



EMC® Watch4net®

Version 6.4u2

Administration Tool Administration Guide

P/N: pending

Rev : 41247

EMC²

Copyright ©2013 - 2014 EMC Corporation. All rights reserved. Published in the USA.

Published October, 2014

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC², EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to EMC Online Support (<https://support.emc.com>).

Contents

Preface	iii
Intended Audience	iii
Prerequisites	iii
Where to get help	v
1 Product Overview	1
2 Configuration	3
3 Administration-Tool Definition	5
1 User	5
1.1 listUser	5
1.2 getUser	6
1.3 createUser	7
1.4 updateUser	9
1.5 deleteUser	10
2 Role	12
2.1 listRole	12
2.2 getRole	13
2.3 createRole	14
2.4 updateRole	16
2.5 deleteRole	18
3 Profile	20
3.1 listProfile	20
3.2 getProfile	21
3.3 createProfile	22
3.4 updateProfile	23
3.5 deleteProfile	24
4 Report pack	26
4.1 listReportPack	26

4.2	getReportPack	27
4.3	createReportPack	28
4.4	updateReportPack	29
4.5	deleteReportPack	30
5	User and role relation	31
5.1	linkUserRole	31
5.2	linkUserRole	32
5.3	getUserRoles	34
5.4	getRoleUsers	35
4	Appendix A	37
1	Configuration Example	37

Preface

The purpose of this guide is to provide information about the Administration Tool for APG. It contains all the necessary information to install, configure and use it.

Intended Audience

This guide is intended to be read by any *APG* administrator who will install, configure and use the Administration Tool.

Prerequisites

This guide assumes that you have an *APG* installation ready with the webservice- gateway module installed.

Where to get help

EMC support, product, and licensing information can be obtained as follows:

PRODUCT INFORMATION - For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

<https://support.emc.com>

TECHNICAL SUPPORT - Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

1

Product Overview

The Administration Tool provides a way, via a web-service, to create, update, get, list and delete users, roles, profiles and report packs. This service definition is accessible through the url of the webservice-gateway, usually at <https://<apg.server>:48443/Tools/Administration-Tool/<Instance>/?wsdl>.

2

Configuration

The Administration Tool needs a configuration file to access *APG* master database. This file is located in the <APG Root>/Tools/Administration-Tool/<Instance>/conf directory. In case this file do not exist, it must be created in this folder.

Global structure of the configuration file Here is the global structure of the configuration file:

- <master-database>: the connection to the master database.

See [Appendix A](#) page [37](#) for an example.

Administration-Tool Definition

1 User

1.1 listUser

`user* listUser(filter?)` throws `MasterAccessorServiceException`

List all the users that matches the list of search filters. Each of the search filters are combined together in an "and" manner.

Parameters:

- *filter?*: the filters of the search. (optional)

Returns:

- *user**: a list of matching users. (zero or more)

Throws:

- *MasterAccessorServiceException*:
 - if there is an error while getting the list of users from the database.
 - if a field in the filter does not exists or is not accessible.
 - if the field "id" is provided and is not a number.

Request Samples:

```
<listUser>
  <!--Zero or more repetitions:-->
  <filter key="?"?</filter>
</listUser>
```

Response Samples:

```
<listUserResponse>
  <user name="admin" id="1"/>
  <user name="dduclos" id="5"/>
  <user name="emc" id="6"/>
  <user name="executive" id="2"/>
  <user name="operator" id="3"/>
  <user name="user" id="4"/>
</listUserResponse>
```

1.2 getUser

user **getUser**(user) throws MasterAccessorServiceException

Returns the user matching the key.

Parameters:

- *user*: the key (id and/or name) used to search the user.

Returns:

- *user*: the user matching the key or NULL if it does not exists.

Throws:

- *MasterAccessorServiceException*:
 - if the key is NULL.
 - if there is an error while getting the user from the database.

