDone**: the number of hours worked per week and per employee

These **Chart** fragments are present in the incoming messages of the type Employee for incoming history of hours paid, hours completed, and YTD accrued hours.

Definition




```
<xs:element name="Chart">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="ChartValue"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Day"/>
      </xs:sequence>
      <xs:attribute name="name" type="non-empty-stringMax80" use="required"/>
   </xs:complexType>
</xs:element>
```

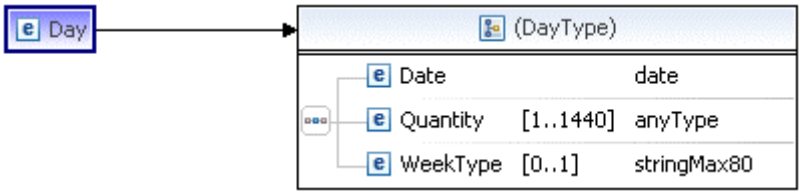
Content Description

Name	Category	Type	Cardinality	Description
name	attribute	non-empty-	1	

		stringMax80		
Scope	element	Scope	0..1	
ChartValue	element	ChartValue	0..*	
Day	element	Day	0..*	

Day Element

Day Definition



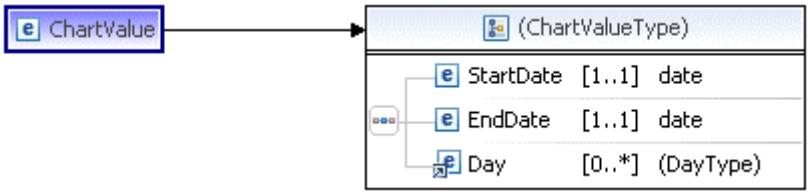

```
<xs:element name="Day">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="Quantity" maxOccurs="1440"/>
         <xs:element name="WeekType" type="stringMax80" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Day Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	1..1	
Quantity	element	xs:anyType	1..1440	
WeekType	element	stringMax80	0..1	

ChartValue Element

Definition




```
<xs:element name="ChartValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="StartDate" type="xs:date" minOccurs="1"/>
         <xs:element name="EndDate" type="xs:date" minOccurs="1"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Day"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
StartDate	element	xs:date	1..1	
EndDate	element	xs:date	1..1	
Day	element	Day	0..*	

Contact Element

Description

The **Contact** fragment is attached to an employee and gives a history (via the **ContactValue** element) of all contact information.

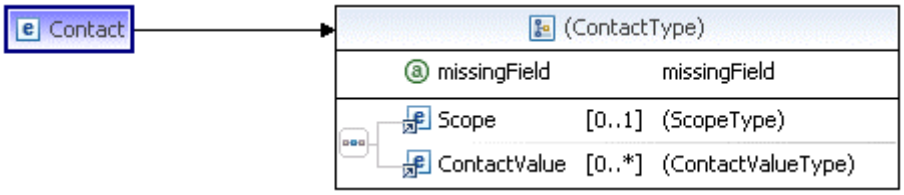
```
<?xml version="1.0" encoding="UTF-8"?>
<msg:Employee xmlns:msg="http://www.oracle.com/ows/idk/Employee"
                 xmlns="http://www.oracle.com/ows/idk/Core"
>

<Contact>
  <Scope>
    <EmployeeID IDType="HRID">049996</EmployeeID>
    <StartDate>2004-01-01</StartDate>
    <EndDate>2004-12-31</EndDate>
  </Scope>
  <ContactValue>
    <Date>2004-01-04</Date>
    <HomePhone>0607080910</HomePhone>
    ...
  </ContactValue>
  <ContactValue>
    <Date>2004-01-04</Date>
    <HomePhone>0607080912</HomePhone>
    ...
  </ContactValue>
</Contact>
```

NB: The **Date** field is mandatory.

The **Contact** element may have an optional attribute value **missingField**="null. This attribute value specifies that the predefined fields (such as **CellPhone** have their value set to null if they are not specified in the imported message.

Definition



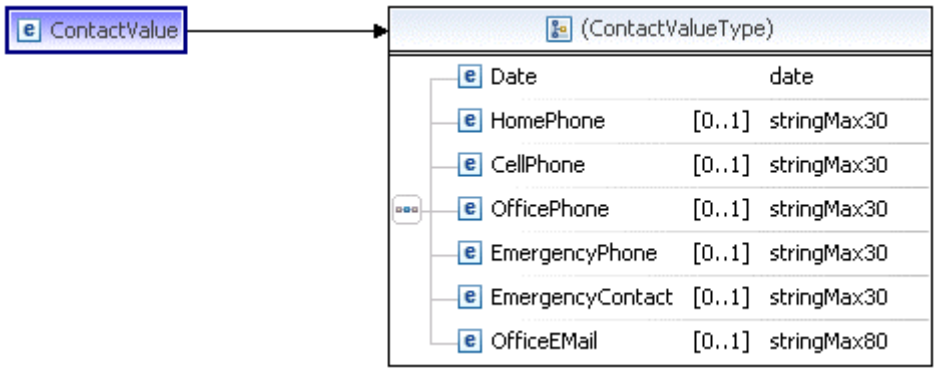

```
<xs:element name="Contact">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="ContactValue"/>
      </xs:sequence>
      <xs:attribute name="missingField" type="missingField" use="optional"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
missingField	attribute	missingField	0..1	
Scope	element	Scope	0..1	
ContactValue	element	ContactValue	0..*	

ContactValue Element

ContactValue Definition




```
<xs:element name="ContactValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="HomePhone" type="stringMax30" minOccurs="0"/>
         <xs:element name="CellPhone" type="stringMax30" minOccurs="0"/>
         <xs:element name="OfficePhone" type="stringMax30" minOccurs="0"/>
         <xs:element name="EmergencyPhone" type="stringMax30" minOccurs="0"/>
         <xs:element name="EmergencyContact" type="stringMax30" minOccurs="0"/>
         <xs:element name="OfficeEMail" type="stringMax80" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

ContactValue Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	1..1	The effective date of the value.
HomePhone	element	stringMax30	0..1	The employee's home phone number.
CellPhone	element	stringMax30	0..1	The employee's cell phone number.

OfficePhone	element	stringMax30	0..1	The employee's office phone number.
EmergencyPhone	element	stringMax30	0..1	The phone number to use in case of emergency.
EmergencyContact	element	stringMax30	0..1	The name of the person to contact in case of emergency.
OfficeEMail	element	stringMax80	0..1	The employee's official e-mail address.

Contract Element

Description

The **Contract** fragment is attached to an employee's file (contract) and provides a history of all fields attached to one contract (file) of one employee. It has its own list of predefined fields. Each field has an element and contains a field name, for example <MinWeeklyDuration> <MinWorkingDays> etc. However, it can have extra informative fields declared at the application setup and recorded in the CCD. These informative fields are given by the element <Field name="myExtraField"> its value</Filed> similar to a <Variable> element.

Groups of Fields

Generally for a dated history, fields that are not specified for an effective date are reset to null. So for any field change, all fields have to be given again, even if there has been no change since the previous effective date. This is a way for resetting some fields to null at some effective date. <Contract> has an improved and a richer behavior:

- Fields are grouped according to their business meaning, for example HRContract WorkRules WorkPatterns UserFields
- For an effective date of change, if a group of fields does not have any change since the previous effective date, you need not give the same values for the group. rather to suppose that they are reset to null, it is assumed they are not changed;
- If at least one field in a group has a change at an effective date, all fields of the same group have to be given. Otherwise, missing fields are reset to null based on the standard behavior.
- If for an effective date, a group is given without any field, it means that all fields of group are reset to null at that date.

This behavior allows generating messages dedicated to a single group of fields (those known together in the source system). However, it is possible to proceed as if groups did not exist and to provide for each effective date all values for all fields for all groups.

Groups are bounded by sub tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg:Employee xmlns:msg="http://www.oracle.com/ows/idk/Employee"
                 xmlns="http://www.oracle.com/ows/idk/Core">
```

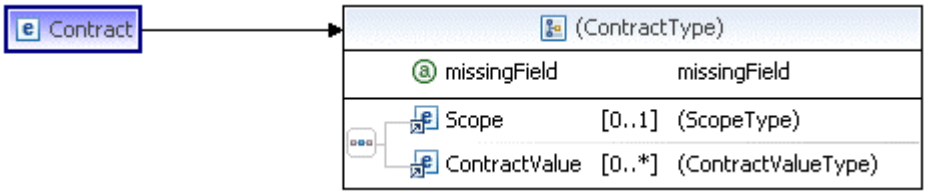
```
<Contract>
  <Scope>
    <EmployeeID IDType="HRID" fileID="1">049996</EmployeeID>
    <StartDate>2004-01-01</StartDate>
    <EndDate>2004-12-31</EndDate>
  </Scope>
  <ContractValue>
    <Date>2004-01-04</Date>
    <WorkRules>
      <MinWeeklyDuration>20.00</MinWeeklyDuration>
      <MinWOrkingDays>4</MinWorkingDays>
            ...
      <Field name="myInfo">my value 1</Field>
    </WorkRules>
  </ContractValue>
  <ContractValue>
    <Date>2004-01-04</Date>
    <WorkRules>
      <MinWeeklyDuration>24.00</MinWeeklyDuration>
      <MinWOrkingDays>4</MinWorkingDays>
            ...
      <Field name="myInfo">my value 2</Field>
    </WorkRules>
            ...
   </ContractValue>
            ...
  </Contract>
</msg:Employee>
```

Note: Refer to the Oracle Workforce Scheduling Interface Functional Guide to view all the fields allowed in a <Contract> fragment and its group. Creation During the assignment import, all undefined contract values are set to the default values. The ContractType value has to exist in the CCD, otherwise the import will fail. Update If you update one field of a group, it automatically updates all the others of the same group. When reassigning a contract, all undefined contract values are set to the default contract values.

The tag Contract may have an optional attribute missingField="null" or missingField="default". When a field is missing (for example ContractType):

- When this attribute is absent, the existing value is left unchanged at that time;
- When the attribute is missingField="null", the value is reset to null. When the attribute is missingField="default", the value is reset to its default value

Definition



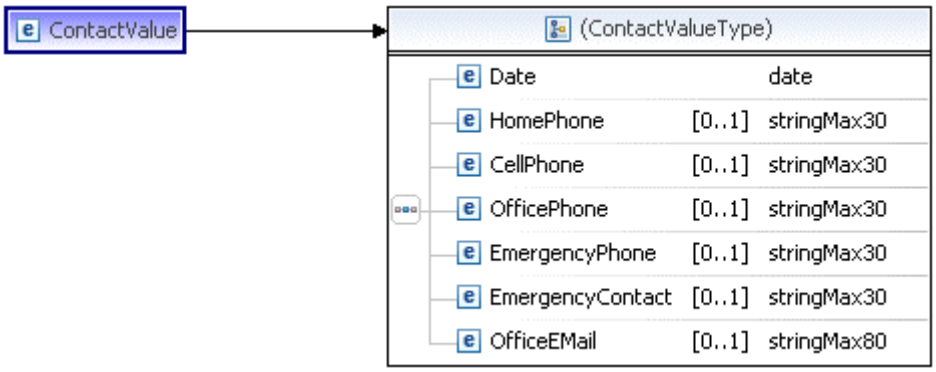

```
<xs:element name="Contract">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="ContractValue"/>
      </xs:sequence>
      <xs:attribute name="missingField" type="missingField" use="optional"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
missingField	attribute	missingField	0..1	
Scope	element	Scope	0..1	
ContractValue	element	ContractValue	0..*	

ContractValue Element

ContractValue Definition




```
<xs:element name="ContractValue">
   <xs:complexType>
      <xs:choice maxOccurs="unbounded" minOccurs="0">
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="PayType" type="stringMax255"/>
         <xs:group minOccurs="0" ref="ContractGroup"/>
      </xs:choice>
   </xs:complexType>
</xs:element>
```

ContractValue Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	0..*	Effective date of the value.
PayType	element	stringMax255	0..*	
HRContract	element	HRContract	0..*	
WorkRules	element	WorkRules	0..*	
WorkPatterns	element	WorkPatterns	0..*	

Options	element	Options	0..*	
UserFields	element	UserFields	0..*	

CrossStore Element

Description

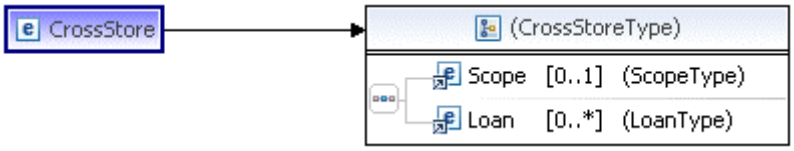
In configuring the business hierarchy, you create store and department libraries. You can define subnodes under store units (store group) and subnodes under department units (mandatory or optional) which are referenced under store units.

