Title of Document

Status of This Document

Choose one of:

Grid Working Document (GWD)

Grid Final Draft (GFD)

Grid Recommendation

Obsolete. This document is replaced by/obsoleted by GFD-xxx [REFERENCE].

Historical

Document Change History

Feb 2010 – v 0\_2 – initial structure off the document added

March-May 2010 – v0\_3 - added introduction, motivation, Instrument Element description

Copyright Notice

Copyright © Open Grid Forum (2010,2011). Some Rights Reserved. Distribution is unlimited.

Trademark

XXXX is a registered trademark and service mark of the Open Grid Forum. [include if applicable]

Abstract

Informative document abstract of 1-2 paragraphs.

This document provides information to the community on interfaces for accessing remote instruments in distributed environment. In this specification we describe the Instrument Element (IE) service that provides the e-Infrastructure with an abstraction of real instruments and  grid  users   with  an   interactive  interface  to  access and control  them. The IE represents a virtualization of instruments and sensors and data sources or a virtualization of group of instruments, sensors and data sources. The interface to a single instrument is called Instrument Manager (IM). It is a protocol adapter that allows the middleware to talk to the physical instrument or more precisely, its control system.

The instruments are described by Shared Information Model.

Distribution is unlimited.

Contents

[Abstract 1](#_Toc294000451)

[Contents 1](#_Toc294000452)

[1 Introduction 3](#_Toc294000453)

[1.1 Motivation 3](#_Toc294000454)

[1.2 Definition of the Term Activity within the Scope of this Document 3](#_Toc294000455)

[2 Notational Conventions 4](#_Toc294000456)

[2.1 Namespaces 4](#_Toc294000457)

[3 Use Cases 4](#_Toc294000458)

[4 Architecture 4](#_Toc294000459)

[5 Instrument Element 4](#_Toc294000460)

[Instrument Manager DTD 12](#_Toc294000461)

[6 Security Considerations 14](#_Toc294000462)

[7 Glossary 14](#_Toc294000463)

[8 Contributors 15](#_Toc294000464)

[9 Acknowledgments 15](#_Toc294000465)

[10 Intellectual Property Statement 15](#_Toc294000466)

[11 Disclaimer 16](#_Toc294000467)

[12 Full Copyright Notice 16](#_Toc294000468)

[13 References 16](#_Toc294000469)

# Introduction

This document provides information to the community on interfaces for accessing remote instruments in distributed environment.

In this specification we describe the Instrument Element (IE) service that provides the e-Infrastructure with an abstraction of real instruments and  grid  users   with  an   interactive  interface  to  access and control  them. The IE represents a virtualization of instruments and sensors and data sources or a virtualisation of groups of instruments, sensors or data soruces. The interface to a single instrument is called Instrument Manager (IM). It is a protocol adapter that allows the middleware to talk to the physical instrument or more precisely, its control system.

The instruments are described by Shared Information Model.

## Motivation

Integration of scientific instruments into e-Infrastructure empowers possibilities in conducting experiments. Access to equipment infrastructures is very often a precondition for successful research in many scientific disciplines. The necessity of using unique and expensive equipment, which is often locally unavailable, is the key issue for the success of a number of experiments. Thus the spreading of remote instrumentation techniques and technologies that allow remote and shared access to laboratory instruments opens up new opportunities for researchers. The term instrumentation covers a wide range of laboratory equipment that is necessary for experimental sciences, such as biochemistry, physics and astronomy. Remote instrumentation is the method of providing data acquisition and/or control of scientific instruments from remote locations.

Integration with the infrastructures like Grid or HPC is used to integrate operations on instruments with computing farms where complex models and computations coming from instruments could run, and store a large amount of data. The Grid handles issues related to authorization, resource management, data transfer and storing. The network infrastructure is used with the available mechanism for QoS handling. Instrumentation as a service allows composing atomic experimental actions into measurement chains irrelevant of the location of the cooperating instruments.