Request Samples:

```
<getUser>
  <user id="?" name="?">
</getUser>
```

Response Samples:

```
<getUserResponse>
  <user username="admin" id="1">
    <user-status>admin</user-status>
    <disabled>false</disabled>
    <custom-report-allowed>true</custom-report-allowed>
  </user>
</getUserResponse>
```

1.3 createUser

user* createUser(*user*) throws MasterAccessorServiceException

Adds the user in the master database.

Parameters:

- *user**: the user to be added.

Returns:

- *user*: the key (id and/or name) of the added user (one).

Throws:

- *MasterAccessorServiceException*:
 - if the user is NULL.
 - if there is an error while saving the user in the database.
 - if the load of the selected profile id does not work.
 - if there is a validation error.

Request Samples:

```
<createUser>
  <user id="?" username="?">
    <!--Optional:-->
    <password?></password>
    <!--Optional:-->
    <title?></title>
    <!--Optional:-->
    <first-name?></first-name>
    <!--Optional:-->
    <last-name?></last-name>
    <!--Optional:-->
    <email?></email>
    <!--Optional:-->
    <filter?></filter>
    <!--Optional:-->
    <user-status?></user-status>
    <!--Optional:-->
    <disabled>false</disabled>
    <!--Optional:-->
    <profile-id?></profile-id>
    <!--Optional:-->
    <custom-report-allowed?></custom-report-allowed>
  </user>
</createUser>
```

Response Samples:

```
<createUserResponse>
  <user name="test7" id="16"/>
</createUserResponse>
```

1.4 updateUser

user **updateUser**(*user*) throws *MasterAccessorServiceException*

Updates the user.

Parameters:

- *user*: the user to be updated. Only the id will be used to search the user.

Returns:

- *user*: the key (id and/or name) of the updated user.

Throws:

- *MasterAccessorServiceException*:
 - if there is an error while getting the user from the database.
 - if the user is NULL.
 - if the user does not have an id.
 - if the user does not exists.
 - if there is an error while saving the user in the database.

Request Samples:

```
<updateUser>
  <user id="?" username="?">
    <!--Optional:-->
    <password>?</password>
    <!--Optional:-->
    <title>?</title>
    <!--Optional:-->
    <first-name>?</first-name>
    <!--Optional:-->
    <last-name>?</last-name>
    <!--Optional:-->
    <email>?</email>
    <!--Optional:-->
    <filter>?</filter>
    <!--Optional:-->
    <user-status>?</user-status>
    <!--Optional:-->
    <disabled>?</disabled>
    <!--Optional:-->
    <profile-id>?</profile-id>
    <!--Optional:-->
    <custom-report-allowed>?</custom-report-allowed>
  </user>
</updateUser>
```

Response Samples:

```
<updateUserResponse>
  <user name="test7" id="16"/>
</updateUserResponse>
```

1.5 deleteUser

user **deleteUser**(user) throws MasterAccessorServiceException

Deletes the user matching the key.

Parameters:

- *user*: the key (id and/or name) of the user to be deleted.

Returns:

- *user*: the key (id and/or name) of the deleted user.

Throws:

- *MasterAccessorServiceException*:
 - if the key is NULL.
 - if there is an error while getting the user from the database.
 - if the user does not exists.
 - if there is an error while deleting the user from the database.

Request Samples:

```
<deleteUser>
  <user id="?" name="?" />
</deleteUser>
```

Response Samples:

```
<deleteUserResponse>
  <user name="test7" id="16" />
</deleteUserResponse>
```

2 Role

2.1 listRole

`role* listRole(filter?)` throws `MasterAccessorServiceException`

List all the roles that matches the list of search filters. Each of the search filters are combined together in an "and" manner.

Parameters:

- *filter?*: the filters of the search. (optional)

Returns:

- *role**: a list of matching roles or an empty list if there is no match.

Throws:

- *MasterAccessorServiceException*:
 - if there is an error while getting the list of roles from the database.
 - if a field in the filter does not exists or is not accessible.
 - if the field "id" is provided and is not a number.