For greater flexibility in meeting staffing requirements, you can lend employees to destination stores and departments on an hourly or daily basis, and the manager at the destination location can then schedule the person's activities. The application accounts for absences so that employees are not loaned to other stores at those times.

The OWS application displays loan information in the daily and weekly schedules, and employee maintenance team schedules, and maintains a history of the loans. To-do list notifications inform managers when an employee is loaned to them.

To have greater control over the conditions that permit cross-store scheduling, you can specify optimization parameters during configuration (ignore departments, strict department scheduling, and department preference).

Definition




```
<xs:element name="CrossStore">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Loan"/>
      </xs:sequence>
   </xs:complexType>
```

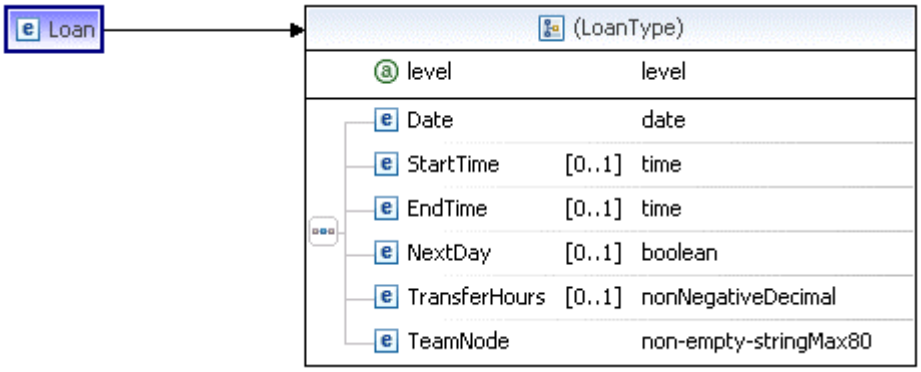
```
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	0..1	
Loan	element	Loan	0..*	

Loan Element

Loan Definition




```
<xs:element name="Loan">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="StartTime" type="xs:time" minOccurs="0"/>
         <xs:element name="EndTime" type="xs:time" minOccurs="0"/>
         <xs:element name="NextDay" type="xs:boolean" minOccurs="0"/>
         <xs:element name="TransferHours" type="nonNegativeDecimal"
minOccurs="0"/>
         <xs:element name="TeamNode" type="non-empty-stringMax80"/>
      </xs:sequence>
      <xs:attribute name="level" type="level" use="required"/>
   </xs:complexType>
</xs:element>
```

Loan Content Description

Name	Category	Type	Cardinality	Description
level	attribute	level	1	Indicates whether the loan will be daily or hourly (detail value)
Date	element	xs:date	1..1	Date of loan of the employee
StartTime	element	xs:time	0..1	If the loan type is hourly, then the Start time indicates the start time of the loan
EndTime	element	xs:time	0..1	If the loan type is hourly, then the End time indicates the end time of the loan
NextDay	element	xs:boolean	0..1	True or false
TransferHours	element	nonNegativeDecimal	0..1	The total hours the employee is loaned
TeamNode	element	non-empty-stringMax80	1..1	The team node to which the employee is loaned

Event Element

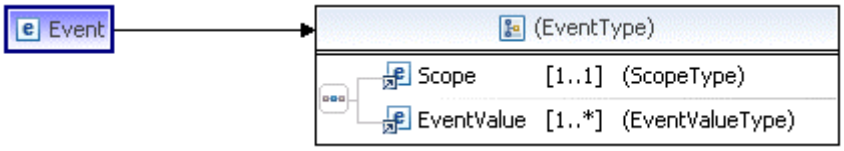
Description

The **Event** fragment allows you to define event at a particular node and choose which drivers will be impacted, and the value of the impact, for each event. It can either be a single daily value for an impact, or one can make the impact cyclic over several days. An example is given below.

```
<Event>
  <Scope>
    <PartyID IDType="RegionID">California</PartyID>
    <StartDate>1953-01-01</StartDate>
    <EndDate>9999-12-31</EndDate>
  </Scope>
  <EventValue>
   <EventName>Thanksgiving</EventName>
   <EventID>Thanksgiving</EventID>
   <Driver>
     <DriverName>CleanFridg</DriverName>
     <DriverIndex>0</DriverIndex>
     <DriverImpact>15</DriverImpact>
   </Driver>
   <Driver>
     <DriverName>DrFrozPal</DriverName>
     <DriverIndex>0</DriverIndex>
     <DriverImpact>20</DriverImpact>
   </Driver>
  </EventValue>
</Event>
```

In the above example, the Thanksgiving event has been defined at the Region node, for example, at California from 01/01/1953 to 12/31/9999. Driver CleanFridg is impacted by this event at 15% on first day and the Driver DrFrozPal is impacted by 20% on the first day.

Definition



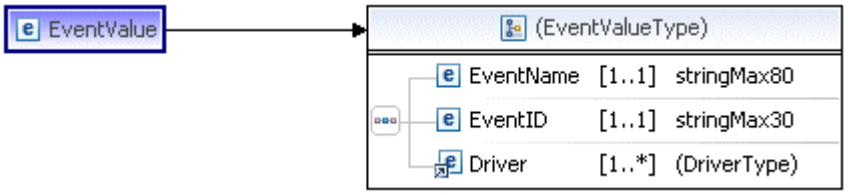

```
<xs:element name="Event">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="1" ref="Scope"/>
         <xs:element minOccurs="1" maxOccurs="unbounded" ref="EventValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	1..1	
EventValue	element	EventValue	1..*	

EventValue Element

EventValue Definition




```
<xs:element name="EventValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="EventName" type="stringMax80" minOccurs="1"/>
```

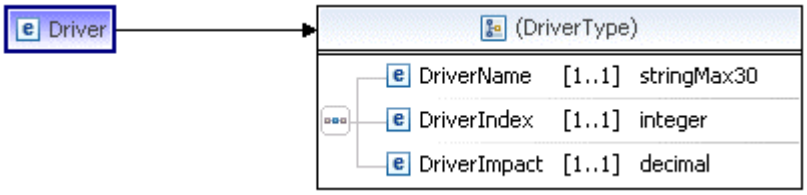
```
        <xs:element name="EventID" type="stringMax30" minOccurs="1"/>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="Driver"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

EventValue Content Description

Name	Category	Type	Cardinality	Description
EventName	element	stringMax80	1..1	
EventID	element	stringMax30	1..1	
Driver	element	Driver	1..*	

Driver Element

Driver Definition



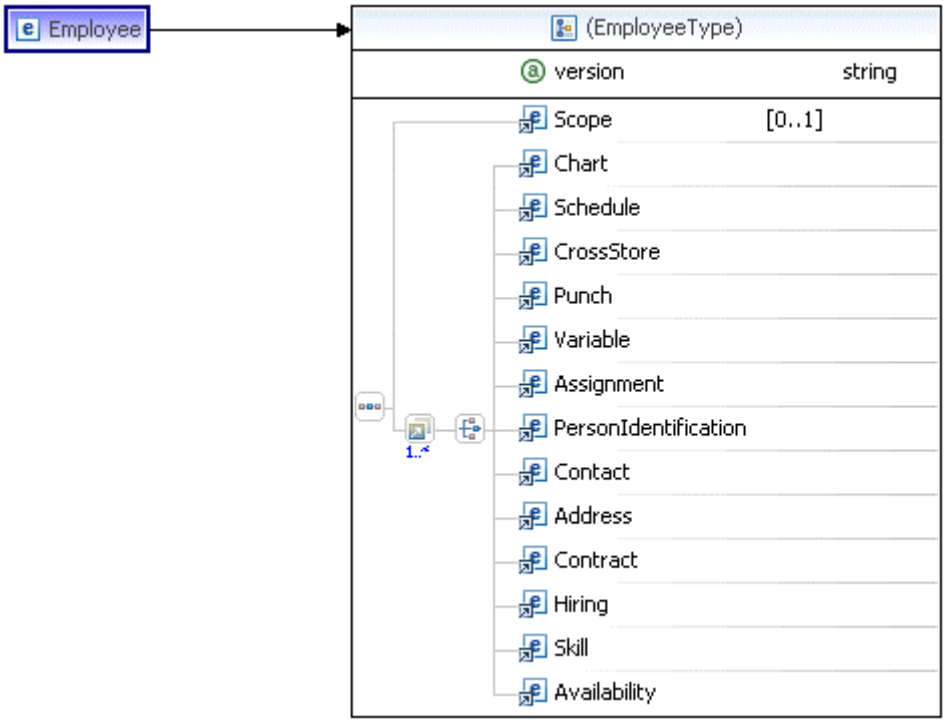

```
<xs:element name="Driver">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="DriverName" type="stringMax30" minOccurs="1"/>
         <xs:element name="DriverIndex" type="xs:integer" minOccurs="1"/>
         <xs:element name="DriverImpact" type="xs:decimal" minOccurs="1"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Driver Content Description

Name	Category	Type	Cardinality	Description
DriverName	element	stringMax30	1..1	
DriverIndex	element	xs:integer	1..1	
DriverImpact	element	xs:decimal	1..1	

Employee Element

Definition



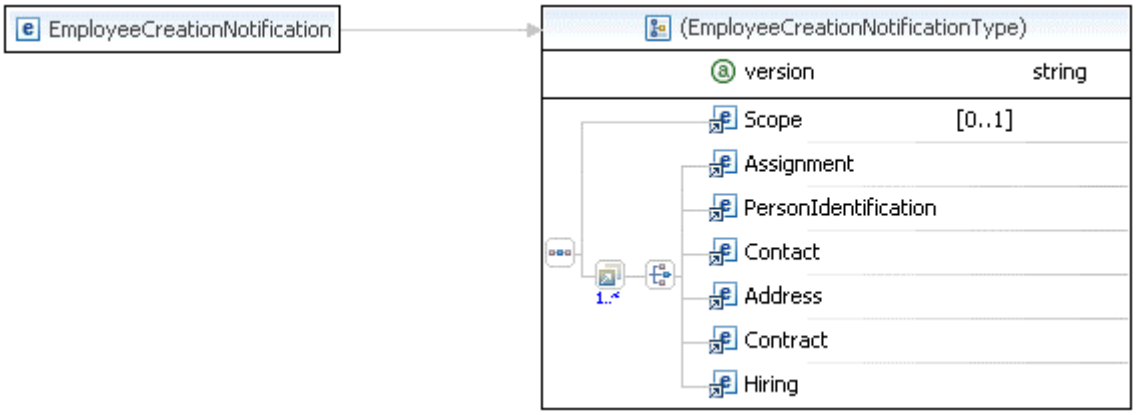

```
<xs:element name="Employee">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="ows:Scope"/>
         <xs:group maxOccurs="unbounded" ref="Block"/>
      </xs:sequence>
      <xs:attribute default="1.0" name="version" type="xs:string"
use="optional"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
version	attribute	xs:string	0..1	
Scope	element	Scope	0..1	
Chart	element	Chart	0..*	
Schedule	element	Schedule	0..*	
CrossStore	element	CrossStore	0..*	
Punch	element	Punch	0..*	
Variable	element	Variable	0..*	
Assignment	element	Assignment	0..*	
PersonIdentification	element	PersonIdentification	0..*	
Contact	element	Contact	0..*	
Address	element	Address	0..*	
Contract	element	Contract	0..*	
Hiring	element	Hiring	0..*	
Skill	element	Skill	0..*	
Availability	element	Availability	0..*	

EmployeeCreationNotification Element

Definition