There are many ways to access instruments in a Grid environment. The first attempt to standardize the access to instruments was the GTCP (Globus Teleoperation Control Protocol) by C.Kassleman and L.Pearlman. GTCP together with the emerging control systems inspired the design of the Instrument Element by the GRIDCC project later (maintained by ELETTRA) the DORII project. A parallel attempt was named CIMA (Common Instrument Middleware Architecture) by R.McMullen.

There is a need for accessing remote instruments by using standard interfaces that are well defined and specified by standard approaches/procedures.

## Definition of the Term Activity within the Scope of this Document

### Goal

This version of the specification aims to:

* Describe Instrument Element interfaces
* Describe Instrument Manager interfaces
* Describe the Shared Information Model (SIM)

### Out of Scope

For this version of the specification we consider the following topics to be out of scope. We will

consider these important topics for consideration in future revisions:

# Notational Conventions

Only include this section if applicable.

The key words ‘MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC 2119 [BRADNER], except that the words do not appear in uppercase.

In addition to the terms introduced in **[RFC2119]**, additional terms commonly used in this

document are defined in the Glossary in the back.

## Namespaces

The following is an XML or other code example:

The following namespaces are used in this document:

|  |  |
| --- | --- |
| **Prefix** | **Namespace** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| xs/xsd | http://www.w3.org/2001/XMLSchema |
| Xsi | <http://www.w3.org/2001/XMLSchema-instance> |
| Wsdl | <http://schemas.xmlsoap.org/wsdl> |

# Use Cases

There are many use cases of the Instrument Element concept steaming from the requirements of system level science. In simple words scientist need often to compare the result of real measurements done via real instruments or sensors and the results of in-silico experiments or simulations. The natural workflow involves then integrating instruments with the traditional Grid resources like storage and clusters.

The concept of instrument is on the other side quite generic and can be considered as a way to integrate in an existing system a plethora of other components.

Another interesting and emerging source of use cases is growing from the basic product of most research infrastructure, namely, scientific data sets and from the need to curate, preserve and track what happens to scientific data during their whole lifecycle.

Instrument Elements can in fact play an important role at the beginning of the scientific data lifecycle as they are closer than any other component to the data sources.

Instrument Elements can get a dataset produced by real instrument, add metadata coming from other sources like scientific business management systems and from the experimental field and transfer the data to data catalogues after appropriate conversion in standard formats.

This step can really add value to scientific data and make them easily usable by all the interested stakeholders.

# Architecture

# Instrument Element

**IE connection:**

A set of operations that regard user sessions with the IE server. Client interaction with the IE starts with opening a session. Successive calls to the IE must contain the session ID information.

Opens a session to the Instrument Element. This is the first call that starts the interaction with the IE. This method returns a String that is the identifier for the session. That ID must be used by all other requests to the IE.

**public** String openSession(String userID, String password)

<wsdl:message name=*"openSessionRequest"*>

<wsdl:part element=*"tns1:openSession"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"openSessionResponse"*>

<wsdl:part element=*"tns1:openSessionResponse"* name=*"parameters"* />

</wsdl:message>

**public** String openSessionWithCredential(String credential)

<wsdl:message name=*"openSessionWithCredentialResponse"*>

<wsdl:part element=*"tns1:openSessionWithCredentialResponse"*

name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"openSessionWithCredentialRequest"*>

<wsdl:part element=*"tns1:openSessionWithCredential"* name=*"parameters"* />

</wsdl:message>

Closes the session. This method will invalidate the session identifier, so all successive class with that session identifier will fail.

**public** **void** closeSession(String sessionID)

<wsdl:message name=*"closeSessionRequest"*>

<wsdl:part element=*"tns1:closeSession"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"closeSessionResponse"*>

<wsdl:part element=*"tns1:closeSessionResponse"* name=*"parameters"* />

</wsdl:message>

**IE browsing resources (IMs):**

Set of operations that regard browsing of the deployed resources on the IE server. The resources are represented by Instrument Managers that may be grouped in Contexts. (Contexts may be further grouped in other Contexts creating a tree like structrure.)

Returns all Instrument Managers IDs that are children of the root context of this Instrument Element.