Request Samples:

```
<listRole>
  <!--Zero or more repetitions:-->
  <filter key="?">?</filter>
</listRole>
```

Response Samples:

```
<listRoleResponse>
  <role name="Executives_View" id="5"/>
  <role name="Full_Control_Users" id="4"/>
  <role name="Operators_View" id="6"/>
  <role name="Read-Only_&Real-Time_Grapher_Users" id="2"/>
  <role name="Read-Only_Users" id="1"/>
  <role name="Read-Write_Users" id="3"/>
  <role name="Users_View" id="7"/>
</listRoleResponse>
```

2.2 getRole

role **getRole**(*role*) throws `MasterAccessorServiceException`

Returns the role matching the key.

Parameters:

- *role*: the key (id and/or name) used to search the role.

Returns:

- *role*: the role matching the key or NULL if it does not exists.

Throws:

- *MasterAccessorServiceException*:
 - if the key is NULL.
 - if there is an error while getting the role from the database.
 - if there is more than one role matching the key.

Request Samples:

```
<getRole>
  <role id="?" name="?" />
</getRole>
```

Response Samples:

```
<getRoleResponse>
  <role name="Read-Only_Users" id="1">
    <description>This role allows its users to access all tree templates in read-only mode
  .</description>
    <disabled>false</disabled>
    <filter/>
    <template-rights>
      <template-right>
        <access-type>read</access-type>
      </template-right>
    </template-rights>
    <module-rights/>
    <restriction-rights>
      <restriction-right feature="report.refiner">true</restriction-right>
    </restriction-rights>
    <admin-rights/>
  </role>
</getRoleResponse>
```

2.3 createRole

role **createRole**(*role*) throws MasterAccessorServiceException

Adds the role in the master database.

Parameters:

- *role*: the role to be added.

Returns:

- *role*: the key (id and/or name) of the added role (one).

Throws:

- *MasterAccessorServiceException*:
 - if the role is NULL.
 - if there is an error while saving the role in the database.
 - if there is a validation error.

Request Samples:

```
<createRole>
  <!--Optional:-->
  <role id="?" name="?">
    <!--Optional:-->
    <description?></description>
    <!--Optional:-->
    <disabled?></disabled>
    <!--Optional:-->
    <filter?></filter>
    <!--Optional:-->
    <parent-role-id?></parent-role-id>
    <!--Optional:-->
    <template-rights>
      <!--Zero or more repetitions:-->
      <template-right>
        <report-pack-id?></report-pack-id>
        <template-id?></template-id>
        <!--Optional:-->
        <subscription>
          <!--Zero or more repetitions:-->
          <node name="?"></node>
        </subscription>
        <access-type?></access-type>
      </template-right>
    </template-rights>
  <!--Optional:-->
</createRole>
```

```
<module-rights>
  <!--Zero or more repetitions:-->
  <module-right name="?"?</module-right>
</module-rights>
<!--Optional:-->
<restriction-rights>
  <!--Zero or more repetitions:-->
  <restriction-right feature="?"?</restriction-right>
</restriction-rights>
<!--Optional:-->
<admin-rights>
  <!--Zero or more repetitions:-->
  <admin-right>
    <name>?</name>
    <ids>
      <!--1 or more repetitions:-->
      <id>?</id>
    </ids>
  </admin-right>
</admin-rights>
</role>
</createRole>
```

Response Samples:

```
<createRoleResponse>
  <role name="Test_Role_3" id="10"/>
</createRoleResponse>
```

2.4 updateRole

role **updateRole**(*role*) throws MasterAccessorServiceException

Updates the role.

Parameters:

- *role*: the role to be updated. Only the id will be used to search the role.

Returns:

- *role*: the key (id and/or name) of the updated role.

Throws:

- *MasterAccessorServiceException*:
 - if there is an error while getting the role from the database.
 - if the role is NULL.
 - if the role does not have an id.
 - if the role does not exists.
 - if there is an error while saving the role in the database.

