Messaging for a Dynamic World

Mobeon Internal
Function Specification

Approved: Magnus Björkman	No: 16/FS-MAS0001

Copyright Mobeon AB All rights reserved	Author: Per Bernhardsson Title: FS – Configuration Manager	Version: A Date: 2006-04-26	1/13

FS – Configuration Manager

Content

History

Version	Date	Adjustments
A	2006-1005	First version (MHATU)

1 Introduction

This document specifies the functionality of the Configuration Manager component.

2 Definitions

2.1 Configuration

Configuration as used in this document contains parameter and value pairs grouped together in a named group and subgroups.

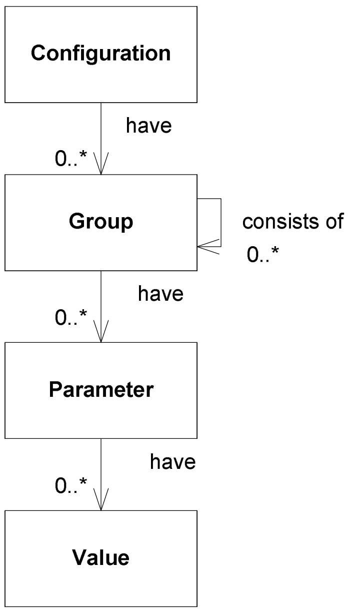


Figure 1: Data model for configuration

The configuration model covers most of the configuration needs by grouping together parameter and value pairs in this manner. The configuration contains many groups that can consist of many parameters with values and subgroups.

As an example a component called callmanager is defined to look for configuration located in a group called "callmanager". This group contains parameter and value pairs specific for the callmanager component.

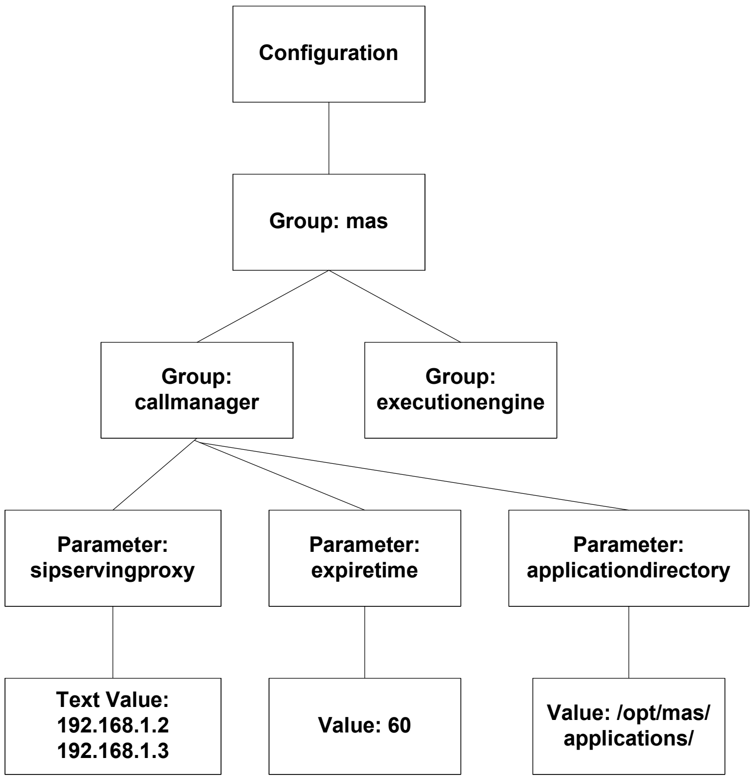

Configuration
Group: mas
Group: callmanager
Group: executionengine
Parameter: sipservingproxy
Parameter: expiretime
Parameter: applicationdirectory
Text Value: 192.168.1.2 192.168.1.3
Value: 60
Value: /opt/mas/ applications/

Figure 2: Example configuration

The callmanager group has three parameters. Two single value parameters, expiretime and applicationdirectory and one text value parameter, sipservingproxy. A text value is a block of text that the Configuration Manager leaves unparsed to the client.

2.2 Configuration storage

The configuration is stored locally on the host in an XML file. The XML file conforms to an XML schema that describes all valid groups and parameters.

3 Function Requirements (Commercial)

No commercial requirements have been identified.

4 Function Specification (Design Related)

4.1 Introduction

The configuration manager component reads and parses XML configuration files. The configuration manager exports three interfaces, one for the manager itself, one interface for managing one configuration, and one for managing configuration groups.