**public** String[] getAllTopInstrumentManagers(String sessionID)

<wsdl:message name=*"getAllTopInstrumentManagersRequest"*>

<wsdl:part element=*"tns1:getAllTopInstrumentManagers"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getAllTopInstrumentManagersResponse"*>

<wsdl:part element=*"tns1:getAllTopInstrumentManagersResponse"*

name=*"parameters"* />

</wsdl:message>

Rerturns all context IDs that are children of the root context of this Instrument Element.

**public** String[] getContexts(String sessionID)

<wsdl:message name=*"getContextsRequest"*>

<wsdl:part element=*"tns1:getContexts"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getContextsResponse"*>

<wsdl:part element=*"tns1:getContextsResponse"* name=*"parameters"* />

</wsdl:message>

Returns the array of IDs for nested contexts of the specified context path.

**public** String[] getContextsInContext(String sessionID, String[] context)

<wsdl:message name=*"getContextsInContextRequest"*>

<wsdl:part element=*"tns1:getContextsInContext"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getContextsInContextResponse"*>

<wsdl:part element=*"tns1:getContextsInContextResponse"* name=*"parameters"* />

</wsdl:message>

Returns the Instrument Manager IDs of the specified context.

**public** String[] getInstrumentManagersInContext(String sessionID,

String[] contextPath)

<wsdl:message name=*"getInstrumentManagersInContextRequest"*>

<wsdl:part element=*"tns1:getInstrumentManagersInContext"*

name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getInstrumentManagersInContextResponse"*>

<wsdl:part element=*"tns1:getInstrumentManagersInContextResponse"*

name=*"parameters"* />

</wsdl:message>

**IM connection:**

A set of operations that regard client connection with an Instrument Manager. A client is either attached or dettached from an IM. A client must attach to an IM before interacting with it.

Returns true if the user is attached to the specified IM, false otherwise.

**public** **boolean** isAttachedToIM(String sessionID, String instrumentManagerID)

<wsdl:message name=*"isAttachedToIMRequest"*>

<wsdl:part element=*"tns1:isAttachedToIM"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"isAttachedToIMResponse"*>

<wsdl:part element=*"tns1:isAttachedToIMResponse"* name=*"parameters"* />

</wsdl:message>

Attaches an user to an Instrument Manager. This method should be used to start the interaction with the specified Instrument Manager.

**public** **void** attachToIM(String sessionID, String instrumentManagerID)

<wsdl:message name=*"attachToIMRequest"*>

<wsdl:part element=*"tns1:attachToIM"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"attachToIMResponse"*>

<wsdl:part element=*"tns1:attachToIMResponse"* name=*"parameters"* />

</wsdl:message>

Detaches an user from an Instrument Manager. This method should be used to close the interaction with an Instrument Manager.

**public** **void** detachFromIM(String sessionID, String instrumentManagerID)

<wsdl:message name=*"detachFromIMRequest"*>

<wsdl:part element=*"tns1:detachFromIM"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"detachFromIMResponse"*>

<wsdl:part element=*"tns1:detachFromIMResponse"* name=*"parameters"* />

</wsdl:message>

**IM info:**

A couple of operations that return the IM description in human-readable format and the current IM state. The description is formatted in a XML file and contains information about the IM state machine, attributes and parameters (type, name, etc.), commands (input parameters) and transitions.

Returns Instrument Manager description or additional information. In the current implementation the IM's XML is returned. The XML is validated using the IM (1) DTD file.

**public** String getInfo(String sessionID, String instrumentManagerID)

<wsdl:message name=*"getInfoRequest"*>

<wsdl:part element=*"tns1:getInfo"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getInfoResponse"*>

<wsdl:part element=*"tns1:getInfoResponse"* name=*"parameters"* />

</wsdl:message>

Returns the state for the specified IM.