```
<xs:element name="EmployeeCreationNotification">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="1" ref="ows:Scope"/>
         <xs:group maxOccurs="unbounded" ref="Block"/>
      </xs:sequence>
      <xs:attribute default="1.0" name="version" type="xs:string"
use="optional"/>
   </xs:complexType>
</xs:element>
```

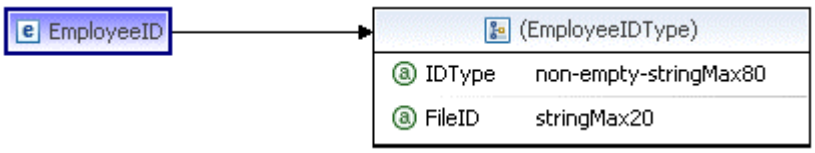
Content Description

Name	Category	Type	Cardinality	Description
version	attribute	xs:string	0..1	
Scope	element	Scope	0..1	

Assignment	element	Assignment	0..*	
PersonIdentification	element	PersonIdentification	0..*	
Contact	element	Contact	0..*	
Address	element	Address	0..*	
Contract	element	Contract	0..*	
Hiring	element	Hiring	0..*	

EmployeeID Element

Definition




```
<xs:element name="EmployeeID">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="stringMax20">
            <xs:attribute name="IDType" type="non-empty-stringMax80"/>
            <xs:attribute name="FileID" type="stringMax20"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
IDType	attribute	non-empty-stringMax80	0..1	
FileID	attribute	stringMax20	0..1	

EventAssignment Element

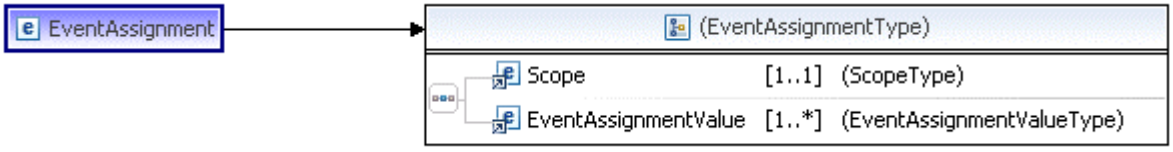
Description

The **EventAssignment** fragment allows you to assign the event that has been defined. One can thus manage dated events for the stores through the organization hierarchy based on the event definitions. The user can assign dated events for each configured hierarchy level.
An example is given below.

```
<EventAssignment>
  <Scope>
    <PartyID IDType="StoreID">San Francisco</PartyID>
    <StartDate>1953-01-01</StartDate>
    <EndDate>9999-12-31</EndDate>
  </Scope>
  <EventAssignmentValue>
    <EventID>Thanksgiving</EventID>
    <StartDate>2007-03-20</StartDate>
    <EndDate>2007-03-28</EndDate>
  </EventAssignmentValue>
</EventAssignment>
```

In the above example, the Thanksgiving event is assigned to the store San Francisco from 03/20/2007 to 03/28/2007.

Definition




```
<xs:element name="EventAssignment">
   <xs:complexType>
      <xs:sequence>
```

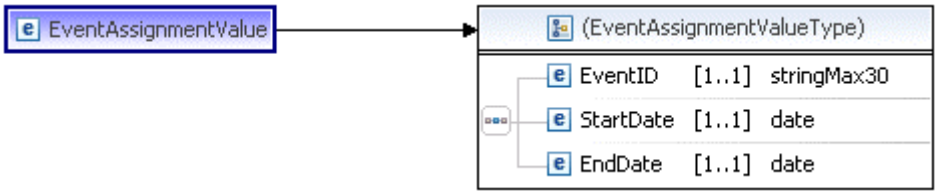
```
        <xs:element minOccurs="1" ref="Scope"/>
        <xs:element minOccurs="1" maxOccurs="unbounded"
ref="EventAssignmentValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	1..1	
EventAssignmentValue	element	EventAssignmentValue	1..*	

EventAssignmentValue Element

EventAssignmentValue Definition




```
<xs:element name="EventAssignmentValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="EventID" type="stringMax30" minOccurs="1"/>
         <xs:element name="StartDate" type="xs:date" minOccurs="1"/>
         <xs:element name="EndDate" type="xs:date" minOccurs="1"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

EventAssignmentValue Content Description

Name	Category	Type	Cardinality	Description

EventID	element	stringMax30	1..1	
StartDate	element	xs:date	1..1	
EndDate	element	xs:date	1..1	

Event Element

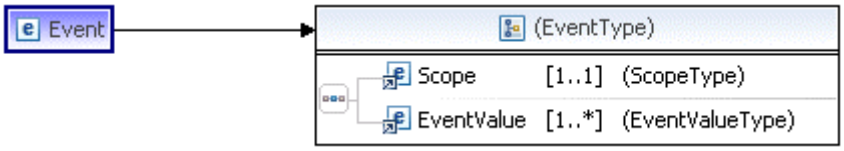
Description

The **Event** fragment allows you to define event at a particular node and choose which drivers will be impacted, and the value of the impact, for each event. It can either be a single daily value for an impact, or one can make the impact cyclic over several days. An example is given below.

```
<Event>
  <Scope>
    <PartyID IDType="RegionID">California</PartyID>
    <StartDate>1953-01-01</StartDate>
    <EndDate>9999-12-31</EndDate>
  </Scope>
  <EventValue>
   <EventName>Thanksgiving</EventName>
   <EventID>Thanksgiving</EventID>
   <Driver>
     <DriverName>CleanFridg</DriverName>
     <DriverIndex>0</DriverIndex>
     <DriverImpact>15</DriverImpact>
   </Driver>
   <Driver>
     <DriverName>DrFrozPal</DriverName>
     <DriverIndex>0</DriverIndex>
     <DriverImpact>20</DriverImpact>
   </Driver>
  </EventValue>
</Event>
```

In the above example, the Thanksgiving event has been defined at the Region node, for example, at California from 01/01/1953 to 12/31/9999. Driver CleanFridg is impacted by this event at 15% on first day and the Driver DrFrozPal is impacted by 20% on the first day.

Definition



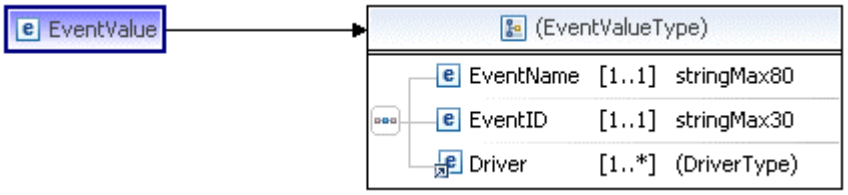

```
<xs:element name="Event">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="1" ref="Scope"/>
         <xs:element minOccurs="1" maxOccurs="unbounded" ref="EventValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	1..1	
EventValue	element	EventValue	1..*	

EventValue Element

EventValue Definition




```
<xs:element name="EventValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="EventName" type="stringMax80" minOccurs="1"/>
```

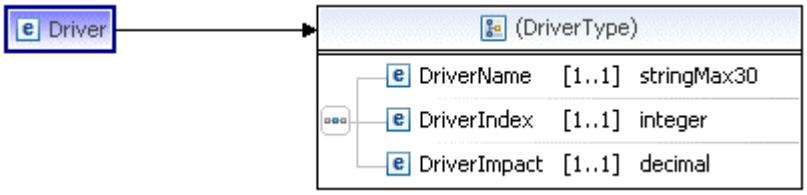
```
        <xs:element name="EventID" type="stringMax30" minOccurs="1"/>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="Driver"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

EventValue Content Description

Name	Category	Type	Cardinality	Description
EventName	element	stringMax80	1..1	
EventID	element	stringMax30	1..1	
Driver	element	Driver	1..*	

Driver Element

Driver Definition




```
<xs:element name="Driver">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="DriverName" type="stringMax30" minOccurs="1"/>
         <xs:element name="DriverIndex" type="xs:integer" minOccurs="1"/>
         <xs:element name="DriverImpact" type="xs:decimal" minOccurs="1"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Driver Content Description

Name	Category	Type	Cardinality	Description
DriverName	element	stringMax30	1..1	
DriverIndex	element	xs:integer	1..1	
DriverImpact	element	xs:decimal	1..1	

ExportKPI Element

Definition

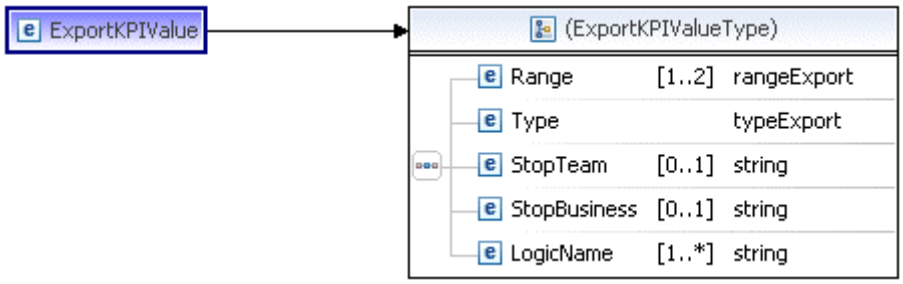

```
<xs:element name="ExportKPI">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element ref="ExportKPIValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	0..1	
ExportKPIValue	element	ExportKPIValue	1..1	

ExportKPIValue Element

ExportKPIValue Definition




```
<xs:element name="ExportKPIValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Range" type="rangeExport" maxOccurs="2"/>
         <xs:element name="Type" type="typeExport"/>
         <xs:element name="StopTeam" type="xs:string" minOccurs="0"/>
         <xs:element name="StopBusiness" type="xs:string" minOccurs="0"/>
         <xs:element name="LogicName" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

ExportKPIValue Content Description

Name	Category	Type	Cardinality	Description
Range	element	rangeExport	1..2	
Type	element	typeExport	1..1	
StopTeam	element	xs:string	0..1	
StopBusiness	element	xs:string	0..1	
LogicName	element	xs:string	1..*	

Field Element

Definition

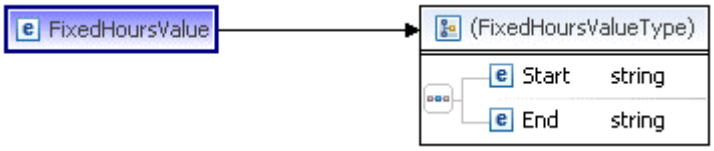

```
<xs:element name="Field">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="stringMax255">
            <xs:attribute name="name" type="non-empty-stringMax80"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
name	attribute	non-empty-stringMax80	0..1	

FixedHoursValue Element

Definition



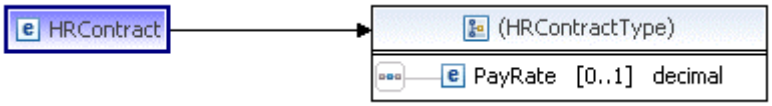

```
<xs:element name="FixedHoursValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Start" type="xs:string"/>
         <xs:element name="End" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Start	element	xs:string	1..1	The start time of the fixed hours.
End	element	xs:string	1..1	The end time of the fixed hours.

HRContract Element

Definition




```
<xs:element name="HRContract">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="PayRate" type="xs:decimal" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
PayRate	element	xs:decimal	0..1	Hourly dollars for the employee for this file.

HRSchedules Element

Definition


```
<xs:element name="HRSchedules">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="1" ref="ows:Scope"/>
         <xs:group maxOccurs="unbounded" ref="Block"/>
      </xs:sequence>
      <xs:attribute default="1.0" name="version" type="xs:string"
use="optional"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
version	attribute	xs:string	0..1	
Scope	element	Scope	0..1	
Schedule	element	Schedule	0..*	

Hiring Element

Description

The **Hiring** fragment is attached to an employee and gives a history of all fields attached to hiring. Thus, the target of **Hiring** is an employee. The **Hiring** fragment has a list of predefined fields, but currently it has only the HiringDate field. However, it can have extra informative fields declared at the application setup and recorded in the CCD. These informative fields are given by the element <Field name="myExtraField"> its value</Filed> similar to a **Variable** element.

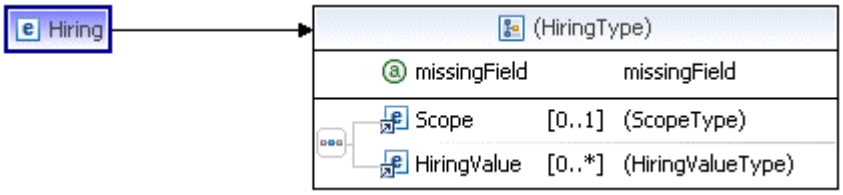
```
<?xml version="1.0" encoding="UTF-8"?>
<msg:Employee xmlns:msg="http://www.oracle.com/ows/idk/Employee"
                 xmlns="http://www.oracle.com/ows/idk/Core" >

  <Hiring>
    <Scope>
      <EmployeeID IDType="HRID" fileID="1">049996</EmployeeID>
      <StartDate>2004-01-01</StartDate>
      <EndDate>2004-12-31</EndDate>
    </Scope>
    <HiringValue>
      <Date>2004-01-04</Date>
      <HiringDate>2004-01-04</HiringDate>
      <Field name="myInfo">my value 1</Field>
              ...
    </HiringValue>
    <HiringValue>
      <Date>2004-01-04</Date>
      <HiringDate>2004-01-04</HiringDate>
      <Field name="myInfo">my value 1</Field>
              ...
    </HiringValue>
              ...
  </Hiring >
```

```
</msg:Employee>
```

The only mandatory field is **Date**.

Definition




```
<xs:element name="Hiring">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="HiringValue"/>
      </xs:sequence>
      <xs:attribute name="missingField" type="missingField" use="optional"/>
   </xs:complexType>
</xs:element>
```

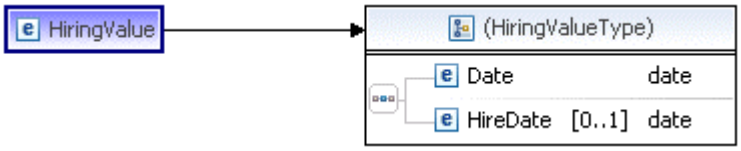
Content Description

Name	Category	Type	Cardinality	Description
missingField	attribute	missingField	0..1	

Scope	element	Scope	0..1	
HiringValue	element	HiringValue	0..*	

HiringValue Element

HiringValue Definition




```
<xs:element name="HiringValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="HireDate" type="xs:date" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

HiringValue Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	1..1	Effective date of the value.
HireDate	element	xs:date	0..1	The date indicating the first working day for the employee.