Request Samples:

```
<updateRole>
  <role id="?" name="?">
    <!--Optional:-->
    <description?></description>
    <!--Optional:-->
    <disabled?></disabled>
    <!--Optional:-->
    <filter?></filter>
    <!--Optional:-->
    <parent-role-id?></parent-role-id>
    <!--Optional:-->
    <template-rights>
      <!--Zero or more repetitions:-->
      <template-right>
        <report-pack-id?></report-pack-id>
        <template-id?></template-id>
        <!--Optional:-->
        <subscription>
          <!--Zero or more repetitions:-->
          <node name="?"></node>
        </subscription>
        <access-type?></access-type>
      </template-right>
    </template-rights>
  </role>
</updateRole>
```

```
</template-rights>
<!--Optional:-->
<module-rights>
  <!--Zero or more repetitions:-->
  <module-right name="?"?</module-right>
</module-rights>
<!--Optional:-->
<restriction-rights>
  <!--Zero or more repetitions:-->
  <restriction-right feature="?"?</restriction-right>
</restriction-rights>
<!--Optional:-->
<admin-rights>
  <!--Zero or more repetitions:-->
  <admin-right>
    <name?</name>
    <ids>
      <!--1 or more repetitions:-->
      <id?</id>
    </ids>
  </admin-right>
</admin-rights>
</role>
</updateRole>
```

Response Samples:

```
<updateRoleResponse>
  <role name="Test_Role_3" id="10"/>
</updateRoleResponse>
```

2.5 deleteRole

role **deleteRole**(*role*) throws MasterAccessorServiceException

Deletes the role matching the key.

Parameters:

- *role*: the key (id and/or name) of the role to be deleted.

Returns:

- *role*: the key (id and/or name) of the deleted role.

Throws:

- *MasterAccessorServiceException*:
 - if the key is NULL.
 - if there is an error while getting the role from the database.
 - if the role does not exists.
 - if there is an error while deleting the role from the database.
 - if there is more than one role matching the key.

Request Samples:

```
<deleteRole>
  <role id="?" name="?" />
</deleteRole>
```

Response Samples:

```
<deleteRoleResponse>
  <role name="Test_Role_3" id="10" />
</deleteRoleResponse>
```

3 Profile

3.1 listProfile

`profile* listProfile(filter?)` throws `MasterAccessorServiceException`

List all the profiles that matches the list of search filters. Each of the search filters are combined together in an "and" manner.

Parameters:

- *filter?*: the filters of the search. (optional)

Returns:

- *profile**: a list of matching profiles or an empty list if there is no match.

Throws:

- *MasterAccessorServiceException*:
 - if there is an error while getting the list of profiles from the database.
 - if a field in the filter does not exists or is not accessible.
 - if the field "id" is provided and is not a number.

Request Samples:

```
<listProfile>
  <!--Zero or more repetitions:-->
  <filter key="?">?</filter>
</listProfile>
```

Response Samples:

```
<listProfileResponse>
  <profile name="Default_Profile" id="0"/>
  <profile name="Test_Profile" id="1"/>
  <profile name="Test_Profile_2" id="2"/>
</listProfileResponse>
```

3.2 getProfile

profile **getProfile**(*profile*) throws *MasterAccessorServiceException*

Returns the role matching the key.

Parameters:

- *profile*: the key (id and/or name) used to search the profile.

Returns:

- *profile*: the profile matching the key or NULL if it does not exists.

Throws:

- *MasterAccessorServiceException*:
 - if the key is NULL.
 - if there is an error while getting the profile from the database.
 - if there is more than one profile matching the key.

Request Samples:

```
<getProfile>
  <profile id="?" name="?" />
</getProfile>
```

Response Samples:

```
<getProfileResponse>
  <profile name="Default_Profile" id="0">
    <description/>
    <settings>
      <setting name="refresh-rate">5</setting>
      <setting name="home-report">lookup/{W4N_DS_INFRA}</setting>
    </settings>
  </profile>
</getProfileResponse>
```

3.3 createProfile

profile **createProfile**(*profile*) throws MasterAccessorServiceException

Adds the profile in the master database.