**public** String getState(String sessionID, String instrumentManagerID)

<wsdl:message name=*"getStateRequest"*>

<wsdl:part element=*"tns1:getState"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getStateResponse"*>

<wsdl:part element=*"tns1:getStateResponse"* name=*"parameters"* />

</wsdl:message>

**IM Actions:**

A set of operations that regard interaction with an attached Instrument Manager. These operations include reading and writing of attributes and parameters, execution of commands and transitions, locking of the IM and subription to an IM attribute for asynchronous (JMS-based) monitoring. Parameters represent device settings, while attributes are device variables. It would make sense that the attribute access is read-only but some control systems allow attribute settings so we kept the possibility for the moment. Transitions are commands that trigger state-change on the device (e.g. turn-on).

Returns the specified IM attribute.

**public** InstrumentManagerAttribute getAttribute(String sessionID,

String instrumentManagerID, String attributeName)

<wsdl:message name=*"getAttributeRequest"*>

<wsdl:part element=*"tns1:getAttribute"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getAttributeResponse"*>

<wsdl:part element=*"tns1:getAttributeResponse"* name=*"parameters"* />

</wsdl:message>

Returns a list of all attributes for the given IM.

**public** InstrumentManagerAttribute[] getAttributes(String sessionID,

String instrumentManagerID, String[] attributesNames)

<wsdl:message name=*"getAttributesRequest"*>

<wsdl:part element=*"tns1:getAttributes"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getAttributesResponse"*>

<wsdl:part element=*"tns1:getAttributesResponse"* name=*"parameters"* />

</wsdl:message>

Returns a list of all IM attributes for the given IM.

**public** InstrumentManagerAttribute[] getAllAttributes(String sessionID,

String instrumentManagerID)

<wsdl:message name=*"getAllAttributesRequest"*>

<wsdl:part element=*"tns1:getAllAttributes"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getAllAttributesResponse"*>

<wsdl:part element=*"tns1:getAllAttributesResponse"* name=*"parameters"* />

</wsdl:message>

Returns a IM parameter.

**public** InstrumentManagerParameter getParameter(String sessionID,

String instrumentManagerID, String parameterName)

<wsdl:message name=*"getParameterRequest"*>

<wsdl:part element=*"tns1:getParameter"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getParameterResponse"*>

<wsdl:part element=*"tns1:getParameterResponse"* name=*"parameters"* />

</wsdl:message>

Returns a list of all parameters for the given IM.

**public** InstrumentManagerParameter[] getParameters(String sessionID,

String instrumentManagerID, String[] parametersNames)

<wsdl:message name=*"getParametersRequest"*>

<wsdl:part element=*"tns1:getParameters"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getParametersResponse"*>

<wsdl:part element=*"tns1:getParametersResponse"* name=*"parameters"* />

</wsdl:message>

Returns a list of all IM parameters for the given IM.

**public** InstrumentManagerParameter[] getAllParameters(String sessionID,

String instrumentManagerID)

<wsdl:message name=*"getAllParametersRequest"*>

<wsdl:part element=*"tns1:getAllParameters"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getAllParametersResponse"*>

<wsdl:part element=*"tns1:getAllParametersResponse"* name=*"parameters"* />

</wsdl:message>

Sets the specified IM attribute with the value contained in the attribute object.

**public** **void** setAttribute(String sessionID, String instrumentManagerID,

InstrumentManagerAttribute attribute)

<wsdl:message name=*"setAttributeResponse"*>

<wsdl:part element=*"tns1:setAttributeResponse"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"setAttributeRequest"*>

<wsdl:part element=*"tns1:setAttribute"* name=*"parameters"* />

</wsdl:message>

Sets a list of IM attributes with the values contained in the attributes objects.

**public** **void** setAttributes(String sessionID, String instrumentManagerID,

InstrumentManagerAttribute[] attributes)

<wsdl:message name=*"setAttributesRequest"*>

<wsdl:part element=*"tns1:setAttributes"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"setAttributesResponse"*>

<wsdl:part element=*"tns1:setAttributesResponse"* name=*"parameters"* />

</wsdl:message>

Sets the specified IM parameter with the value contained in the parameter object.

**public** **void** setParameter(String sessionID, String instrumentManagerID,

InstrumentManagerParameter parameter)

<wsdl:message name=*"setParameterRequest"*>

<wsdl:part element=*"tns1:setParameter"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"setParameterResponse"*>

<wsdl:part element=*"tns1:setParameterResponse"* name=*"parameters"* />

</wsdl:message>

Sets a list of IM parameters with the values contained in the parameters objects.