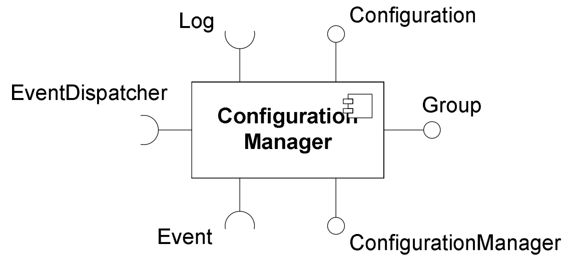


Figure 3: Imported and exported interfaces

4.2 Exported Interfaces

The configuration management package exports three interfaces: ConfigurationManager, Configuration and Group. These interfaces are described below.

4.2.1 Configuration management interface (ConfigurationManager)

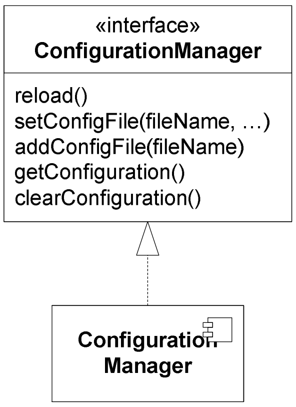


Figure 4: The ConfigurationManager interface

This interface is used to load and distribute instances of the Configuration interface.

4.2.1.1 Functions

4.2.1.1.1 reload

reload()

This method tells the configuration manager to reload the current configuration from disk. If a backup file is used during reload true is returned.

4.2.1.1.2 setConfigFile

setConfigFile(filename, ...)

This method replaces all previous files in the configuration set with a new set. Multiple files can be added at the same time.

4.2.1.1.3 addConfigFile

addConfigFile(filename)

This method adds a new file the name of the configuration set. Multiple files can be added.

4.2.1.1.4 getConfiguration

getConfiguration()

This method returns the latest configuration read from disk.

4.2.1.1.5 clearConfiguration

clearConfiguration()

This method clears the current configuration and resets the list of configuration files.

4.2.2 Configuration interface (Configuration)

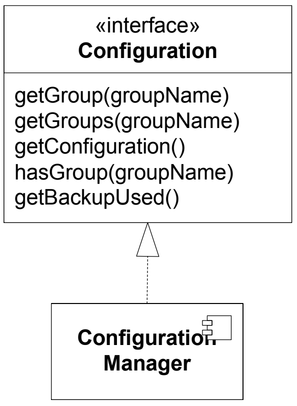


Figure 5: The Configuration interface

This interface is used for locating groups of configuration, i.e. groups of parameter and value pairs. When using this interface the normal user will not have to deal with specific configuration files.

4.2.2.1 Functions

4.2.2.1.1 getGroup

getGroup(groupName)

This function locates the group of parameter and value pairs that have the name "**groupName".** The group returned is accessed through the *Group* interface. If multiple groups are found an error is returned.

4.2.2.1.2 getGroups

getGroups(groupName)

This function locates one or more groups based on the provided group name. The groups returned are accessed through the *Group* interface.

4.2.2.1.3 getConfiguration

getConfiguration()

This function is used after a configuration update to retrieve the new instance of the Configuration interface.

4.2.2.1.4 hasGroup

hasGroup(groupName)

The function tries to find the named group and returns true if it was found.

4.2.2.1.5 getBackupUsed

getBackupUsed()

This function returns true if at least one backup file was used while loading the configuration from disk.

4.2.2.2 Parameter Type Description

groupName – a string uniquely identifying a group or a set of groups in the configuration hierarchy. Each level in the hierarchy is described by the name of the group and is separated by a dot '.'. Ex. "mas.callmanager". When accessing groups from the Configuration interface all group names must be absolute from the root of the hierarchy.

4.2.3 Group Interface (Group)

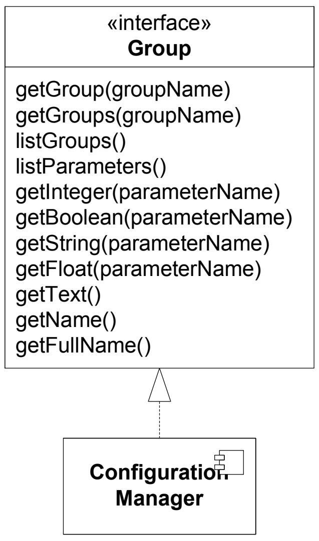


Figure 6: The Group interface

This is the interface to a group and its subgroups and parameters. The interface makes it possible to locate parameters and subgroups within this group.

4.2.3.1 Functions

4.2.3.1.1 getGroup

getGroup(groupName)

This function locates the subgroup in this group with the given name. The group returned is accessed through the *Group* interface. If multiple groups are found an error is returned.

4.2.3.1.2 getGroups

getGroups(groupName)

This function locates one or more groups based on the provided name. The groups returned are accessed through the *Group* interface.