Job Element

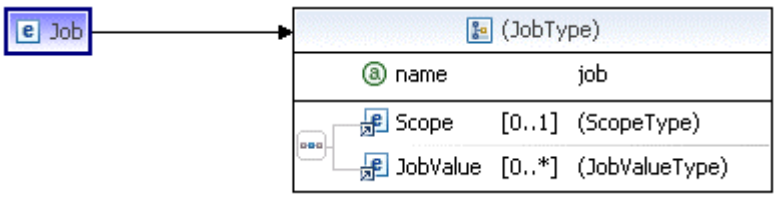
Description

The OWS application includes job management and scheduling modules that the functional administrator can use.

- Using the Job Management module, administrators can search, view, and edit asynchronous jobs that have been launched for the store or its departments. The administrator can view job details and edit these details, change the scheduled date of a pending job, cancel a pending or running job. Detailed logs supply further information about jobs. Predefined queries as well as advanced search criteria filter and streamline searches for jobs.
- Using the Job Scheduler module, administrators can schedule the ExportSchedule and ExportKPIs procedures and the launch batch processes, including: Forecast, Demand, Check, FireButton, and EarnedHours procedures.

For job scheduling, the interface message contains the job lists and parameters associated with each job.

Definition



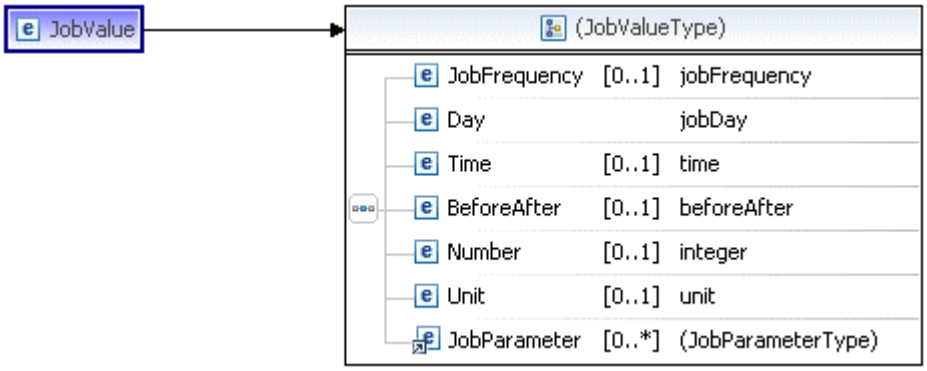

```
<xs:element name="Job">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="JobValue"/>
      </xs:sequence>
      <xs:attribute name="name" type="job" use="required"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
name	attribute	job	1	
Scope	element	Scope	0..1	
JobValue	element	JobValue	0..*	

JobValue Element

JobValue Definition



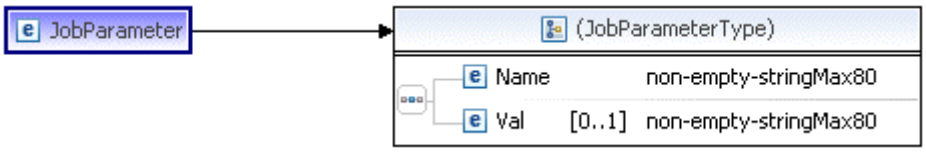

```
<xs:element name="JobValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="JobFrequency" type="jobFrequency" minOccurs="0"/>
         <xs:element name="Day" type="jobDay"/>
         <xs:element name="Time" type="xs:time" minOccurs="0"/>
         <xs:element name="BeforeAfter" type="beforeAfter" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="Number" type="xs:integer" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="Unit" type="unit" minOccurs="0" maxOccurs="1"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="JobParameter"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

JobValue Content Description

Name	Category	Type	Cardinality	Description
JobFrequency	element	jobFrequency	0..1	The frequency
Day	element	jobDay	1..1	The day of export: -1 for Now option in all frequencies, 1 to 31 for Monthly frequency, 0 to 6 for weekly frequency (0 is Sunday), 0 for daily frequency
Time	element	xs:time	0..1	Time of the export
BeforeAfter	element	beforeAfter	0..1	Offset before or after execution date
Number	element	xs:integer	0..1	Offset
Unit	element	unit	0..1	Unit of offset
JobParameter	element	JobParameter	0..*	

JobParameter Element

JobParameter Definition




```
<xs:element name="JobParameter">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Name" type="non-empty-stringMax80"/>
         <xs:element name="Val" type="non-empty-stringMax80" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

JobParameter Content Description

Nam	Categor	Type	Cardinalit	Description

e	y		y	
Name	element	non-empty-stringMax80	1..1	Name of the parameter associated with one of the three jobs to launch
Val	element	non-empty-stringMax80	0..1	The Value of this parameter

KPI Element

Definition

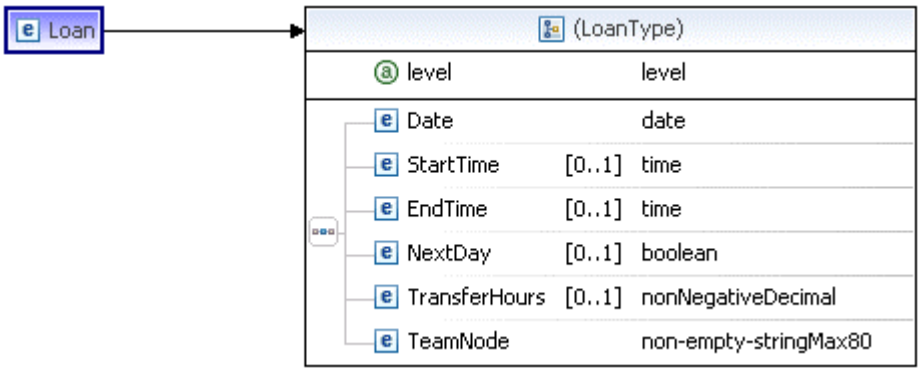

```
<xs:element name="KPI">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="1" ref="ows:Scope"/>
         <xs:group maxOccurs="unbounded" ref="Block"/>
      </xs:sequence>
      <xs:attribute default="1.0" name="version" type="xs:string"
use="optional"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
version	attribute	xs:string	0..1	
Scope	element	Scope	0..1	
Chart	element	Chart	0..*	

Loan Element

Loan Definition




```
<xs:element name="Loan">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="StartTime" type="xs:time" minOccurs="0"/>
         <xs:element name="EndTime" type="xs:time" minOccurs="0"/>
         <xs:element name="NextDay" type="xs:boolean" minOccurs="0"/>
         <xs:element name="TransferHours" type="nonNegativeDecimal"
minOccurs="0"/>
         <xs:element name="TeamNode" type="non-empty-stringMax80"/>
      </xs:sequence>
      <xs:attribute name="level" type="level" use="required"/>
   </xs:complexType>
</xs:element>
```

Loan Content Description

Name	Category	Type	Cardinality	Description
level	attribute	level	1	Indicates whether the loan will be daily or hourly (detail value)
Date	element	xs:date	1..1	Date of loan of the employee
StartTime	element	xs:time	0..1	If the loan type is hourly, then the Start time indicates the start

				time of the loan
EndTime	element	xs:time	0..1	If the loan type is hourly, then the End time indicates the end time of the loan
NextDay	element	xs:boolean	0..1	True or false
TransferHours	element	nonNegativeDecimal	0..1	The total hours the employee is loaned
TeamNode	element	non-empty-stringMax80	1..1	The team node to which the employee is loaned

Login Element

Description

The **Login** fragment specifies the list of fields identifying a login. A login is associated with a group and defines a set of access information. This information contains the authentication information (i.e. password), the role assigned to the user, the business context provided and some default settings (e.g. default language, time zone) activated when this login is used to enter the application.

```
<msg:Login xmlns:msg="http://www.oracle.com/ows/idk/Login"
              xmlns="http://www.oracle.com/ows/idk/Core">
  <Login>
    <Scope>
      <LoginID>MyLogin</LoginID>
      <StartDate>2006-06-01</StartDate>
      <EndDate>9999-12-31</EndDate>
    </Scope>
    <LoginValue>
      <LoginGroup>MyLoginGroup</LoginGroup>
        <Password>MyPassword</Password>
        <Accesses>
          <Access>
            <StartDate>2006-06-01</StartDate>
            <EndDate>9999-12-31</EndDate>
            <Language>en_us</Language>
            <TimeZone>0</TimeZone>
          </Access>
        </Accesses>
        <Roles>
          <Role>
            <StartDate>2006-06-01</StartDate>
            <EndDate>9999-12-31</EndDate>
            <RoleName>STORE MANAGER</RoleName>
```

```
            <BusinessNode type="StoreID">Store</BusinessNode>
            <TeamNode type="StoreID">Store</TeamNode>
          </Role>
        </Roles>
   </LoginValue>
  </Login>
</msg:Login>
```

Processing

Creation:

You can create a new login, when the <LoginID> value does not exist. In this case, all the fields are mandatory:

- A login group
- A password
- At least one access with start date, end date, language and time zone
- At least one role with start date, end date, role, business and team node

Update:

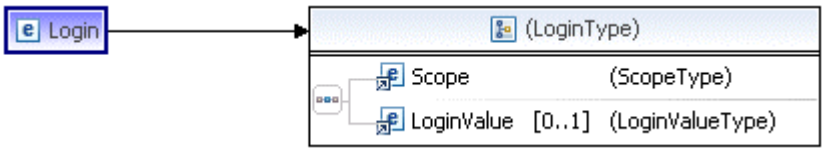
You can update a login, when the <LoginID> value is the name of an existing login. The updates are:

- Change of the login group. If a no existing login group name is defined, a new login group is created.
- Change of the password
- Add a new access < new a>

Deletion:

You can delete a login, when there id not <LoginValue> in the message.