**public** **void** setParameters(String sessionID, String instrumentManagerID,

InstrumentManagerParameter[] parameters)

<wsdl:message name=*"setParametersRequest"*>

<wsdl:part element=*"tns1:setParameters"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"setParametersResponse"*>

<wsdl:part element=*"tns1:setParametersResponse"* name=*"parameters"* />

</wsdl:message>

Returns a list of all possible transitions for the specified IM.

**public** InstrumentManagerTransition[] getTransitions(String sessionID,

String instrumentManagerID)

<wsdl:message name=*"getTransitionsRequest"*>

<wsdl:part element=*"tns1:getTransitions"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getTransitionsResponse"*>

<wsdl:part element=*"tns1:getTransitionsResponse"* name=*"parameters"* />

</wsdl:message>

Returns a list of all commands for the specified IM that are available in the current state.

**public** InstrumentManagerCommand[] getCommands(String sessionID,

String instrumentManagerID)

<wsdl:message name=*"getCommandsRequest"*>

<wsdl:part element=*"tns1:getCommands"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"getCommandsResponse"*>

<wsdl:part element=*"tns1:getCommandsResponse"* name=*"parameters"* />

</wsdl:message>

Executes a command on the instrument.

**public** **void** executeCommand(String sessionID, String instrumentManagerID,

InstrumentManagerCommand command)

<wsdl:message name=*"executeCommandRequest"*>

<wsdl:part element=*"tns1:executeCommand"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"executeCommandResponse"*>

<wsdl:part element=*"tns1:executeCommandResponse"* name=*"parameters"* />

</wsdl:message>

Subscribes the user to an attribute. This method allows the user to asynchronously receive notice about data value changes of the chosen attribute using JMS. The user must first subscribe himself to the same JMS provider of this IE instance.

**public** **void** subscribeToAttribute(String sessionID,

String instrumentManagerID, String attributeName)

<wsdl:message name=*"subscribeToAttributeRequest"*>

<wsdl:part element=*"tns1:subscribeToAttribute"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"subscribeToAttributeResponse"*>

<wsdl:part element=*"tns1:subscribeToAttributeResponse"* name=*"parameters"* />

</wsdl:message>

Unsubscribes a user from an attribute. If there are no remaining subscribed users, the IM will stop publishing for the attribute.

**public** **void** unsubscribeFromAttribute(String sessionID,

String instrumentManagerID, String attributeName)

<wsdl:message name=*"unsubscribeFromAttributeRequest"*>

<wsdl:part element=*"tns1:unsubscribeFromAttribute"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"unsubscribeFromAttributeResponse"*>

<wsdl:part element=*"tns1:unsubscribeFromAttributeResponse"*

name=*"parameters"* />

</wsdl:message>

Returns true if the user is attached to the specified IM, false otherwise.

**public** **boolean** isSubscribedToAttribute(String sessionID,

String instrumentManagerID, String attributeName)

<wsdl:message name=*"isSubscribedToAttributeRequest"*>

<wsdl:part element=*"tns1:isSubscribedToAttribute"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"isSubscribedToAttributeResponse"*>

<wsdl:part element=*"tns1:isSubscribedToAttributeResponse"*

name=*"parameters"* />

</wsdl:message>

Subscribes the user to a list of attributes. This method allows the user to asynchronously receive notice about data values changes of the chosen attributes using JMS. The user must first subscribe himself to the same JMS provider of this IE instance.

**public** **void** subscribeToAttributes(String sessionID,

String instrumentManagerID, String[] attributesNames)

<wsdl:message name=*"subscribeToAttributesRequest"*>

<wsdl:part element=*"tns1:subscribeToAttributes"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"subscribeToAttributesResponse"*>

<wsdl:part element=*"tns1:subscribeToAttributesResponse"*

name=*"parameters"* />

</wsdl:message>

Unsubscribes a user from a list of attributes. If there are no remaining subscribed users, the IM will stop publishing for the attributes.