Parameters:

- *profile*: the profile to be added.

Returns:

- *profile*: the key (id and/or name) of the added profile (one).

Throws:

- *MasterAccessorServiceException*:
 - if the profile is NULL.
 - if there is an error while saving the profile in the database.
 - if there is a validation error.

Request Samples:

```
<createProfile>
  <profile id="?" name="?">
    <!--Optional:-->
    <description?></description>
    <!--Optional:-->
    <settings>
      <!--Zero or more repetitions:-->
      <setting name="?"></setting>
    </settings>
  </profile>
</createProfile>
```

Response Samples:

```
<createProfileResponse>
  <profile name="Test_Profile_4" id="5"/>
</createProfileResponse>
```

3.4 updateProfile

`profile` **updateProfile**(*profile*) throws `MasterAccessorServiceException`

Updates the profile.

Parameters:

- *profile*: the profile to be updated. Only the id will be used to search the profile.

Returns:

- *profile*: the key (id and/or name) of the updated profile.

Throws:

- *MasterAccessorServiceException*:
 - if there is an error while getting the profile from the database.
 - if the profile is NULL.
 - if the profile does not have an id.
 - if the profile does not exists.
 - if there is an error while saving the profile in the database.

Request Samples:

```
<updateProfile>
  <profile id="?" name="?">
    <!--Optional:-->
    <description>?</description>
    <!--Optional:-->
    <settings>
      <!--Zero or more repetitions:-->
      <setting name="?">?</setting>
    </settings>
  </profile>
</updateProfile>
```

Response Samples:

```
<updateProfileResponse>
  <profile name="Test_Profile_4" id="5"/>
</updateProfileResponse>
```

3.5 deleteProfile

profile **deleteProfile**(*profile*) throws *MasterAccessorServiceException*

Deletes the profile matching the key.

Parameters:

- *profile*: the key (id and/or name) of the profile to be deleted.

Returns:

- *profile*: the key (id and/or name) of the deleted profile.

Throws:

- *MasterAccessorServiceException*:
 - if the key is NULL.
 - if there is an error while getting the profile from the database.
 - if the profile does not exists.
 - if there is an error while deleting the profile from the database.
 - if there is more than one profile matching the key.

Request Samples:

```
<deleteProfile>
  <profile id="?" name="?" />
</deleteProfile>
```

Response Samples:

```
<deleteProfileResponse>
  <profile name="Test_Profile_4" id="5" />
</deleteProfileResponse>
```

4 Report pack

4.1 listReportPack

report-pack* **listReportPack**(*filter?*) throws `MasterAccessorServiceException`

List all the report packs that matches the list of search filters. Each of the search filters are combined together in an "and" manner.

Parameters:

- *filter?*: the filters of the search. (optional)

Returns:

- *report-pack**: a list of matching report packs or an empty list if there is no match.

Throws:

- *MasterAccessorServiceException*:
 - if there is an error while getting the list of report packs from the database.
 - if a field in the filter does not exists or is not accessible.
 - if the field "id" is provided and is not a number.

Request Samples:

```
<listReportPack>
  <!--Zero or more repetitions:-->
  <filter key="?">?</filter>
</listReportPack>
```

Response Samples:

```
<listReportPackResponse>
  <report-pack name="Default_ReportPack" id="6"/>
  <report-pack name="ReportPack_for_APG_Health" id="1"/>
  <report-pack name="ReportPack_for_EMCIonix_ITOps" id="3"/>
  <report-pack name="ReportPack_for_EMCSmarts_Events" id="4"/>
  <report-pack name="ReportPack_for_Smarts_Health" id="5"/>
</listReportPackResponse>
```

4.2 getReportPack

report-pack **getReportPack**(*report-pack*) throws *MasterAccessorServiceException*

Returns the report pack matching the key.

Parameters:

- *report-pack*: the key (id and/or name) used to search the report-pack.

