# URBS parameter adapter

For URBS there are two adapters, one made by the developers of the URBS code Don Carroll and one made by Deltares. The URBS adapter developed by Don Carroll takes care of the conversion of the time series files from PI XML format to native URBS files and model logging. The URBS adapter developed by Deltares supports the conversion of model parameters from standard Delft-FEWS PI-format to URBS ini files.

The model adapters activities for URBS are therefore split in two adapter documents:

- URBS parameter adapter from Deltares (this wiki), responsible for:
  - Conversion of the Delft-FEWS PI Parameter XML file to the *.ini file that will be used by the URBS pre adapter. This parameters file can incorporate location attribute modifiers written by Delft-FEWS
  - Documentation of this adapter can be found on this page.
- URBS pre and post adapter from Don Carroll, responsible for:
  - Extraction of the start date, end date and time increment for the data series from the startDate, endDate and timeStep.
    - Please note that the pre and post adapter will assume a GMT+10 time zone, even if you specify differently. For any questions regarding this behavior (which does not correspond to the documentation, see link below), please contact Don Carroll.
  - Conversion of time series files for rainfall, gauging station and inflow data (i.e. the ".r", ".g" or '.i" files) from Delf-FEWS PI XML to native URBS formatted files.
  - Creation of the modelAdapter.bat file, to start the URBS model
  - Conversion of the URBS output files to a single Delft-FEWS PI XML time series file
  - Conversion of log messages from both the model run and the pre- and post adapter to Delft-FEWS PI XML log files
  - For more info see URBS pre and post adapter

## Version control of URBS parameter adapter

- 2018.02 and up: fews-urbs-parameter-adapter.jar (updated version due to upgraded jre version java 2013)
- Pre 2018.02: fews-urbs-parameter-adapter.jar
- Deprecated: UrbsIniAdapter.jar

## URBS Catchment Layout

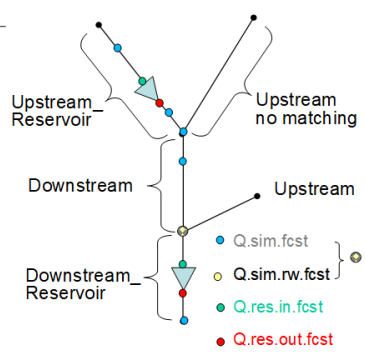
The modeling unit of URBS is the URBS sub-catchment. Within an URBS sub-catchment, the vector file defines the inputs and outputs to the model. Input is rain at the subareas as well as flows from any upstream sub-catchment. Outputs calculated in the sub-catchment are simulated river flows (Q.sim.fcst) and possibly simulated reservoir inflow (Q.res.in.fcst) and outflows (Q.res.out.fcst). In addition, URBS can rewrite an input from an upstream catchment as an (adjusted) output. This output is adjusted when the matching (i.e. error correction) is turned on for this station. Since it is very inconvenient for the FEWS configuration to have two similar flow parameters at a similar location, a different parameter is chosen for flows that are rewritten after (possible) adjustment: Q.sim.rw.fcst.

Given that some sub-catchments have been modeled without a monitoring station that can provide an observed level/rated flow for matching, the following common situations have been identified with different combinations of model input and output time series (see Figure):

- Upstream sub-catchments without matching stations
- Upstream sub-catchments (with matching stations)
- Upstream sub-catchment stations with reservoirs (and matching stations)
- Downstream sub-catchments (with matching stations)
- Downstream sub-catchments with reservoirs.

Typical URBS catchment layouts

| Type | Input→U | Output→F |
|---|---|---|
| Upstream W/O matching | P.merged<br>Q.rated | <br>Q.sim.fcst |
| Upstream + reservoir | P.merged<br>Q.rated | Q.res.in.fcst<br>Q.res.out.fcst<br>Q.sim.fcst |
| Downstream | P.merged<br><br>Q.sim.fcst | Q.sim.fcst<br>Q.rated<br>Q.sim.rw.fcst |
| Downstream + reservoir | P.merged<br>Q.rated<br>Q.sim.fcst | Q.res.in.fcst<br>Q.res.out.fcst<br>Q.sim.rw.fcst<br>Q.sim.fcst |