**public** **void** unsubscribeFromAttributes(String sessionID,

String instrumentManagerID, String[] attributesNames)

<wsdl:message name=*"unsubscribeFromAttributesRequest"*>

<wsdl:part element=*"tns1:unsubscribeFromAttributes"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"unsubscribeFromAttributesResponse"*>

<wsdl:part element=*"tns1:unsubscribeFromAttributesResponse"*

name=*"parameters"* />

</wsdl:message>

Locks an Instrument Manager in order to have the exclusive access to the given instrument. Some commands and setAttribute/Parameter might be still executed by other users depending on the Instrument Manager configuration. The getAttribute/Parameter allowed.

**public** **int** lockInstrumentManager(String sessionID, String instrumentManagerID)

<wsdl:message name=*"lockInstrumentManagerRequest"*>

<wsdl:part element=*"tns1:lockInstrumentManager"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"lockInstrumentManagerResponse"*>

<wsdl:part element=*"tns1:lockInstrumentManagerResponse"*

name=*"parameters"* />

</wsdl:message>

Checks if the given Instrument Manager is currently locked.

**public** **boolean** instrumentManagerLocked(String sessionID,

String instrumentManagerID)

<wsdl:message name=*"instrumentManagerLockedRequest"*>

<wsdl:part element=*"tns1:instrumentManagerLocked"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"instrumentManagerLockedResponse"*>

<wsdl:part element=*"tns1:instrumentManagerLockedResponse"*

name=*"parameters"* />

</wsdl:message>

Unlocks an Instrument Manager.

**public** **int** unlockInstrumentManager(String sessionID, String instrumentManagerID)

<wsdl:message name=*"unlockInstrumentManagerRequest"*>

<wsdl:part element=*"tns1:unlockInstrumentManager"* name=*"parameters"* />

</wsdl:message>

<wsdl:message name=*"unlockInstrumentManagerResponse"*>

<wsdl:part element=*"tns1:unlockInstrumentManagerResponse"*

name=*"parameters"* />

</wsdl:message>

**Exception:**

Instrument Element error messages returned to a client.

<wsdl:message name=*"InstrumentElementException"*>

<wsdl:part element=*"tns1:fault"* name=*"fault"* />

</wsdl:message>

## Instrument Manager DTD

<!ELEMENT instrumentManager (stateMachine,attribute\*,parameter\*,command\*)>

<!ELEMENT attribute ((booleanValue|stringValue|shortValue|intValue|longValue|floatValue|doubleValue|calendarValue|vectorValue|enumerationValue),cachingPolicy) >

<!ELEMENT parameter ((booleanValue|stringValue|shortValue|intValue|longValue|floatValue|doubleValue|calendarValue|vectorValue|enumerationValue),cachingPolicy) >

<!ELEMENT commandParameter (booleanValue|stringValue|shortValue|intValue|longValue|floatValue|doubleValue|calendarValue|vectorValue|enumerationValue)>

<!ELEMENT command (commandParameter\*) >

<!ELEMENT cachingPolicy (direct|polling|user)>

<!ELEMENT direct EMPTY>

<!ELEMENT polling EMPTY>

<!ELEMENT user (#PCDATA)>

<!ELEMENT booleanValue EMPTY>

<!ELEMENT calendarValue EMPTY>

<!ELEMENT doubleValue EMPTY>

<!ELEMENT enumerationValue (listValues)>

<!ELEMENT floatValue EMPTY>

<!ELEMENT intValue EMPTY>

<!ELEMENT longValue EMPTY>

<!ELEMENT shortValue EMPTY>

<!ELEMENT stringValue EMPTY>

<!ELEMENT vectorValue ((booleanValue|stringValue|shortValue|intValue|longValue|floatValue|doubleValue|calendarValue|vectorValue|enumerationValue)\*)>

<!ELEMENT listValues ((booleanValue|stringValue|shortValue|intValue|longValue|floatValue|doubleValue|calendarValue|vectorValue|enumerationValue)\*) >

