ChannelFinder – Enhanced Directory Service API Description

prepared by: Ralph Lange Version 3.2 (20 Apr 10)

Directory Data Structure

The directory contains directory entries.

Each directory entry consists of a *channel name*, an arbitrary set of *properties* (name-value pairs), and an arbitrary set of *tags* (names).

Each of these elements has an *owner* group, which can be set by the user that created it.

All names and values are strings. Tags and property names are forced to be lower case; the first character must be alphanumerical.

Service Type

The ChannelFinder service is implemented as a REST style web service. Directory data can be uploaded and retrieved in XML or JSON notation.

Authorization, Authentication, and Encryption

No authentication or encryption is required to query the service.

All methods that are modifying the directory require authentication and proper authorization to succeed. To avoid compromising authentication data, encrypted transport is required for these operations. Standard framework-supplied web server techniques are used.

For each authenticated user an additional directory request is made (default implementation LDAP) to query the user's group memberships.

Ownership Restrictions

All properties with the same name and all tags with the same name must have the same owner. Users can only set the ownership of entries to a group they belong to. (Unless they have Admin role.)

This allows to bind properties and tags to a certain user group, making sure only users of that group can change property values and/or add the property or tag to new channels.

XML Representation

Table 1 and Table 2 show the XML and JSON representations of directory entries, which is the payload format of the web service transactions.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<channels>
  <channel name="SR:C01-MG:G02A&lt;QDP:H2&gt;Fld:SP" owner="irmis">
    <properties>
      <property name="domain" value="storage ring" owner="irmis"/>
      <property name="cell" value="01" owner="irmis"/>
      <property name="element" value="quadrupole" owner="irmis"/>
      <property name="unit" value="field" owner="irmis"/>
      <property name="type" value="setpoint" owner="irmis"/>
    </properties>
    <tags>
      <tag name="joes-quaps" owner="operator"/>
      <tag name="archived" owner="irmis"/>
    </tags>
  </channel>
  <channel name="SR:C01-MG:G02A&lt;QDP:H2&gt;Fld:RB" owner="irmis">
  …
  </channel>
</channels>
```

Table 1: XML Representation of Directory Data

```
{"channels":{"channel":[
  {"@name":"SR:C01-MG:G02A<QDP:H2>Fld:SP","@owner":"irmis",
    "properties":{"property":[
      {"@name":"domain","@value":"storage ring","@owner":"irmis"},
      {"@name":"cell","@value":"01","@owner":"irmis"},
      {"@name":"element","@value":"quadrupole","@owner":"irmis"},
      {"@name":"unit","@value":"field","@owner":"irmis"},
      {"@name":"type","@value":"setpoint","@owner":"irmis"}]},
    "tags":{"tag":[
      {"@name":"joes-quaps","@owner":"operator"},
      {"@name":"archived","@owner":"irmis"}]}},
  {"@name":"SR:C01-MG:G02A<QDP:H2>Fld:RB","@owner":"irmis",
  …
  ]}}
```

Table 2: JSON Representation of Directory Data

Web Service URLs and Operations

The ChannelFinder web service tries to follow the rules for REST style services. i.e. it operates on hierarchically organized data, where the URL specifies the entity, the HTTP method specifies the operation, and the payload specifies the data.
PUT operations are always idempotent and replace the entity with the payload, POST operations add or update subordinates, GET retrieves, and DELETE deletes the entity.

All shown URLs are relative to the service's base URL, which is configured in the web server.

Query Operations

Query by Pattern

```
.../channels?prop1=patt1&prop2=patt2&~tag=patt3&~name=patt4...
```

Method: GET Required Role: None

Return the list of channels which match all given expressions, i.e. the expressions are combined in a logical AND.

There are three types of expressions:

1. Property wildcards: <name>=<pattern>
   Matches if a channel has a property with the given *name*, and its value matches the given *pattern*. Multiple expressions for the same property name are combined in a logical OR.
2. Tag wildcards: ~tag=<pattern>
   Matches if a channel has a tag that matches the given *pattern*.
3. Channel name wildcards: ~name=<pattern>
   Matches if a channel name matches the given *pattern*.

Special keywords, e.g. "~tag" and "~name" for tag and channel name matches, have to start with the tilde character, else they are treated as property names.

The patterns may contain file glob wildcards, i.e. "?" for a single character and "*" for any number of characters.

Examples:

```
.../channels?domain=storage+ring&element=*+corrector&type=readback
```

Returns a list of all readback channels for storage ring correctors.

```
.../channels?cell=14&type=setpoint&~tag=archived
```

Returns a list of all archived setpoint channels in cell 14.

```
.../channels?~name=SR:C01-MG:G02A%3CQDP:H2%3EFld:*
```

Returns a list of all channels whose names start with "`SR:C01-MG:G02A<QDP:H2>Fld:`".

Channel Operations

Add Multiple Channels

```
.../channels
```

Method: POST Payload: List of Channels Required Role: Channel

Add the channels in the payload to the directory. Existing channels are replaced by the payload data.

Retrieve a Channel

`.../channel/<name>`

Method: GET Required Role: None

Return a single channel with the given *name*.

Create/Update a Channel

`.../channel/<name>`

Method: PUT Payload: Single Channel Required Role: Channel

Create or completely replace the existing channel *name* with the payload data.

Delete a Channel

`.../channel/<name>`

Method: DELETE Required Role: Channel

Delete the existing channel *name* and all its properties and tags.

The authenticated user must be a member of the group that owns the channel to be deleted. (A user with Admin role can delete any channel.)

Add/Update Properties and Tags of a Channel

`.../channel/<name>`

Method: POST Payload: Single Channel Required Role: Property

Create or update (replace) properties and tags of the existing channel *name* with the payload data.

The authenticated user must be a member of all groups that own properties and tags to be updated. (A user with Admin role can update tags and properties owned by any group.)

Tag Operations

Retrieve Channels by Tag

`.../tags/<name>`

Method: GET Required Role: None

Return the list of channels that are tagged with the given *name*.

Note: This operation is equivalent to: .../channels/query?Tag=<name>

Set Tag on Channels

`.../tags/<name>`

Method: PUT Payload: List of Channels Required Role: Tag

Set tag with the given *name* exclusively on all channels in the payload data, removing it from all channels that are not included in the payload. If the tag already exists in the directory, the payload data does not require tag or property entries (owner of the existing entries will be used for added tags).

The authenticated user must belong to the group that owns the tags to be removed. (A user with Admin role can set or remove tags owned by any group.)

Add Tag to Channels

`.../tags/<name>`

Method: POST Payload: List of Channels Required Role: Tag

Add tag with the given *name* to all channels in the payload data. If the tag already exists in the directory, the payload data does not require tag or property entries (ownership of added tags is forced to match the existing entries).

Remove Tag from Channels

`.../tags/<name>`

Method: DELETE Required Role: Tag

Remove tag with the given *name* from all channels.

The authenticated user must belong to the group that owns the tags to be removed. (A user with Admin role can remove any tags.)

Single Tag Operations

Add Tag to Channel

`.../tags/<tag_name>/<channel_name>`

Method: PUT Payload: Single Tag Required Role: Tag

Add tag with the given *tag_name* to the channel with the given *channel_name*.

The owner attribute of the payload is optional for tags that already exist in the directory.

Remove Tag from Channel

`.../tags/<tag_name>/<channel_name>`

Method: DELETE Required Role: Tag

Remove tag with the given *tag_name* from the channel with the given *channel_name*.

The authenticated user must belong to the group that owns the tag to be removed. (A user with Admin role can remove any tag.)