Definition




```
<xs:element name="Login">
   <xs:complexType>
```

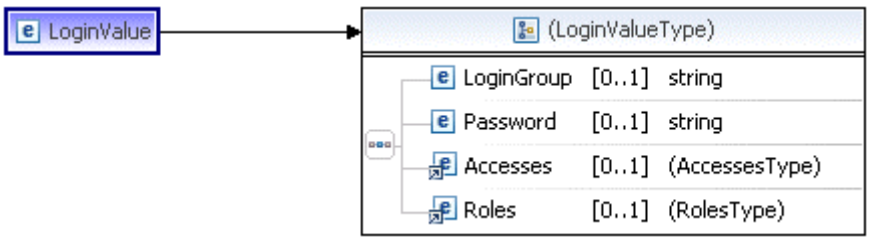
```
      <xs:sequence>
         <xs:element ref="Scope"/>
         <xs:element minOccurs="0" ref="LoginValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	1..1	
LoginValue	element	LoginValue	0..1	

LoginValue Element

LoginValue Definition



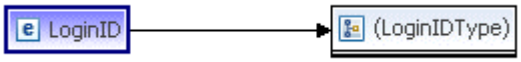

```
<xs:element name="LoginValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="LoginGroup" type="xs:string" minOccurs="0"/>
         <xs:element name="Password" type="xs:string" minOccurs="0"/>
         <xs:element minOccurs="0" ref="Accesses"/>
         <xs:element minOccurs="0" ref="Roles"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

LoginValue Content Description

Name	Category	Type	Cardinality	Description
LoginGroup	element	xs:string	0..1	
Password	element	xs:string	0..1	
Accesses	element	Accesses	0..1	
Roles	element	Roles	0..1	

LoginID Element

Definition




```
<xs:element name="LoginID">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="stringMax20"/>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

Content Description

Options Element

Definition

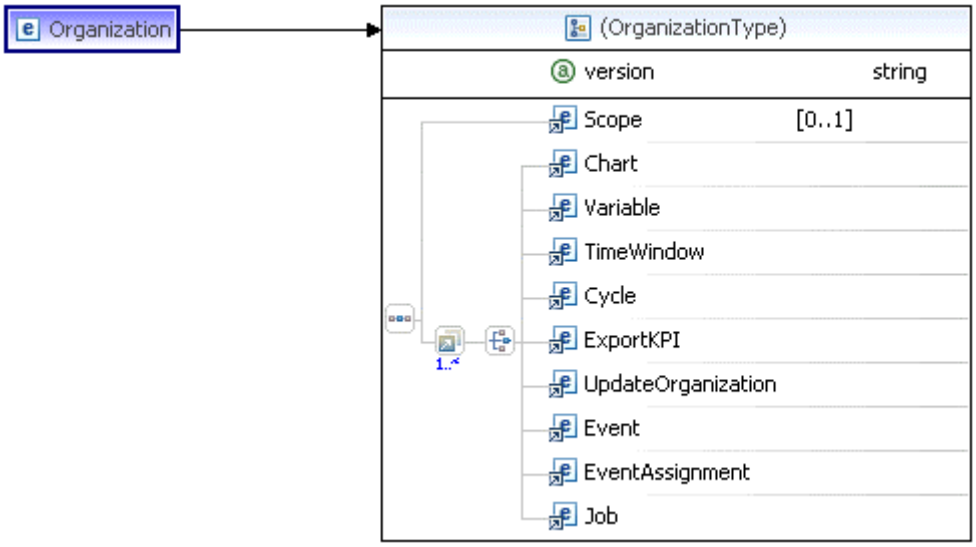

```
<xs:element name="Options">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Field"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Field	element	Field	0..*	

Organization Element

Definition




```
<xs:element name="Organization">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="ows:Scope"/>
         <xs:group maxOccurs="unbounded" ref="Block"/>
      </xs:sequence>
      <xs:attribute default="1.0" name="version" type="xs:string"
use="optional"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
version	attribute	xs:string	0..1	

Scope	element	Scope	0..1	
Chart	element	Chart	0..*	
Variable	element	Variable	0..*	
TimeWindow	element	TimeWindow	0..*	
Cycle	element	Cycle	0..*	
ExportKPI	element	ExportKPI	0..*	
UpdateOrganization	element	UpdateOrganization	0..*	
Event	element	Event	0..*	
EventAssignment	element	EventAssignment	0..*	
Job	element	Job	0..*	

PartyID Element

Definition


```
<xs:element name="PartyID">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="non-empty-stringMax80">
            <xs:attribute name="IDType" type="non-empty-stringMax80"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
IDType	attribute	non-empty-stringMax80	0..1	

PersonIdentification Element

Description

The **PersonIdentification** fragment contains a list of fields identifying an employee. It is not dated and it is valid at any time in the person's life. The target of **PersonIdentification** is an employee.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg:Employee xmlns:msg="http://www.oracle.com/ows/idk/Employee"
                 xmlns="http://www.oracle.com/ows/idk/Core" >
  <PersonIdentification>
    <Scope>
      <EmployeeID IDType="HRID">049996</EmployeeID>
      <StartDate>2004-01-01</StartDate>
      <EndDate>2004-12-31</EndDate>
    </Scope>
    <PersonIdentificationValue>
      <FirstName>Smith</FirstName>
          ...
    </PersonIdentificationValue>
  </PersonIdentification>
</msg:Employee>
```

At least one of the following fields is mandatory:

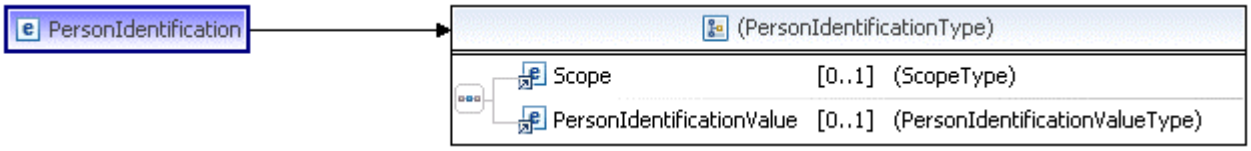
- <FirstName>
- <LastName>

The personal ID refers to one of these fields, and at least one is mandatory:

- <HRID>
- <SSN>
- <Badge>

There may be more mandatory fields. They are described in the CCD (GUI-EMPLOYEE-FIELDS-FIELD, optional attribute = "false").

Definition



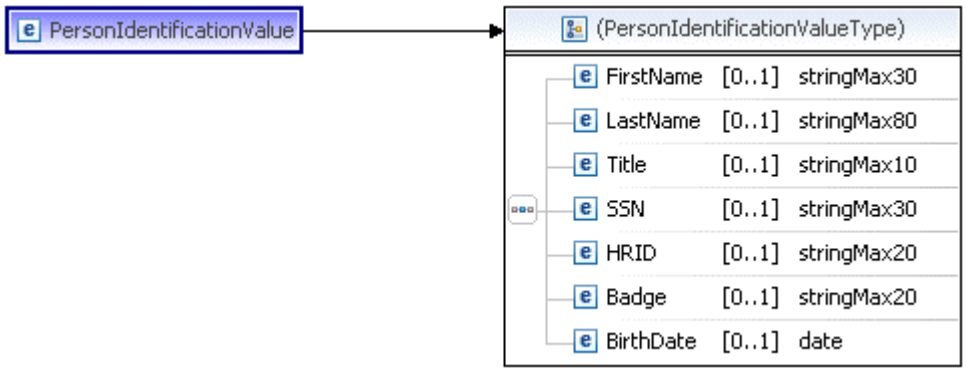

```
<xs:element name="PersonIdentification">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" ref="PersonIdentificationValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	0..1	
PersonIdentificationValue	element	PersonIdentificationValue	0..1	

PersonIdentificationValue Element

PersonIdentificationValue Definition



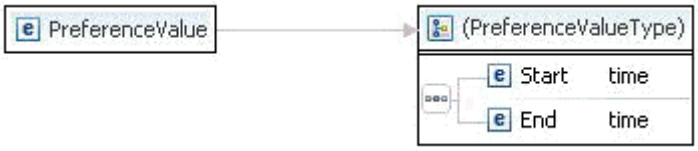
```
<xs:element name="PersonIdentificationValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="FirstName" type="stringMax30" minOccurs="0"/>
         <xs:element name="LastName" type="stringMax80" minOccurs="0"/>
         <xs:element name="Title" type="stringMax10" minOccurs="0"/>
         <xs:element name="SSN" type="stringMax30" minOccurs="0"/>
         <xs:element name="HRID" type="stringMax20" minOccurs="0"/>
         <xs:element name="Badge" type="stringMax20" minOccurs="0"/>
         <xs:element name="BirthDate" type="xs:date" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

PersonIdentificationValue Content Description

Name	Category	Type	Cardinality	Description
FirstName	element	stringMax30	0..1	The employee's first name.
LastName	element	stringMax80	0..1	The employee's surname.
Title	element	stringMax10	0..1	The employee's title, such as Mr, Ms.
SSN	element	stringMax30	0..1	The employee's Social Security Number. This number can be used for selecting the employee in messages rather than the HRID.
HRID	element	stringMax20	0..1	The employee number for the HRMS system. This number is the usual way for selecting an employee in messages.
Badge	element	stringMax20	0..1	The employee's badge number. This number can be used for selecting the employee in messages rather than the HRID.
BirthDate	element	xs:date	0..1	The employee's date of birth.

PreferenceValue Element

Definition




```
<xs:element name="PreferenceValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Start" type="xs:string"/>
         <xs:element name="End" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Start	element	xs:string	1..1	The start time of the preferred hours.
End	element	xs:string	1..1	The end time of the preferred hours.

Punch Element

Definition

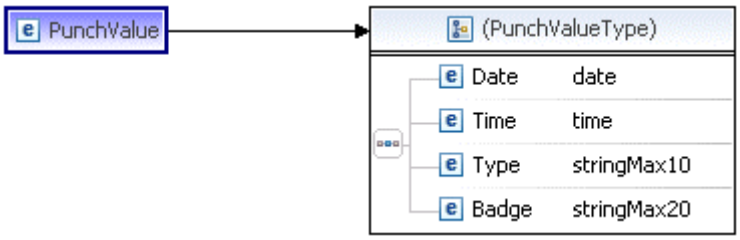

```
<xs:element name="Punch">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="PunchValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
PunchValue	element	PunchValue	0..*	

PunchValue Element

PunchValue Definition



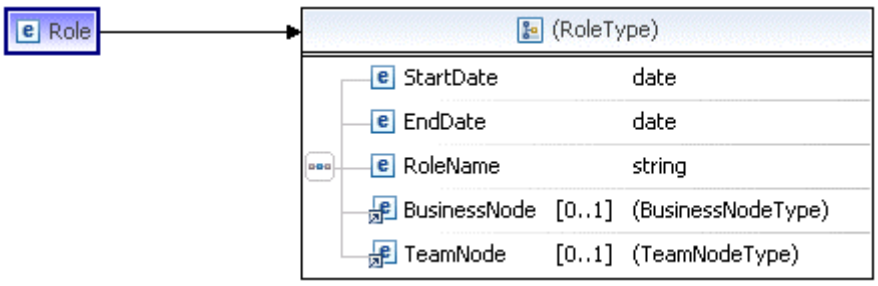
```
<xs:element name="PunchValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="Time" type="xs:time"/>
         <xs:element name="Type" type="stringMax10"/>
         <xs:element name="Badge" type="stringMax20"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

PunchValue Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	1..1	
Time	element	xs:time	1..1	
Type	element	stringMax10	1..1	
Badge	element	stringMax20	1..1	

Role Element

Role Definition




```
<xs:element name="Role">
```

```
   <xs:complexType>
      <xs:sequence>
         <xs:element name="StartDate" type="xs:date"/>
         <xs:element name="EndDate" type="xs:date"/>
         <xs:element name="RoleName" type="xs:string"/>
         <xs:element minOccurs="0" ref="BusinessNode"/>
         <xs:element minOccurs="0" ref="TeamNode"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Role Content Description

Name	Category	Type	Cardinality	Description
StartDate	element	xs:date	1..1	
EndDate	element	xs:date	1..1	
RoleName	element	xs:string	1..1	
BusinessNode	element	BusinessNode	0..1	
TeamNode	element	TeamNode	0..1	

Roles Element

Definition

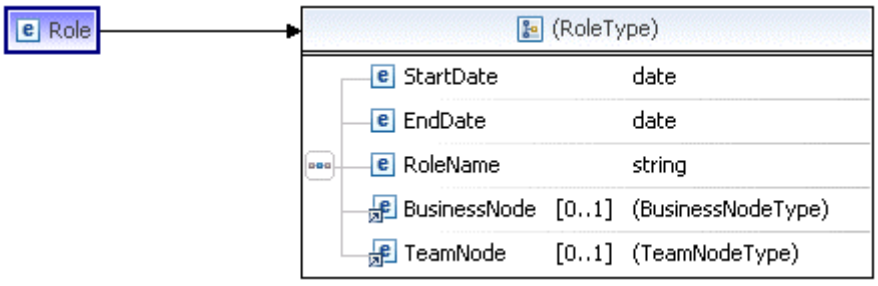

```
<xs:element name="Roles">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Role"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Role	element	Role	0..*	

Role Element

Role Definition




```
<xs:element name="Role">
   <xs:complexType>
```

```
      <xs:sequence>
         <xs:element name="StartDate" type="xs:date"/>
         <xs:element name="EndDate" type="xs:date"/>
         <xs:element name="RoleName" type="xs:string"/>
         <xs:element minOccurs="0" ref="BusinessNode"/>
         <xs:element minOccurs="0" ref="TeamNode"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Role Content Description

Name	Category	Type	Cardinality	Description
StartDate	element	xs:date	1..1	
EndDate	element	xs:date	1..1	
RoleName	element	xs:string	1..1	
BusinessNode	element	BusinessNode	0..1	
TeamNode	element	TeamNode	0..1	

Schedule Element

Description

A **Schedule** fragment has a scope whose target is an employee's file. It provides all shifts for that employee in the given time period.