<!ELEMENT stateMachine (status\*)>

<!ELEMENT status EMPTY>

<!ATTLIST status statusName ID #REQUIRED>

<!ATTLIST instrumentManager name CDATA #REQUIRED>

<!ATTLIST instrumentManager implementation CDATA #REQUIRED>

<!ATTLIST instrumentManager id ID #REQUIRED>

<!ATTLIST instrumentManager initialStatus CDATA #REQUIRED>

<!ATTLIST attribute name CDATA #REQUIRED>

<!ATTLIST attribute description CDATA #REQUIRED>

<!ATTLIST attribute enableInStatus CDATA #REQUIRED>

<!ATTLIST attribute unit CDATA *""*>

<!ATTLIST attribute accessibility (READ|WRITE|READWRITE) *"READ"*>

<!ATTLIST attribute implementation CDATA *""*>

<!ATTLIST attribute subscribable (TRUE|FALSE) *"FALSE"*>

<!ATTLIST attribute lockable (TRUE|FALSE) *"FALSE"*>

<!ATTLIST parameter name CDATA #REQUIRED>

<!ATTLIST parameter description CDATA #REQUIRED>

<!ATTLIST parameter enableInStatus CDATA #REQUIRED>

<!ATTLIST parameter unit CDATA *""*>

<!ATTLIST parameter implementation CDATA *""*>

<!ATTLIST parameter lockable (TRUE|FALSE) *"FALSE"*>

<!ATTLIST command name CDATA #REQUIRED>

<!ATTLIST command initialStatus IDREF #REQUIRED>

<!ATTLIST command finalStatus IDREF #REQUIRED>

<!ATTLIST command errorStatus IDREF #REQUIRED>

<!ATTLIST command description CDATA *""*>

<!ATTLIST command implementation CDATA *""*>

<!ATTLIST command lockable (TRUE|FALSE) *"FALSE"*>

<!ATTLIST commandParameter name CDATA #REQUIRED>

<!ATTLIST commandParameter description CDATA #REQUIRED>

<!ATTLIST commandParameter unit CDATA *""*>

<!ATTLIST commandParameter mandatory (TRUE|FALSE) *"TRUE"*>

<!ATTLIST polling time CDATA #REQUIRED>

<!ATTLIST user implementation CDATA #REQUIRED>

<!ATTLIST stringValue value CDATA *""*>

<!ATTLIST enumerationValue validate (TRUE|FALSE) #REQUIRED>

<!ATTLIST listValues defaultValue CDATA #REQUIRED>

<!ATTLIST booleanValue value CDATA *""*>

<!ATTLIST shortValue value CDATA *""*>

<!ATTLIST shortValue min CDATA *""*>

<!ATTLIST shortValue max CDATA *""*>

<!ATTLIST shortValue validate (TRUE|FALSE) *"TRUE"*>

<!ATTLIST intValue value CDATA *""*>

<!ATTLIST intValue min CDATA *""*>

<!ATTLIST intValue max CDATA *""*>

<!ATTLIST intValue validate (TRUE|FALSE) *"TRUE"*>

<!ATTLIST longValue value CDATA *""*>

<!ATTLIST longValue min CDATA *""*>

<!ATTLIST longValue max CDATA *""*>

<!ATTLIST longValue validate (TRUE|FALSE) *"TRUE"*>

<!ATTLIST doubleValue value CDATA *""*>

<!ATTLIST doubleValue min CDATA *""*>

<!ATTLIST doubleValue max CDATA *""*>

<!ATTLIST doubleValue validate (TRUE|FALSE) *"TRUE"*>

<!ATTLIST floatValue value CDATA *""*>

<!ATTLIST floatValue min CDATA *""*>

<!ATTLIST floatValue max CDATA *""*>

<!ATTLIST floatValue validate (TRUE|FALSE) *"TRUE"*>

<!ATTLIST calendarValue value CDATA *""*>

<!ATTLIST calendarValue min CDATA *""*>

<!ATTLIST calendarValue max CDATA *""*>

<!ATTLIST calendarValue validate (TRUE|FALSE) *"TRUE"*>

<!ATTLIST vectorValue maxSize CDATA *""*>

<!ATTLIST vectorValue validate (TRUE|FALSE) *"TRUE"*>

# Security Considerations

Please refer to RFC 3552 [RESCORLA] for guidance on writing a security considerations section. This section is required in all documents, and should not just say “there are no security considerations.” Quoting from the RFC:

“Most people speak of security as if it were a single monolithic property of a protocol or system, however, upon reflection, one realizes that it is clearly not true. Rather, security is a series of related but somewhat independent properties. Not all of these properties are required for every application.

