Fews Workflow Runner service

Introduction

The Fews Workflow Runner service uses XFire, a java SOAP framework. This framework allows a client application to obtain a proxy instance to the FewsWebServiceRunner API. With this API the client can run workflows on the MC from the client code. The timeseries produced by the workflow run can read by the client application. Before a client application can access the FEWS system there is some configuration work that needs to be done.

Parallel processing functionality from 2017.01 (<multipleForecastingShells>true</multipleForecastingShells>) is currently not possible via this service.

> ⚠️ *User's looking to use XFire on a new project, should use CXF instead. CXF is a continuation of the XFire project and is considered XFire 2.0. It has many new features, a ton of bug fixes, and is now JAX-WS compliant! XFire will continue to be maintained through bug fix releases, but most development will occur on CXF now. For more information see the XFire/Celtix merge FAQ and the CXF website.*

Fews Workflow Runner Service API

Description of the methods provided by the Fews Workflow Runner Service API.

```
TimeSeries[] runFewsWorkflow(String clientId, String workflowId, Date forecastStartDateTime,
                Date forecastDateTime0, Date forecastEndDateTime, TimeSeries[] inputTimeSeries)
                throws Exception;
```

Runs a FEWS workflow on the MC.

- **clientId**: A descriptive id used in logging and passed as *user id* in the taskProperties. Required
- **workflowId**: A workflow id known by the MC configuration. Required
- **forecastStartDateTime**: The start time of the forecast. If provided a module state at or before the start time will be used. When not specified the the forecast will start at the last available warm state or will use a cold state when no warm state is available. **WARNING !** Because XFire does not support nulls for date/times pass new Date(0) instead of null. Optional
- **forecastDateTime0**: The time for new saved states during this run, a time observed data is likely to be available for all stations. When not specified the current time will be used. **WARNING!** Because XFire does not support nulls for date/times pass new Date(0) instead of null.
- **forecastEndDateTime**: The end time of the forecast. When not specified a default is used specified in the fews configuration. **WARNING!** Because XFire does not support nulls for date/times pass new Date(0) instead of null. Optional.
- **inputTimeSeries**: The input timeseries required by the workflow.
- **returns**: The output timeseries produced by the workflow.
- **throws**: An exception when something goes wrong.

Installing a Workflow Runner Service

The Workflow Runner Service actually consists of two service components. The first service component is the **McTaskWebService** and is hosted by the MC. The second service component is the **FewsWebService** which is the component being described under heading Fews Workflow Runner Service API. The FewsWebService is started up by the client application.

Installing the MC Service component

The MC Service component is started up by the MC (TaskWebServiceRunner). The client application does not use this service directly. The only configuration required for this component is that the following line is added to the MC configuration file; **fews.master.mc.conf**.

```
<webservice port="8899" sleeptime="15" timeout="600"/>
```

Installing the Workflow Runner

The Workflow Runner service requires a configuration file that is an instance of the **WebService.xsd**.

- **port**: This is the port number on which the FewsWebService will be hosted. This port must be accessible by the client application.
- **timeOutSeconds**: This is the length of time that the FewsWebService will wait for the workflow to complete runnging.
- **inputPiTimeSeriesFile**: This is the file from which the MC workflow run will read the input timeseries. When calling the FewsWebService API the timeseries passed as argument will be written to this file. This file must therefore match the file configured in the MC workflow.
- **outputPiTimeSeriesFile [1..>**: These are the files to which the MC workflow will write the output timeseries. When calling the FewsWebService API the timeseries are read from the output files after the workflow run is completed. These timeseries are returned by the call. These files must therefore match the files configured in the MC workflow.
- **mcTaskWebService**: This contains information that allows the FewsWebService to connect to a specific running instance of the McTaskWebService. Although this entry is optional it is required!

Example of a WebService xml file.

Starting on Windows

Step 1: Install the webservice package.TODO: Attach the webservice package!

Step 2: Make a new FewsWebService.exe and FewsWebService.jpif file in the \bin directory. The FewsWebService.jpif must contain the following information.

```
..\jre
-mx512m
-cp
$JARS_PATH$
nl.wldelft.fews.webservice.FewsWebServiceRunner
<path>/WebService.xml
```

Step 4: Start the FewsWebServiceRunner by clicking on the FewsWebServiceRunner.exe.

Step 5: Stop the FewsWebServiceRunner by killing the application using the System Monitor.

Install windows service

TODO: Package not available!

Starting on Linux

Step 1: Install the webservice package.TODO: Attach the webservice package!

Step 2: Set the correct paths in the fews_webservice.sh script.

Step 3: Start the fews_webservice.sh script by typing **./fews_webservice.sh start**

Step 4: To stop the service type **./fews_webservice.sh stop**

To make sure that the service keeps running there is also a 'watcher' script. This script should be run as a cron job. What this script does is, check if the web fews webservice script is still running. If the service is not running the it is restarted.

Example code

Here are some examples of how a client application would instantiate a FewsWebService and fire of a workflow to the MC.

Before starting the client will require the following library: xfire-all-1.2.5.jar. This library can be found in the bin directory of the FEWS system.

Setting up a connection

```
ObjectServiceFactory serviceFactory = new ObjectServiceFactory();
Service service = serviceFactory.create(FewsWebService.class);
XFireProxyFactory proxyFactory = new XFireProxyFactory();
 //port Number must be equal to the number configured in the WebService.xml
int portNumber = 8100;
//localhost can be replaced by the ip address of the machine on which the WebService is running.
FewsWebService serviceProxy = (FewsWebService) proxyFactory.create(service, "http://localhost:" + portNumber + "
/FewsWebService")
```

Running Workflows

```
FewsWebService proxy = getProxy(); //See section "Setting up a connection"

TimeSeries[] inputArrays = getInputArrays(); //Get the timeseries required to run workflow.

//Run a workflow
try {
TimeSeries[] outputArrays = proxy.runFewsWorkflow("MyWebService", "workflowId", new Date(0), new Date(0), new
Date(0), inputArrays);
} catch (Exception e){
// do something.
}
```

Appendix

WebService XSD

Example of the WebService XSD