4.2.3.1.3 listGroups

listGroups()

This functions returns for a list of names for all groups one level down from the current group.

4.2.3.1.4 listParameters

listParameters()

This function returns a list of names for all parameters in the current group.

4.2.3.1.5 getInteger

getInteger(name, [defaultValue])

This function returns an integer value associated with the parameter. If the parameter is missing or wrong type in the configuration, an error is returned unless a default value has been supplied. If the default value is returned a log is written.

4.2.3.1.6 getBoolean

getBoolean(name, [defaultValue])

This function returns a boolean value associated with the parameter. If the parameter is missing or wrong type in the configuration, an error is returned unless a default value has been supplied. If the default value is returned a log is written.

4.2.3.1.7 getString

getString(name, [defaultValue])

This function returns a string value associated with the parameter. These strings may be anything that can be stored as a string. If the parameter is missing from the configuration an error is returned unless a default value has been supplied. If the default value is returned a log is written.

4.2.3.1.8 getFloat

getFloat(name, [defaultValue])

This function returns a float value associated with the parameter. If the parameter is missing or wrong type in the configuration, an error is returned unless a default value has been supplied. If the default value is returned a log is written.

4.2.3.1.9 getText

getText()

This function returns the text value associated with the current group. Only one text value is supported for each group. It is not possible to supply a default value for this method.

4.2.3.1.10 getName

getName()

This function returns the name of the current group.

4.2.3.1.11 getFullName

getFullName()

This function returns the full name of the current group.

4.2.3.2 Parameter Type Description

groupName – a string uniquely identifying a group or a set of groups in the configuration hierarchy. When accessing subgroups from the Group interface all group names must be relative from the current group of the hierarchy.

4.2.4 Verification tool

The Configuration Management provides an external verification tool that can be used to verify a configuration file prior to loading it into the application.

4.3 Imported Interfaces

The following interfaces are imported from external sources.

- Log – To write logs to file.
- EventDispatcher – To send configuration update events.
- Event – Required by EventDispatcher.

4.4 Events

4.4.1 Generated

4.4.1.1 Configuration has changed

This event is sent when the configuration manager has new or changed configuration. Every component registered to receive this event must reread its configuration.

If the read configuration is incorrect for some reason the configuration file should be ignored and the previous configuration should be used. If a syntax error occurs during startup of the component the component should shut down.

In the event of a corrupt configuration file the configuration manager must log this fact adding as much information as possible to identify the problem.

4.5 Consumed events

No consumed events.

4.6 Function Specification

4.6.1 Retrieving the configuration for the first time

This function is initiated by the application framework.

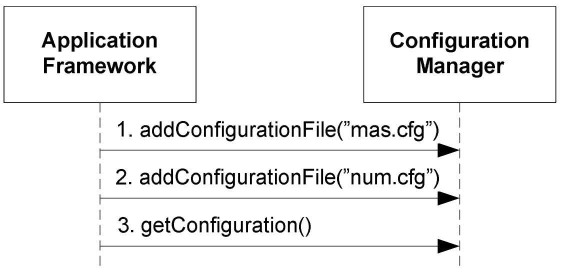

Application Framework
Configuration Manager
1. addConfigurationFile(”mas.cfg”)
2. addConfigurationFile(”num.cfg”)
3. getConfiguration()

Figure 7: Retireveing the configuration

This scenario is used when the application framework needs to retrieve the configuration the first time.

1. A configuration file is added to the Configuration Manager.
2. A second configuration file is added to the Configuration Manager.
3. A configuration is retrieved from the Configuration Manager.

4.6.2 Configuration has changed

This function is initiated by the operator.

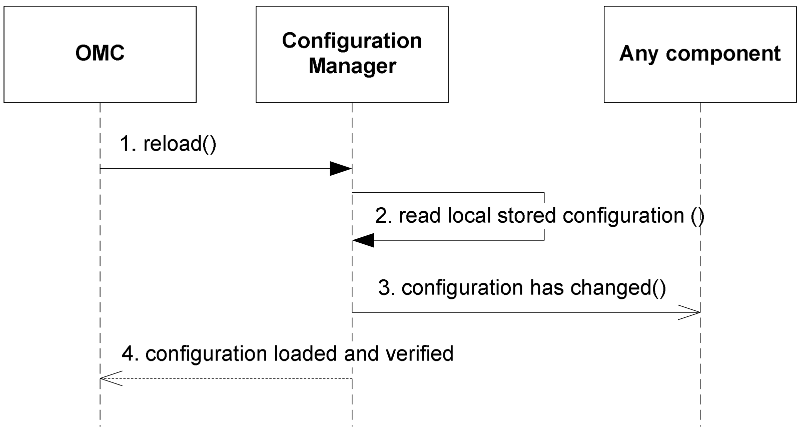

OMC
Configuration Manager
Any component
1. reload()
2. read local stored configuration ()
3. configuration has changed()
4. configuration loaded and verified

Figure 8: Configuration has changed

This scenario is used when the local configuration has been changed by the operator and the operator has initiated the reload configuration procedure.