Returns:

- *report-pack*: the report-pack matching the key (zero or one).

Throws:

- *MasterAccessorServiceException*:
 - if the key is NULL.
 - if there is an error while getting the report pack from the database.
 - if there is more than one report pack matching the key.

Request Samples:

```
<getReportPack>
  <report-pack id="?" name="?" />
</getReportPack>
```

Response Samples:

```
<getReportPackResponse>
  <report-pack name="ReportPack_for_APG_Health" id="1">
    <file>
      <xop:Include href="cid:7bc8adb8-6b9b-4a9f-b8ba-3740def3c93a@example.jaxws.sun.com"
xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
    </file>
  </report-pack>
</getReportPackResponse>
```

4.3 createReportPack

report-pack **createReportPack**(*report-pack*) throws *MasterAccessorServiceException*

Adds the report pack in the master database.

Parameters:

- *report-pack*: the report pack to be added.

Returns:

- *report-pack*: the key (id and/or name) of the added report pack (one).

Throws:

- *MasterAccessorServiceException*:
 - if the report pack is NULL.
 - if there is an error while saving the report pack in the database.
 - if there is a validation error.

Request Samples:

```
<createReportPack>
  <report-pack id="?" name="?">
    <!--Optional:-->
    <file>cid:~/file>
  </report-pack>
</createReportPack>
```

Response Samples:

```
<createReportPackResponse>
  <report-pack name="ReportPack_for_Network_Traffic_Flows" id="10"/>
</createReportPackResponse>
```

4.4 updateReportPack

report-pack **updateReportPack**(*report-pack*) throws `MasterAccessorServiceException`

Updates the report pack. If an ARP file is provided, the templates of the report pack, matched by their names, will be updated.

Parameters:

- *report-pack*: the report pack to be updated. Only the id is used to search the report pack.

Returns:

- *report-pack*: the key (id and/or name) of the updated report pack.

Throws:

- *MasterAccessorServiceException*:
 - if there is an error while getting the report pack from the database.

- if the report pack is NULL.
- if the report pack does not have an id.
- if the report pack does not exists.
- if there is an error while saving the report pack in the database.

Request Samples:

```
<updateReportPack>
  <report-pack id="?" name="?">
    <!--Optional:-->
    <file>cid:~/file>
  </report-pack>
</updateReportPack>
```

Response Samples:

```
<updateReportPackResponse>
  <report-pack name="ReportPack_for_Network_Traffic_Flows" id="11"/>
</updateReportPackResponse>
```

4.5 deleteReportPack

report-pack **deleteReportPack**(*report-pack*) throws `MasterAccessorServiceException`

Deletes the report pack matching the key.

Parameters:

- *report-pack*: the key (id and/or name) of the report-pack to be deleted.

Returns:

- *report-pack*: the key (id and/or name) of the deleted report pack.

Throws:

- *MasterAccessorServiceException*:
 - if the key is NULL.
 - if there is an error while getting the report pack from the database.
 - if the report pack does not exists.
 - if there is an error while deleting the report pack from the database.
 - if there is more than one report pack matching the key.

Request Samples:

```
<deleteReportPack>
  <report-pack id="?" name="?" />
</deleteReportPack>
```

Response Samples:

```
<deleteReportPackResponse>
  <report-pack name="ReportPack_for_Network_Traffic_Flows" id="11" />
</deleteReportPackResponse>
```

5 User and role relation

5.1 linkUserRole

UserRoleRelationKey **linkUserRole**(*user*, *role*) throws MasterAccessorServiceException

Links a user and a role.

Parameters:

- *user*: the key (id and/or name) of the user.
- *role*: the key (id and/or name) of the role.

Returns:

- *UserRoleRelationKey*: a key (id and/or name) containing the keys (ids and/or names) of the linked user and roles.

Throws:

- *MasterAccessorServiceException*:
 - if the user or the role key is NULL.
 - if the user or the role does not exists.
 - if there is an error while linking the user and the role.