```
<Schedule>
  <Scope>
    <EmployeeID IDType='HRID' fileID='1'>0464</?>
    <StartDate>2004-01-04</StartDate>
    <EndDate>2004-01-10</EndDate>
  </Scope>
  <Shift>
...
  </Shift>
  <Shift>
...
  </Shift>
</Schedule>
```

<Shift> Elements

A **Shift** element gives detailed information about the work or time off of an employee for one specific activity in a day. A Shift element always has a **<Date>** element, which specifies the day on which the shift starts.

```
<Shift>
  <Date>2004-01-05<Date>
```

An **Activity** element can also be shown as either a routine workday or vacation day.

```
<Activity>sale</Activity>
```

```
The start time and end time of the shift is shown below.

  <StartTime>08:30:00</StartTime>
  <EndTime>11:30:00</EndTime>

The organization unit the employee is scheduled for export only and for an
activity, not an absence).
```

```
<PartyID IDType='StoreID'>0436</PartyID>
```

<WorkingHours>: For an import of absences, it is the number of hours, which the absence is counted for.

```
<WorkingHours>7.5</WorkingHours>
```

Incoming Schedules

Employee messages can contain Schedule fragments that show daily employee absences. The required elements are:

- Date
- Activity: i.e., sick, vacation, other
- WorkingHours: the amount of hours of absence

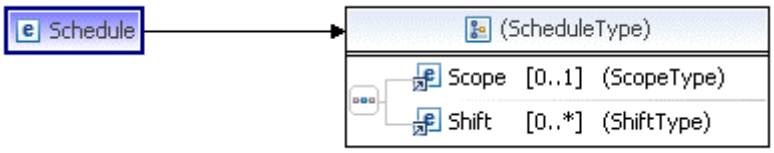
Generated Schedules

Oracle Workforce Scheduling uses the following parameters to generate **HRSchedules** messages:

- Time Period
- An organization unit as a target

Elements exported in **Schedule** fragments are: **Date Activity StartTime EndTime PartyID**.

Definition



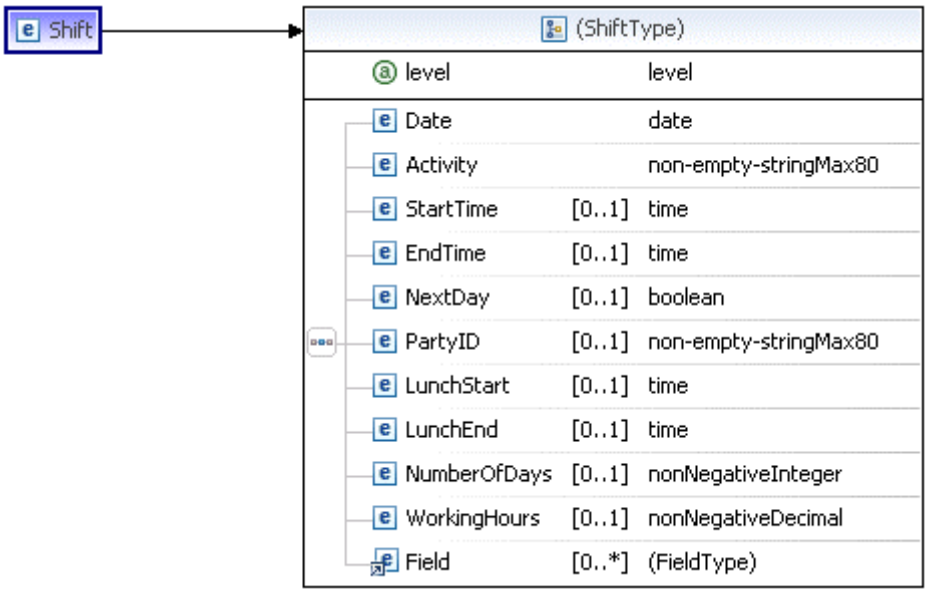

```
<xs:element name="Schedule">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Shift"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	0..1	
Shift	element	Shift	0..*	

Shift Element

Shift Definition




```
<xs:element name="Shift">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="Activity" type="non-empty-stringMax80"/>
         <xs:element name="StartTime" type="xs:time" minOccurs="0"/>
         <xs:element name="EndTime" type="xs:time" minOccurs="0"/>
         <xs:element name="NextDay" type="xs:boolean" minOccurs="0"/>
         <xs:element name="PartyID" type="non-empty-stringMax80"
minOccurs="0"/>
         <xs:element name="LunchStart" type="xs:time" minOccurs="0"/>
         <xs:element name="LunchEnd" type="xs:time" minOccurs="0"/>
         <xs:element name="NumberOfDays" type="xs:nonNegativeInteger"
minOccurs="0"/>
         <xs:element name="WorkingHours" type="nonNegativeDecimal"
minOccurs="0"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Field"/>
      </xs:sequence>
      <xs:attribute name="level" type="level" use="optional"/>
   </xs:complexType>
</xs:element>
```

Shift Content Description

Name	Category	Type	Cardinality	Description
level	attribute	level	0..1	The shift level of detail.
Date	element	xs:date	1..1	The first day on which the activity starts.
Activity	element	non-empty-stringMax80	1..1	The code of the activity or the absence as defined in the CCD.
StartTime	element	xs:time	0..1	The start time of the activity.
EndTime	element	xs:time	0..1	The end time of the activity.
NextDay	element	xs:boolean	0..1	
PartyID	element	non-empty-stringMax80	0..1	For business activities only (not absences), indicating the organization unit the activity is being performed
LunchStart	element	xs:time	0..1	
LunchEnd	element	xs:time	0..1	
NumberOfDays	element	xs:nonNegativeInteger	0..1	
WorkingHours	element	nonNegativeDecimal	0..1	The number of working hours for which the absence is counted. This element appears only in imported absences messages.
Field	element	Field	0..*	

Scope Element

Description

Scope of a Building Fragment

A <Scope> element comprises:

- a target:
    - **<PartyID>** when an organization unit is the target of the fragment
    - **<EmployeeID>** when an employee or the file (contract) of an employee is the target of the fragment
    - **<LoginID>** when a login is the target of the fragment
    - **<WeekType>** to specify the week type when importing the value(eg: in case of Cyclic Chart for Profiles)
    - **<Contract>** when we need to know explicitly on which contract we want to import the value
- a **<StartDate>** sets the first day of the time period: if this date is not present, it is the first day set by Oracle Workforce Scheduling.
- an **<EndDate>** sets the last day of the time period: if this date is not present, it is the last day set by Oracle Workforce Scheduling.

Example:

```
<Scope>
  <PartyID IDType='StoreID'>0464</PartyID>
  <StartDate>2004-01-04</StartDate>
  <EndDate>2004-01-10</EndDate>
</Scope>
```

This scope corresponds to the first week of 2004 for store 0464.

Target of a Scope: Organization Units

They can be stores, districts or any organization unit identified by Oracle Workforce Scheduling. They are identified by an element such as:

```
<PartyID IDType="StoreID">0432</PartyID>
```

Target of a Scope: Employee/File

An employee is identified by:

- A standard employee HR number (HRID)
- A social security number (SSN)
- A badge number (Badge)

Usual identification:

```
<EmployeeID IDType='HRID'>14234</EmployeeID>
```

Identification through SSN:

```
<EmployeeID IDType='SSN'>78956745</EmployeeID>
```

Target of a Scope: Login

A login is identified by its name such as:

```
<LoginID>MyLogin</LoginID>
```

File Identification of an Employee

The attribute fileID identifies a file for a person. When the customer's HR system does not manage several contracts per employee, a dummy default value of **FiledID='1'** must be provided in the messages. The file identification is ignored in the scope of **<PersonalIdentification> <Address> <Contact> <Hiring>** fragments, relative to a person and not to one of her/his files.

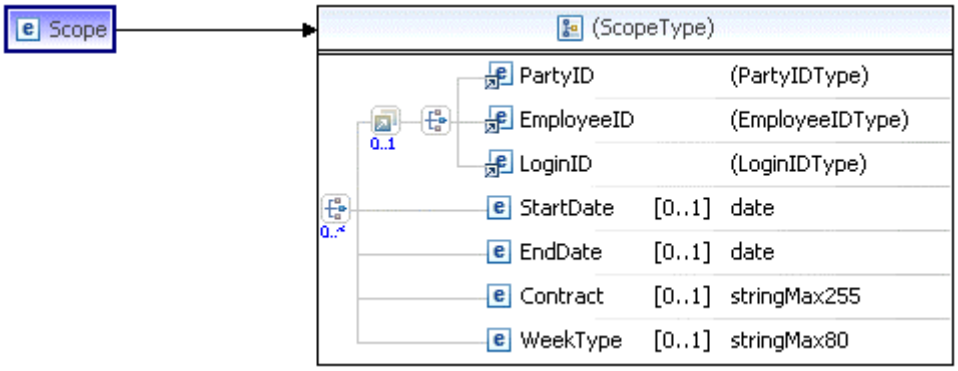
```
<EmployeeID IDType='SSN' fileID='1'>14234</EmployeeID>
```

Scope of a Message

The purpose of a scope of a message is to provide default values for all the fragments contained in that message and which do not have a scope. For Schedule OUT messages, Oracle Workforce Scheduling generates a message with:

- **PartyId** with store identification
- **StartDate EndDate** being the first and last day of the week

Definition




```
<xs:element name="Scope">
   <xs:complexType>
      <xs:choice maxOccurs="unbounded" minOccurs="0">
         <xs:group minOccurs="0" ref="Target"/>
         <xs:element name="StartDate" type="xs:date" minOccurs="0"/>
         <xs:element name="EndDate" type="xs:date" minOccurs="0"/>
         <xs:element name="Contract" type="stringMax255" minOccurs="0"/>
         <xs:element name="WeekType" type="stringMax80" minOccurs="0"/>
      </xs:choice>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
PartyID	element	PartyID	0..*	
EmployeeID	element	EmployeeID	0..*	
LoginID	element	LoginID	0..*	
StartDate	element	xs:date	0..*	
EndDate	element	xs:date	0..*	
Contract	element	stringMax25	0..*	

		5		
WeekType	element	stringMax80	0..*	

Skill Element

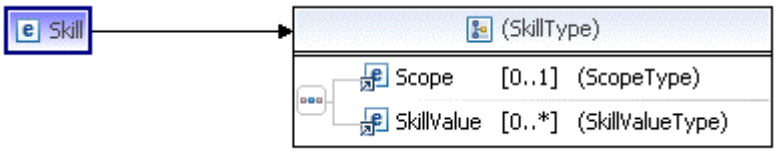
Description

The **Skill** element gives (via the **SkillValue** element) the ordered list of activities that the employee can perform, by order of preference.

```
<Scope>
  <EmployeeID IDType="HRID" fileID="1">002048823</EmployeeID>
</Scope>

<Skill>
  <SkillValue>
    <Date>2004-02-01</Date>
    <Skills>Cashier;Footwear</Skills>
  </SkillValue>
  <SkillValue>
    <Date>2004-06-01</Date>
    <Skills>Softgoods;Footwear</Skills>
  </SkillValue>
</Skill>
```

Definition




```
<xs:element name="Skill">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="SkillValue"/>
```

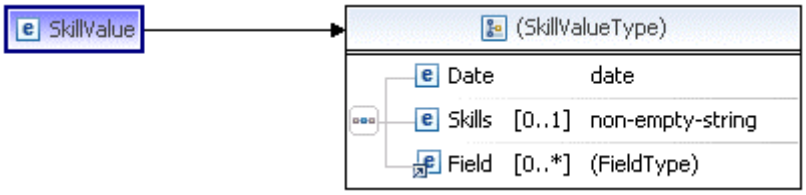
```
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	0..1	
SkillValue	element	SkillValue	0..*	

SkillValue Element

SkillValue Definition




```
<xs:element name="SkillValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="Skills" type="non-empty-string" minOccurs="0"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Field"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

SkillValue Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	1..1	The effective date of the value.
Skills	element	non-empty-	0..1	The skills expressed as a list of values

		string		separated by a semicolon.
Field	element	Field	0..*	

TeamNode Element

Definition


```
<xs:element name="TeamNode">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="non-empty-stringMax80">
            <xs:attribute name="type" type="non-empty-stringMax80"
use="required"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
type	attribute	non-empty-stringMax80	1	

TimeWindow Element

Description

The purpose of a **TimeWindow** fragment is to define a slot of time (Start End) for every day of the week numbered from 0 to 6 and an optional quantity, for every type of week.

