ChannelFinder – Enhanced Directory Service API Description

prepared by: Ralph Lange for ChannelFinder V1.1 (25 Jan 12)

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 the element.

All names and values are strings.

Tags and property names are case insensitive when used in pattern matching, the capitalization used when adding a tag or property for the first time is enforced in all later operations. Channel names are always case sensitive. Owner (group) names are always case insensitive, they are forced to lower case in all write operations.

Service Type

The ChannelFinder service is implemented as a REST style web service, which – in this context – means:

- The URL specifies the data element that the operation works upon.
- The HTTP method specifies the type of operation.
  GET: retrieve or query, does not modify data
  PUT: create or update, replacing the addressed element
  POST: create or update subordinates of the addressed element
  DELETE: delete the addressed element
  All operations are idempotent, i.e. when repeatedly applying the identical operation, only the first execution will change the database.
- The payload (HTTP body) always contains a representation of data.
- See http://en.wikipedia.org/wiki/Representational_State_Transfer for a detailed discussion.

Directory data can be uploaded and retrieved in XML or JSON notation, the client specifies the type using standard HTTP headers (“Content-Type”, “Accepts”).

Ownership Restrictions

All channels, properties, and tags have an owner. The owner is intended to be the name of a group, the service can find out membership through LDAP. Users can only set the ownership of entries to a group they belong to. (Unless they have Admin role.)

This allows to bind each property or tags to a certain user group, making sure only users of that group can change property values, the ownership of existing properties and tags, or delete properties and tags..

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 (implemented through LDAP) to query the user's group memberships.

The service's web descriptor defines four roles that are used in authorization, listed here ordered by decreasing rights. Each level includes all prior levels.

- Administrator: Overrides all ownership restrictions.
- ChannelMod: Needed for all operations that create, modify, or delete channels.
- PropertyMod: Needed for all operations that create, modify, or delete properties, or to modify property values.
- TagMod: Needed for all operations that create, modify, or delete tags.