We can loosely divide security goals into those related to protecting communications (COMMUNICATION SECURITY, also known as COMSEC) and those relating to protecting systems (ADMINISTRATIVE SECURITY or SYSTEM SECURITY). Since communications are carried out by systems and access to systems is through communications channels, these goals obviously interlock, but they can also be independently provided.”

# Glossary

Recommended but not required.

# Contributors

Contact information for authors. You can also use this section to recognize contributions by other people who are not listed as authors, but made a useful contribution.

The title page should list the Corresponding Authors (or Editors), who are committed to taking permanent stewardship for this document – receiving communication in the future and otherwise being responsive to its content. Corresponding authors will be sought to process any error reports. The title page should contain at least one and at most three (Corresponding) Author/Editors, unless there are compelling reasons to list more.

Corresponding authors must be indicated as part of the Contributors or Authors section.

Contributors are individuals who assisted with a document’s preparation, and whose contributions are recognized in the document.

The OGF prefers the use of full first names (not initials). Complete contact information for authors must be included. Contributors are listed after authors, and do not need to have complete contact information. The nature of the contribution may be recognized.

John M. Doe

Institution1

Address

Country

Email: jdoe@example.com

Jane Foo-Bar (Corresponding Author)

Open Grid Forum Office

P.O. Box 2326

Joliet, Illinois 60434

USA

Email: jane.foobar@example.net

# Acknowledgments

Include if desired. Contributors to the document may also be listed in the previous section.

# Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights, which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

# Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

# Full Copyright Notice

Copyright (C) Open Grid Forum (2011). Some Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the OGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

# References

Note that only permanent documents should be cited as references. Other items, such as Web pages or working groups, should be cited inline (i.e., see the Open Grid Forum, http://www.ogf.org) or as a footnote. In-text citations can refer to work in progress. To refer to a current working draft, an inline citation might simply read, (the XYZ working group has a draft in progress to address this topic. It may be found via the WG’s Web page at www.example.org). Hyperlinks to transient documents should be avoided (for example, a link to a current draft of a document should not be used, if the document is likely to be replaced in the near future)

References should conform to a standard such as used by IEEE/ACM[[1]](#footnote-1), MLA[[2]](#footnote-2), Chicago[[3]](#footnote-3) or similar. Include an author, year, title, publisher, place of publication. For online materials, also add a URL and an access date. It may be useful, but is not required to separate out “normative references,” as described in [BUSH].

Some sample citations:

[BRADNER] Scott Bradner. Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. The Internet Society. March 1997. http://tools.ietf.org/html/rfc2026

[BUSH] Randy Bush, Thomas Narten. Clarifying when Standards Track Documents may Refer Normatively to Documents at a Lower Level. RFC 3967. The Internet Society. December 2004. http://tools.ietf.org/html/rfc3967

[CATLETT] Charlie Catlett, Cees de Laat, David Martin, Gregory B. Newby, Dane Skow. GFD-C.152: Open Grid Forum Document Process and Requirements. Open Grid Forum. June 2009. http://www.ogf.org/documents/GFD.152.pdf

[RESCORLA] Eric Rescorla, Brian Korver, Internet Architectures Board, Guidelines for Writing RFC Text on Security Considerations. RFC 3552. The Internet Society. July 2003. http://tools.ietf.org/html/rfc3552

1. http://www.computer.org/portal/web/publications/style\_refs [↑](#footnote-ref-1)
2. http://library.concordia.ca/help/howto/mla.php [↑](#footnote-ref-2)
3. http://www.chicagomanualofstyle.org/tools\_citationguide.html [↑](#footnote-ref-3)