All time windows that can be imported are described in the CCD, and so are specific to each customer. Week type names are defined in the CCD. For example: NormalWeek PeakWeek LowWeek EasternWeek, etc. The **TimeWindow** element does not specify when the defined slots are effectively applied. This is the purpose of the **Cycle** fragment. The **Cycle** fragment specifies what is the sequence of week types, for a given time period. For example, the sequence specified by a **Cycle** fragment is Normal Normal Peak for a 12 weeks time scope. The sequence of the three weeks (Normal Normal Peak) is cyclically repeated until the 12 weeks time scope ends.

TimeWindow

The name of the **TimeWindow** fragment is defined in the CCD. The scope has a **PartyID** tag because the data is linked to the business organization. The Start and End values are rounded to the nearest quarter hour. Index values belong to the interval [0;6], where 0 corresponds to the first day of the week and 6 to the last day. A day that is not set in the message appears as a closed day. An import on an existing time window, replaces previous values.

```
<msg:Organization xmlns:msg=...>
  <TimeWindow name="StoreOpeningHours">
    <Scope>
      <PartyID IDType="StoreID" >313</PartyID>
    </Scope>
    <WeekType name="NormalWeek">
      <TimeWindowValue>
        <Index>0</Index>
        <Start>09:00:00</Start>
        <End>18:00:00</End>
      </TimeWindowValue>
              ...
    </ WeekType >
    <WeekType name="LowWeek">
```

```
                ...
     </WeekType>
                ...
   </TimeWindow>
</msg:Organization>
```

Cycle

The purpose of the <Cycle> fragment is to specify which time window is to apply for which effective week: the start and end scope gives the interval to which the cycle is applied. The scope can have more weeks that the number of weeks specified in sequence: they are cyclically repeated. All index of a week in the sequence values MUST be filled in, NO BREAK in an INDEX value is allowed. Specifically, all values from 0 to the maximum index value are filled in for <Cvalue> tag. <Cycle> is attached to an organization and gives a series to iterate over the scope. The cycle begin with the type of week given in offset value, this integer refers to one of week type of the series. One single <CycleValue> MUST be specified (it gives the full sequence of N weeks). There can be more than one type of cycle for a given store (business).

```
<msg:Organization xmlns:msg=...>
  <Cycle>
    <Scope>
      <PartyID IDType="StoreID" >313</PartyID>
      <StartDate>2004-01-01</StartDate>
      <EndDate>2004-12-31</EndDate>
    </Scope>
    <CycleValue>
      <RotaName>Rota1</RotaName>
      <OffSet>2</OffSet>
      <CValue index="0">NormalWeek</CValue>
      <CValue index="1">NormalWeek</CValue>
      <CValue index="2">LowWeek</CValue>
            ...
    </CycleValue>
  </Cycle>
</msg:Organization>
```

Following is an example of an incorrect fragment due to the wrong numbering of weeks in the sequence:

```
<?xml version="1.0" encoding="UTF-8"?>
<msg:Organization xmlns:msg=...>
  <Cycle>
    <Scope>
```

```
      <PartyID IDType="StoreID" >313</PartyID>
      <StartDate>2004-01-01</StartDate>
      <EndDate>2004-12-31</EndDate>
    </Scope>
    <CycleValue>
      <RotaName>Rota1</RotaName>
      <OffSet>2</OffSet>
      <CValue index="0">NormalWeek</CValue>
      <CValue index="2">LowWeek</CValue>
    </CycleValue>
  </Cycle>
</msg:Organization>
```

Error, no <Cvalue> tag with index value equals to 1.

Creation:

You can create a new cycle, when the following conditions are met:

- <RotaName> value is not the id of an existing cycle.
- <OffSet> value is less than or equal to the upper index.
- All values from 0 to the maximum index are filled in for <CValue> tag
- <CValue> value is an existing week type.

Deletion:

You can delete a cycle, when the following conditions are met:

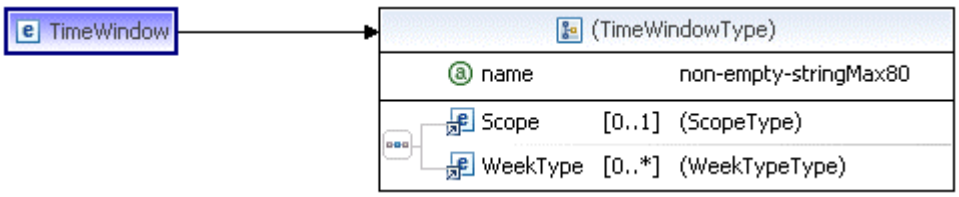
- <RotaName> value is the id of an existing cycle.
- <OffSet> value is equal to -1.
- No <CValue> tag.

Update:

You can update a cycle, when the following conditions are met:

- <RotaName> value is the id of an existing cycle.
- <OffSet> value is less than the upper index.
- The number of weeks must be the same
- <CValue> value is an existing week type.

Definition



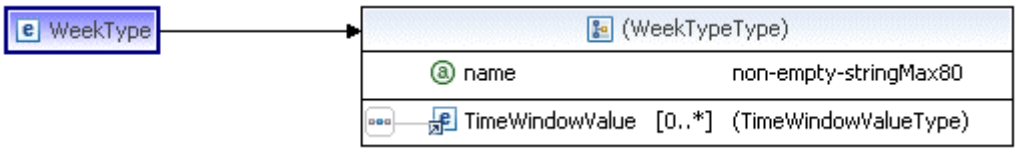

```
<xs:element name="TimeWindow">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="WeekType"/>
      </xs:sequence>
      <xs:attribute name="name" type="non-empty-stringMax80" use="required"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
name	attribute	non-empty-stringMax80	1	Name of the time window are defined in the CCD.
Scope	element	Scope	0..1	
WeekType	element	WeekType	0..*	

WeekType Element

WeekType Definition




```
<xs:element name="WeekType">
   <xs:complexType>
```

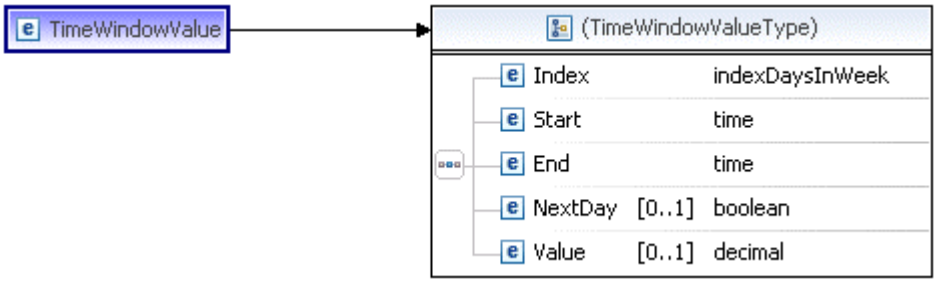
```
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="TimeWindowValue"/>
      </xs:sequence>
      <xs:attribute name="name" type="non-empty-stringMax80" use="required"/>
   </xs:complexType>
</xs:element>
```

WeekType Content Description

Name	Category	Type	Cardinality	Description
name	attribute	non-empty-stringMax80	1	
TimeWindowValue	element	TimeWindowValue	0..*	

TimeWindowValue Element

TimeWindowValue Definition



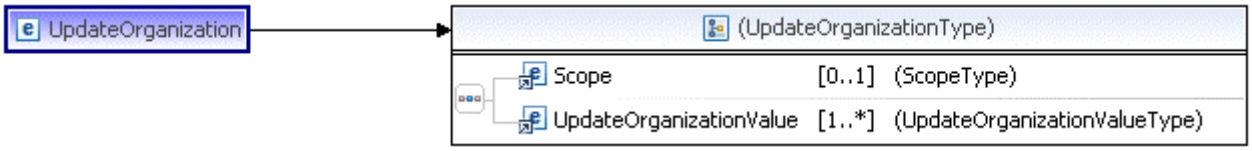

```
<xs:element name="TimeWindowValue">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Index" type="indexDaysInWeek"/>
         <xs:element name="Start" type="xs:time"/>
         <xs:element name="End" type="xs:time"/>
         <xs:element name="NextDay" type="xs:boolean" minOccurs="0"/>
         <xs:element name="Value" type="xs:decimal" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

TimeWindowValue Content Description

Name	Category	Type	Cardinality	Description
Index	element	indexDaysInWeek	1..1	Index values belong to the interval [0;6], where 0 corresponds to the first day of the week and 6 to the last day. A day that is not set in the message appears as a closed day.
Start	element	xs:time	1..1	The start time of the slot.
End	element	xs:time	1..1	The end time of the slot.
NextDay	element	xs:boolean	0..1	
Value	element	xs:decimal	0..1	The quantity associated with the slot.

UpdateOrganization Element

Definition



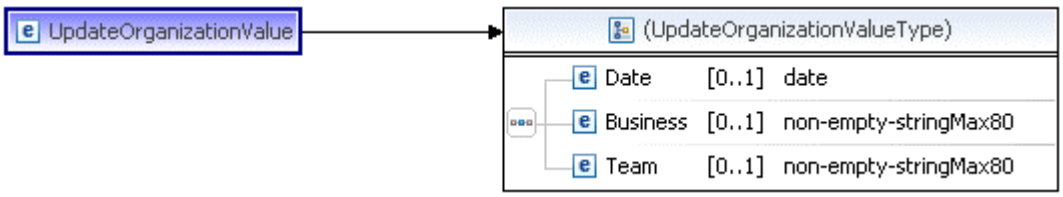

```
<xs:element name="UpdateOrganization">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element maxOccurs="unbounded" ref="UpdateOrganizationValue"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Scope	element	Scope	0..1	
UpdateOrganizationValue	element	UpdateOrganizationValue	1..*	

UpdateOrganizationValue Element

UpdateOrganizationValue Definition




```
<xs:element name="UpdateOrganizationValue">
```

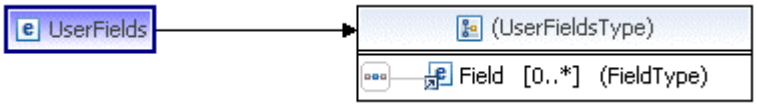
```
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Date" type="xs:date" minOccurs="0"/>
         <xs:element name="Business" type="non-empty-stringMax80" minOccurs="0"/>
         <xs:element name="Team" type="non-empty-stringMax80" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

UpdateOrganizationValue Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	0..1	
Business	element	non-empty-stringMax80	0..1	
Team	element	non-empty-stringMax80	0..1	

UserFields Element

Definition




```
<xs:element name="UserFields">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Field"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Field	element	Field	0..*	

Variable Element

Description

A **Variable** fragment is a dated history of changes, for a time period marked by explicit start and end dates. History records have an effective date and one or more fields. The attribute name of a **Variable** fragment defines the variable and its associated list of fields.

For example, for a Store

- The <Variable name='Store'> variable has fields such as <Field name='Manager'> <Field name='Location'> etc.
- The <Variable name='Organization'> variable allows you to describe two hierarchies for the organization: Team and Business. It has predefined fields BusinessParent,TeamParent UnitName.

A variable can be related to an employee, to an employee file (such as Contract), or to an organizational unit (such as Store).

Dated Variable

Inside a time period (defined by a start date and end date specified by a **Scope** element), the value of all fields that belong to a variable may change over time. If one of those fields changes its value at time t, then all the fields must be given as well.

The rule used to process dated **Variable** elements is the following:

- Values before the start time are left unchanged
- Values after the end date are left unchanged
- Values in the middle are replaced by those given in the Variable fragment

```
<Variable name='Store'>
  <Scope>
    <PartyID IDType='StoreID'>0464</PartyID>
    <StartDate>2003-11-01</StartDate>
    <EndDate>2004-01-01</EndDate>
  </Scope>
```