XML Representation

Table 1 and Table 2 show the XML and JSON representations of directory entries, i.e. 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)

```
{"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 (Channels)

Payload data for properties and tags is the same as when part of a channel. Property and tag payloads may embed a <channels> list. This allows the operation that creates a tag to also attach the tag to a set of channels, and the operation that creates a property to set the property on the embedded list of channels. In the latter case the property's by-instance values are taken from the matching <property> item inside the property's channel list.

Null Values and Empty Property Value Strings

Property values of null are not defined. They are never returned by the service, and specifying them in payload data sent to the service fails as an error.

Empty property value strings are never returned by the service. Inside a sngle channel POST operation payload, empty strings are interpreted as “delete property” requests. Specifying them in any other operation payload data fails as an error.

Web Service URLs and Operations

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

Bulk Channel Operations

List Channels / Query by Pattern

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

Method: GET Returns: List of Channels 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. Value wildcards: <name>=<pattern>
   True 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 name wildcards: ~tag=<pattern>
   True if a channel has a tag or property whose name matches the given *pattern*.
3. Channel name wildcards: ~name=<pattern>
   True 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 in a value wildcard expression.

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

If called without URL parameters, the operation lists all channels in the directory.

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:`".
Note that a number of special characters need to be escaped in URL expressions – in most cases the browser or API library will do the escaping.

Add Multiple Channels

`.../channels`

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

Add the channels in the payload to the directory. Existing channels are replaced by the payload data. For all channels that are to be replaced or added, the authenticated user is required to be a member of their owner group. (Administrator role overrides this restriction.)

Single Channel Operations

Retrieve a Channel

`.../channels/<name>`

Method: GET Returns: Single Channel Required Role: None

Return the full listing of a single channel with the given *name*.

Create/Update a Channel

`.../channels/<name>`

Method: PUT Payload: Single Channel Required Role: ChannelMod

Create or completely replace the existing channel *name* with the payload data. If the channel exists, the authenticated user is required to be a member of its owner group. (Administrator role overrides this restriction.)

Add/Update Properties and Tags of a Channel

`.../channels/<name>`

Method: POST Payload: Single Channel Required Role: ChannelMod

Add or update properties and tags of the existing channel *name* with the payload data. If the payload channel's name or owner are different from the current values, the database name/owner are changed.

An emtpy string as a payload property value is interpreted as a request to delete that property from the channel. This allows to add/update/remove properties for a channel in one request.

The authenticated user must be a member of the group that owns the channel. If the operation changes the ownership, the user must belong to both the old and the new group. (Administrator role overrides these restrictions.)

Delete a Channel

`.../channels/<name>`

Method: DELETE Required Role: ChannelMod

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. (Administrator role overrides this restriction.)

Bulk Property Operations

List Properties

`.../properties`

Method: GET Returns: List of Properties Required Role: None

Return the list of all properties in the directory.

Add Multiple Properties

`.../properties`

Method: POST Payload: List of Properties Required Role: PropertyMod

Add the properties in the payload to the directory. If a payload property contains an embedded <channels> list, the property is added to all channels in that list. In this case, the value for each property instance is taken from the property definition inside the channel on the embedded channel list. The property is set exclusively on all channels in the embedded list, removing it from all channels that are not included on the list. Existing property values are replaced by the payload data.

For all properties that are to be replaced or added, the authenticated user is required to be a member of their owner group. (Administrator role overrides this restriction.)

Single Property Operations

Retrieve a Property

`.../properties/<name>`

Method: GET Returns: Single Property Required Role: None

Return the property with the given *name*, listing all channels with that property in an embedded <channels> structure.

Create/Update a Property

`.../properties/<name>`

Method: PUT Payload: Single Property Required Role: PropertyMod

Create or completely replace the existing property *name* with the payload data. If the payload contains an embedded <channels> list, the property is added to all channels in that list. In this case, the value for each property instance is taken from the property definition inside the channel in the embedded channel list. The property is set exclusively on all channels in the payload data, removing it from all channels that are not included in the payload. Existing property values are replaced by the payload data.

The authenticated user must belong to the group that owns the property. (Administrator role overrides this restriction.)

Add Property to Multiple Channels

`.../properties/<name>`

Method: POST Payload: Single Property Required Role: PropertyMod

Add property with the given *name* to all channels in the payload data. If the payload contains an embedded <channels> list, the property is added to all channels in that list. In this case, the value for each property instance is taken from the property definition inside the channel in the embedded channel list. Existing property values are replaced by the payload data. If the payload property name or owner are different from the current values, the database name/owner are changed.

The authenticated user must belong to the group that owns the property. If the operation changes the ownership, the user must belong to both the old and the new group. (Administrator role overrides these restrictions.)

Remove Property

`.../properties/<name>`

Method: DELETE Required Role: PropertyMod

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

The authenticated user must belong to the group that owns the property. (Administrator role overrides this restriction.)

Single Property-On-Channel Operations

Add Property to Single Channel

`.../properties/<property_name>/<channel_name>`

Method: PUT Payload: Single Property Required Role: PropertyMod

Add property with the given *property_name* to the channel with the given *channel_name*. An existing property value is replaced by the payload data.

The authenticated user must belong to the group that owns the property. (Administrator role overrides this restriction.)

Remove Property from Single Channel

`.../properties/<property_name>/<channel_name>`

Method: DELETE Required Role: PropertyMod

Remove property with the given *property_name* from the channel with the given *channel_name*.

The authenticated user must belong to the group that owns the property to be removed. (Administrator role overrides this restriction.)

Bulk Tag Operations

List Tags

`.../tags`

Method: GET Returns: List of Tags Required Role: None

Return the list of all tags in the directory.

Add Multiple Tags

`.../tags`

Method: POST Payload: List of Tags Required Role: TagMod

Add the tags in the payload to the directory. If a payload tag contains an embedded <channels> list, the tag is added to all channels in that list. The tag is set exclusively on all channels in the embedded list, removing it from all channels that are not included.

For all tags that are to be replaced or added, the authenticated user is required to be a member of their owner group. (Administrator role overrides this restriction.)

Single Tag Operations

Retrieve a Tag

`.../tags/<name>`

Method: GET Returns: Single Tag Required Role: None

Return the tag with the given *name*, listing all tagged channels in an embedded <channels> structure.

Create/Update a Tag

`.../tags/<name>`

Method: PUT Payload: Single Tag Required Role: TagMod

Create or completely replace the existing tag *name* with the payload data. If the payload contains an embedded <channels> list, the tag is added to all channels in that list. The tag is set exclusively on all channels in the payload data, removing it from all channels that are not included in the payload.

The authenticated user must belong to the group that owns the tag. (Administrator role overrides this restriction.)

Add Tag to Multiple Channels

`.../tags/<name>`

Method: POST Payload: Single Tag Required Role: TagMod

Add tag with the given *name* to all channels in the payload data. If the payload contains an embedded <channels> list, the tag is added to all channels in that list. If the payload tag name or owner are different from the current values, the database name/owner are changed.

The authenticated user must belong to the group that owns the tag. If the operation changes the ownership, the user must belong to both the old and the new group. (Administrator role overrides these restrictions.)

Remove Tag

`.../tags/<name>`

Method: DELETE Required Role: TagMod

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

The authenticated user must belong to the group that owns the tag. (Administrator role overrides this restriction.)

Single Tag-On-Channel Operations

Add Tag to Single Channel

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

Method: PUT Payload: Single Tag Required Role: TagMod

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

The authenticated user must belong to the group that owns the tag. (Administrator role overrides this restriction.)

Remove Tag from Single Channel

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

Method: DELETE Required Role: TagMod

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. (Administrator role overrides this restriction.)

Overview of Operations

Table 3 shows a tabular overview of the ChannelFinder operations.

`.../channels`			`.../properties`			`.../tags`		
GET	*query*	query channels	GET	*list*	list all properties	GET	*list*	list all tags
POST	*add*	add many channels	POST	*add*	add many properties	POST	*add*	add many tags
`.../channels/<chan>`			`.../properties/<prop>`			`.../tags/<tag>`		
GET	*read*	read one channel	GET	*read*	read one property	GET	*read*	read one tag
PUT	*create*	create/replace one channel	PUT	*create*	create/replace one property	PUT	*create*	create/replace one tag
POST	*update*	update one channel, set name/owner	POST	*update*	update one property, set name/owner	POST	*update*	update one tag, set name/owner
DELETE	*remove*	remove a channel	DELETE	*remove*	remove a property from all channels	DELETE	*remove*	remove a tag from all channels
			`.../properties/<prop>/<chan>`			`.../tags/<tag>/<chan>`		
			PUT	*add*	add a property to one channel	PUT	*add*	add a tag to one channel
			DELETE	*remove*	remove a property from one channel	DELETE	*remove*	remove a tag from one channel
requires ChannelMod role (except GET)			*requires PropertyMod role (except GET)*			*requires TagMod role (except GET)*		