Request Samples:

```
<linkUserRole>
  <user id="?" name="?" />
  <role id="?" name="?" />
</linkUserRole>
```

Response Samples:

```
<linkUserRoleResponse>
  <user name="test6" id="15"/>
  <role name="Test_Role_2" id="9"/>
</linkUserRoleResponse>
```

5.2 linkUserRole

UserRoleRelationKey **unlinkUserRole**(*user, role*) throws MasterAccessorServiceException

Unlinks a user and a role.

Parameters:

- *user*: the key (id and/or name) of the user.
- *role*: the key (id and/or name) of the role.

Returns:

- *UserRoleRelationKey*: a key (id and/or name) containing the keys (ids and/or names) of the unlinked user and roles.

Throws:

- *MasterAccessorServiceException*:
 - if the user or the role key is NULL.
 - if the user or the role does not exists.
 - if there is an error while unlinking the user and the role.

Request Samples:

```
<unlinkUserRole>
  <user id="?" name="?" />
  <role id="?" name="?" />
</unlinkUserRole>
```

Response Samples:

```
<unlinkUserRoleResponse>
  <user name="test6" id="15" />
  <role name="Test_Role_2" id="9" />
</unlinkUserRoleResponse>
```

5.3 getUserRoles

role* **getUserRoles**(*user*) throws MasterAccessorServiceException

Lists all the roles linked to a user.

Parameters:

- *user*: the key (id and/or name) of the user.

Returns:

- *role**: a list of ids of the roles linked to the user (zero or more).

Throws:

- *MasterAccessorServiceException*:
 - if the user key is NULL.
 - if the user does not exists.
 - if there is an error while retrieving the user and its roles.

Request Samples:

```
<getUserRoles>
  <user id="?" name="?" />
</getUserRoles>
```

Response Samples:

```
<getUserRolesResponse>
  <role id="5" />
</getUserRolesResponse>
```

5.4 getRoleUsers

`user* getRoleUsers(role)` throws `MasterAccessorServiceException`

Lists all the users linked to a role.

Parameters:

- *role*: the key (id and/or name) of the role.

Returns:

- *user**: a list of ids of the users linked to the role (zero or more).

Throws:

- *MasterAccessorServiceException*:
 - if the role key is NULL.
 - if the role does not exists.
 - if there is an error while retrieving the role and its users.

Request Samples:

```
<getRoleUsers>
  <role id="?" name="?" />
</getRoleUsers>
```

Response Samples:

```
<getRoleUsersResponse>
  <user id="14"/>
  <user id="15"/>
</getRoleUsersResponse>
```


4

Appendix A

1 Configuration Example

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/administration-tool" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.watch4net.com/administration-tool master-
accessor-service-conf.xsd">
  <master-database>
    <!-- Defines the connection to the master database for MySQL. -->
    <datasource driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://localhost:53306/master?autoReconnect=true"
      username="apg" password="watch4net"
      maxActive="10" maxIdle="5" validationQuery="SELECT_1" testOnBorrow="false"
testWhileIdle="true"
      timeBetweenEvictionRunsMillis="10000" minEvictableIdleTimeMillis="60000" maxWait
="10000"
      removeAbandoned="true" removeAbandonedTimeout="60" logAbandoned="true" />
    <!-- Example of connection to the master database for Oracle 10g. -->
```

```
<!--
<datasource driverClassName="oracle.jdbc.OracleDriver"
  url="jdbc:oracle:thin:@localhost:1521:APG"
  username="apg" password="watch4net"
  maxActive="10" maxIdle="5" validationQuery="SELECT 1 FROM DUAL" testOnBorrow="
false" testWhileIdle="true"
  timeBetweenEvictionRunsMillis="10000" minEvictableIdleTimeMillis="60000" maxWait
="10000"
  removeAbandoned="true" removeAbandonedTimeout="60" logAbandoned="true"
  connectionProperties="SetBigStringTryClob=true" />
-->
</master-database>
</config>
```