```
  <Value>
    <Date>2003-11-01</Date>
    <Field name='squareFootage'>5000</Field>
    <Field name='managerContact'>bla</Field>
  </Value>
  <Value>
    <Date>2003-12-01</Date>
    <Field name='squareFootage'>0</Field>
    <Field name='managerContact'>bla</Field>
  </Value>
</Variable>
```

Open Variable Fragments

A variable fragment is called open, when its list of fields is defined at the configuration phase of the application. The variable may be different for each customer:

- <Variable name='Store'> and Department, District, etc.

Note: For each class of an organization, only one variable may be defined.

Predefined Variable Fragments

The following fragments are predefined, i.e. their list of fields cannot be configured. Oracle Workforce Scheduling determines the content:

- PersonIdentification is the fragment containing all permanent fields (not dated) that are attributed to an employee. The fields are first name, last name, title, gender, etc.
- Address is the fragment containing the fields of an employee address (dated)
- Contact is the fragment containing the contact information of an employee (phone numbers, etc...)
- Assignment is the fragment declaring the assignment history of an employee
- Skill is a fragment declaring the history of an employee's skills, an ordered list of activities that the employee may have.

Extendible Variable Fragments

The fragments <Hiring> and <Contract> are predefined and have predefined fields. However, it is possible to add configured fields in the form <Field name="myField1">.

- Hiring gives dated fields, which may be attributed to an employee (the HireDate is predefined).
- Contract gives dated fields , which may be attributed to an employee's contract (MinWorkingDays, etc.).

Rules for Writing Variable Fragments

The name of a variable also defines the type of a target, organization unit, employee's file, or employee. Dates are given in ascending order. Dates before the start date or after the end date are ignored (or truncated), and the scope time period takes precedence. Open Variable Fragments These variables may differ for each customer:

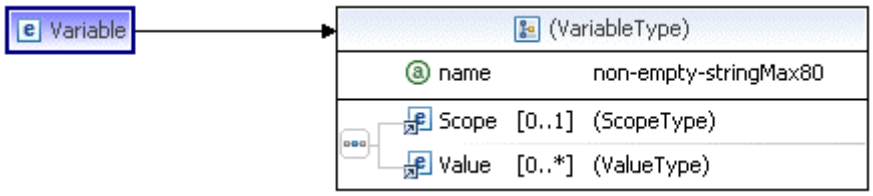
- <Variable name='Store'>, Department, District, etc. For each class of an organization unit, one variable may be defined.
- Some fields are predefined such as <BusinessParent> <TeamParent> <UnitName>. Declaring such values allows organization unit hierarchies to be defined (business or team as well) for the predefined <Variable name='Organization'> fragment.

Variable Processing

The standard processing for IN messages is simply a time slot replacement of variable fields values. However, for some variables, such as Contract, a more complex analysis is performed and the validation of such a fragment is not automatic. Values can be rejected in case of inconsistency. For each field element, there is a specific syntax and an internal technical representation. Variable fragments appear in:

- IN message Organization for variables with an organization unit as a target: <Variable name='Store'> <Variable name='Organization'>
- IN message Employee for variables with an employee (file) as a target: <PersonIdentification> <Address> <Contact> <Hiring> <Contract> <Assignment>.

Definition



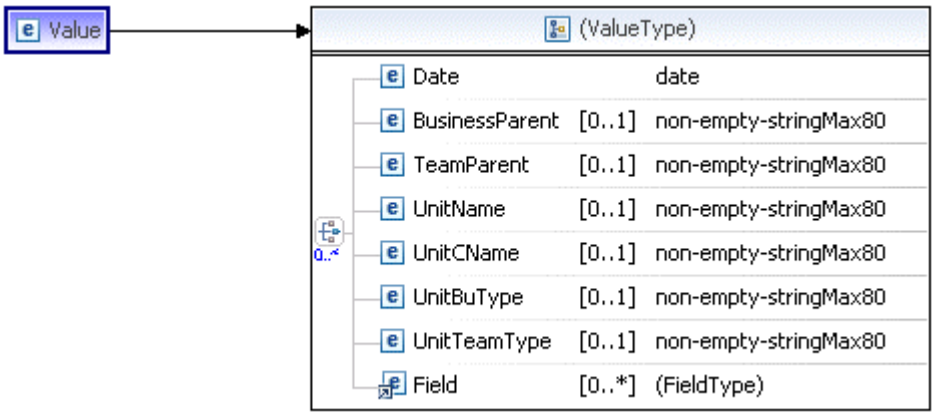

```
<xs:element name="Variable">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" ref="Scope"/>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Value"/>
      </xs:sequence>
      <xs:attribute name="name" type="non-empty-stringMax80" use="required"/>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
name	attribute	non-empty-stringMax80	1	
Scope	element	Scope	0..1	
Value	element	Value	0..*	

Value Element

Value Definition




```
<xs:element name="Value">
   <xs:complexType>
      <xs:choice maxOccurs="unbounded" minOccurs="0">
         <xs:element name="Date" type="xs:date"/>
         <xs:element name="BusinessParent" type="non-empty-stringMax80"
minOccurs="0"/>
         <xs:element name="TeamParent" type="non-empty-stringMax80"
minOccurs="0"/>
         <xs:element name="UnitName" type="non-empty-stringMax80" minOccurs="0"/>
         <xs:element name="UnitCName" type="non-empty-stringMax80"
minOccurs="0"/>
         <xs:element name="UnitBuType" type="non-empty-stringMax80"
minOccurs="0"/>
         <xs:element name="UnitTeamType" type="non-empty-stringMax80"
minOccurs="0"/>
```

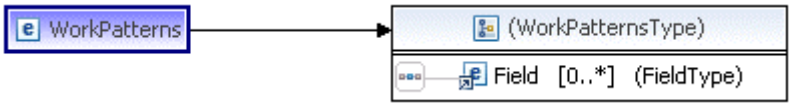
```
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="Field"/>
      </xs:choice>
   </xs:complexType>
</xs:element>
```

Value Content Description

Name	Category	Type	Cardinality	Description
Date	element	xs:date	0..*	The effective date of the value.
BusinessParent	element	non-empty-stringMax80	0..*	The code of the business parent in the "business hierarchy" as defined in the CCD.
TeamParent	element	non-empty-stringMax80	0..*	The code of the business parent in the "team hierarchy" as defined in the CCD.
UnitName	element	non-empty-stringMax80	0..*	The long name of the business unit
UnitCName	element	non-empty-stringMax80	0..*	
UnitBuType	element	non-empty-stringMax80	0..*	
UnitTeamType	element	non-empty-stringMax80	0..*	
Field	element	Field	0..*	

WorkPatterns Element

Definition



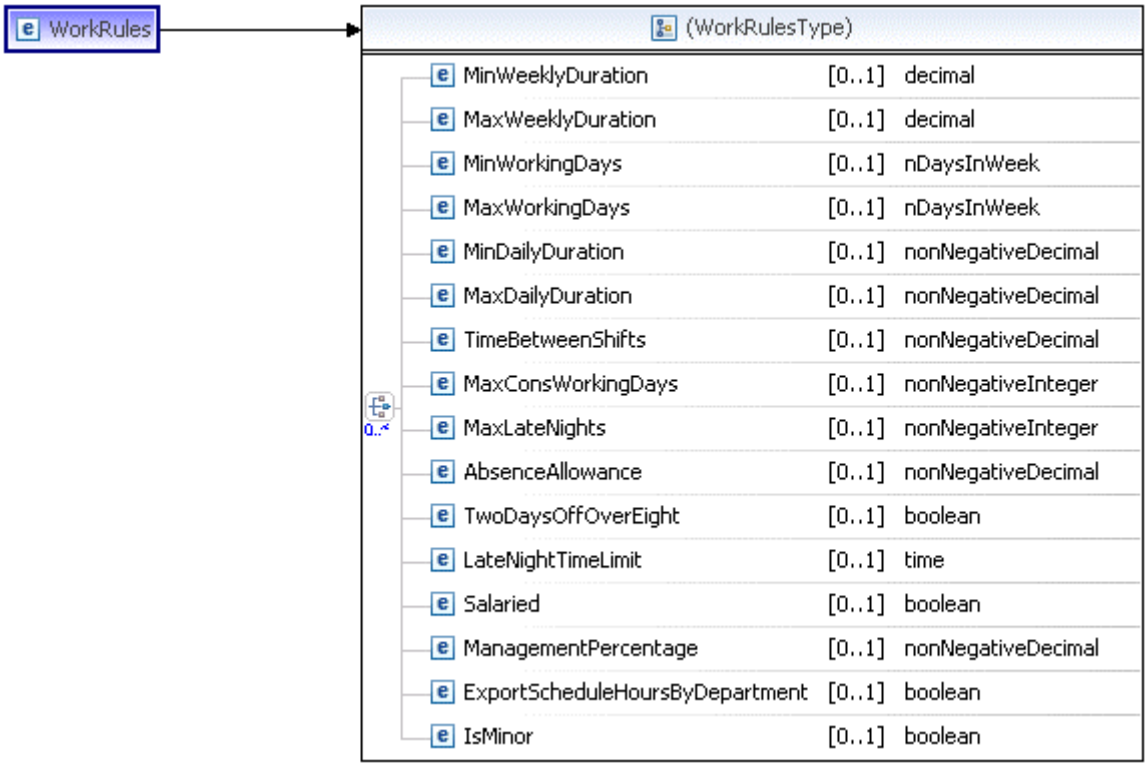

```
<xs:element name="WorkPatterns">
   <xs:complexType>
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="Field"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
Field	element	Field	0..*	

WorkRules Element

Definition




```
<xs:element name="WorkRules">
   <xs:complexType>
      <xs:choice maxOccurs="unbounded" minOccurs="0">
         <xs:element name="MinWeeklyDuration" type="xs:decimal" minOccurs="0"/>
         <xs:element name="MaxWeeklyDuration" type="xs:decimal" minOccurs="0"/>
         <xs:element name="MinWorkingDays" type="nDaysInWeek" minOccurs="0"/>
         <xs:element name="MaxWorkingDays" type="nDaysInWeek" minOccurs="0"/>
         <xs:element name="MinDailyDuration" type="nonNegativeDecimal"
minOccurs="0"/>
         <xs:element name="MaxDailyDuration" type="nonNegativeDecimal"
minOccurs="0"/>
         <xs:element name="TimeBetweenShifts" type="nonNegativeDecimal"
```

```
minOccurs="0"/>
         <xs:element name="MaxConsWorkingDays" type="xs:nonNegativeInteger"
minOccurs="0"/>
         <xs:element name="MaxLateNights" type="xs:nonNegativeInteger"
minOccurs="0"/>
         <xs:element name="AbsenceAllowance" type="nonNegativeDecimal"
minOccurs="0"/>
         <xs:element name="TwoDaysOffOverEight" type="xs:boolean" minOccurs="0"/>
         <xs:element name="LateNightTimeLimit" type="xs:time" minOccurs="0"/>
         <xs:element name="Salaried" type="xs:boolean" minOccurs="0"/>
         <xs:element name="ManagementPercentage" type="nonNegativeDecimal"
minOccurs="0"/>
         <xs:element name="ExportScheduleHoursByDepartment" type="xs:boolean"
minOccurs="0"/>
         <xs:element name="IsMinor" type="xs:boolean" minOccurs="0"/>
      </xs:choice>
   </xs:complexType>
</xs:element>
```

Content Description

Name	Category	Type	Cardinality	Description
MinWeeklyDuration	element	xs:decimal	0..*	
MaxWeeklyDuration	element	xs:decimal	0..*	
MinWorkingDays	element	nDaysInWeek	0..*	
MaxWorkingDays	element	nDaysInWeek	0..*	
MinDailyDuration	element	nonNegativeDecimal	0..*	
MaxDailyDuration	element	nonNegativeDecimal	0..*	
TimeBetweenShifts	element	nonNegativeDecimal	0..*	
MaxConsWorkingDays	element	xs:nonNegativeInteger	0..*	
MaxLateNights	element	xs:nonNegativeInteger	0..*	

AbsenceAllowance	element	nonNegativeDecimal	0..*	
TwoDaysOffOverEight	element	xs:boolean	0..*	
LateNightTimeLimit	element	xs:time	0..*	
Salaried	element	xs:boolean	0..*	
ManagementPercentage	element	nonNegativeDecimal	0..*	
ExportScheduleHoursByDepartment	element	xs:boolean	0..*	
IsMinor	element	xs:boolean	0..*	