1. The OMC uses the method reload to tell the Configuration Manager to reload the configuration.
2. The configuration manager retrieves the configuration from local storage.
3. The configuration manager generates a “configuration has changed” event to all components registered to receive events.
4. The OMC is notified that the configuration was either loaded and verified ok, or if it failed for some reason.

4.6.3 Usage of the configuration

There are a lot of ways how the configuration is used and this scenario is just one of many possible uses. It is based on the example shown previous in this document, see Figure 2: Example configuration.

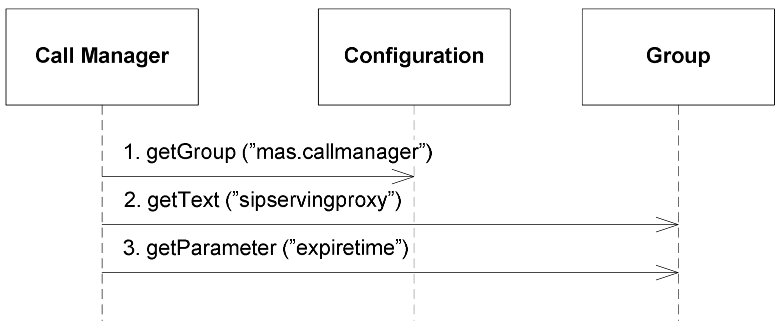

Call Manager
Configuration
Group
1. getGroup (”mas.callmanager”)
2. getText (”sipservingproxy”)
3. getParameter (”expiretime”)

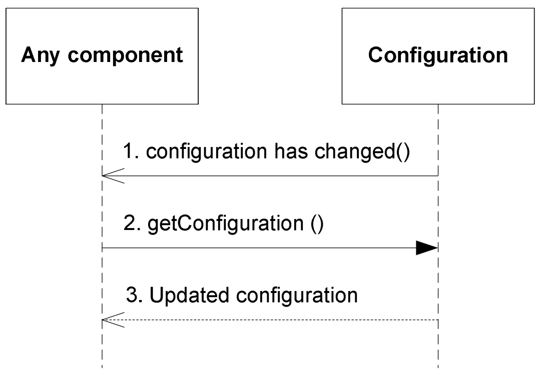
Figure 9: Using the configuration manager

In this scenario the Call Manager Component retrieves its configuration from the configuration manager.

1. Call Manager needs the group called “mas.callmanager” because all parameters needed by the Call Manager is located below that group.
2. Call Manager then uses the retrieved group to query for the “sipservingproxy” parameter.
3. Call Manager then uses the retrieved group to query for the “expiretime” parameter.

4.6.4 The configuration has been updated

When the configuration is updated on disk the Configuration Manager informs all components that are interested.



In this scenario a component retrieves an updated configuration interface when informed of a change.

1. The component gets an event informing it of a configuration update.
2. The component calls its current configuration instance to get an updated configuration instance.

3. The old configuration instance returns an updated configuration which the component can use to read the updated configuration. The old instance is discarded.

4.7 Configuration tool

The configuration tool is used to change, retrieve and validate values from XML-configuration files.

The configuration tool is a java based tool.

Syntax:
java com.mobeon.masp.cfgtool.CfgTool <action>

Actions

-g param [-g param] [-q] [-v] <input file>;

Print the value of one or more parameters.

-s param=value [-s param=value] [-v] <input file> [output file]

Set one or more parameters to the specified value. The result is stored in <output file> if it has been specified. If not <input file> is overwritten.

-m [-v] <old file> <new file> [output file]

Merge <old file> into <new file>. The result is store in <output file> if it has been specified. If not <old file> is overwritten.

-v <input file>

Validate the XML file against its schema.
If validated ok, nothing is returned,
else an error message is returned.

-l <input file>;

List all parameters in the configuration file.

-h

Prints help message

-v

Validate the XML file against its schema.
If validated ok, nothing is returned,
else an error message is returned.

-q

"Quiet mode", only print the value.

The -g, -s, -m, and -l parameters are mutually exclusive and can't be used simultaneously.

Ex.

java com.mobeon.masp.cfgtool.CfgTool -s executionengine.hostname=MAS /apps/mas/cfg/mas.xml

5 Terminology

OMC	A component that initiates the reload function. For example an Operate and Maintain Component.

6 References

1. FS Configuration
14/155 17-1/HDB 101 02