## Delft-FEWS General Adapter

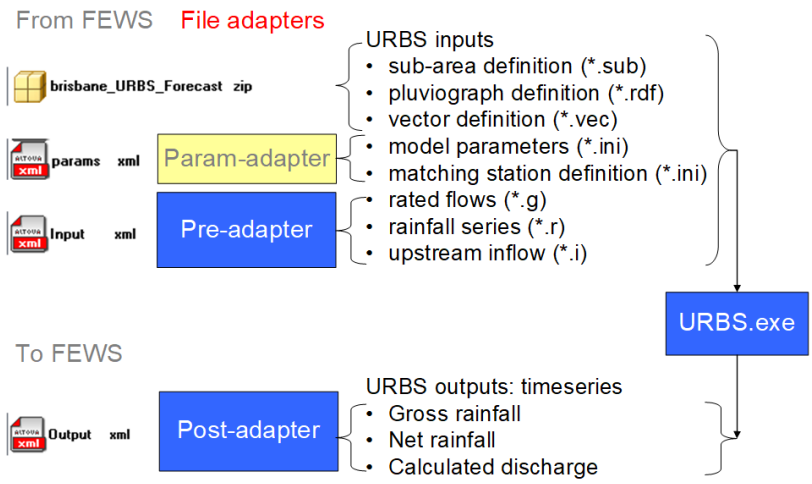
Just like any model deployed within a Delft-FEWS application, the URBS model is executed via the Delft-FEWS General Adapter. In the GA module instance four main activities can be identified:

1. Start-up activities that delete old URBS rainfall input, output and log files from the URBS module folders.

2. Export activities that export the URBS model parameters and rainfall time series into the correct URBS module folders
3. Execute activities that start the URBS parameter adapter, URBS pre-adapter, URBS model executable and URBS post-adapter
4. Import activities that import the flow time series.

This page will describe in more detail the data flow through the various adapters as well as the export, execute and import activities from the General Adapter perspective. Different flavours of this General Adapter file may exist, this depends on the different sub-catchment layouts as indicated above. The differences are however minimal.

The figuer below illustrates the data flow for URBS.




On the input side, two adapters are executed. The URBS vector file, the rdf file and the sub-file are contained in the ModuleDataSet of this sub-catchment, which is unzipped by the general adapter. The URBS parameter adapter converts the parameterization as provided by the sub-catchment location attributes in Delft-FEWS (in the parameters.xml file) into the model.ini file which can be read by the URBS executable. If the Loss model is changed, this adapter will update the rdf file as well. The URBS pre-adapter converts the input time series as provided by Delft-FEWS (in the inputs.xml file) into the various time series files which can be read by the URBS executable. The URBS executable generates a number of output files, which are converted back into a PI-timeseries.xml file by the Post Adapter. The resulting time series file is ingested by the General Adapter. The URBS executable generates a log file. Each adapter generates a PI-diagnostics xml file for its own work, while the post adapter incorporates the log from the URBS executable in its diagnostics file.

The structure of a General Adapter file follows the activities as described above.

### General

First a General section is required to indicate where the model will be run and other properties.

**general**

```
<general>
        <piVersion>1.10</piVersion>
        <rootDir>%REGION_HOME%\Modules\urbs\$CATCHMENT$\$SUBCATCHMENT$</rootDir>
        <workDir>%ROOT_DIR%\model</workDir>
        <exportDir>%ROOT_DIR%\input</exportDir>
        <exportDataSetDir>%REGION_HOME%\Modules</exportDataSetDir>
        <updateExportDataSetDirOnlyOnChange>true</updateExportDataSetDirOnlyOnChange>
        <exportIdMap>IdExportURBS</exportIdMap>
        <importDir>%ROOT_DIR%\output</importDir>
        <importIdMap>IdImportURBS</importIdMap>
        <dumpFileDir>$GA_DUMPFILEDIR$</dumpFileDir>
        <dumpDir>%ROOT_DIR%</dumpDir>
        <diagnosticFile>%ROOT_DIR%\logs\urbsdiag.xml</diagnosticFile>
        <missVal>-99.0</missVal>
        <convertDatum>true</convertDatum>
        <timeZone>
                <timeZoneName>GMT+10</timeZoneName>
        </timeZone>
  </general>
```

### Export Activities

The Export Activities element contains several sub-elements for exporting all data required by URBS.

```
exportActivities
  exportStateActivity
    moduleInstanceId   URBS_State
    stateExportDir     %ROOT_DIR%/states
    stateConfigFile    %ROOT_DIR%\states\states.xml
    stateLocations
      type  file
      stateLocation
        readLocation   states.txt
        writeLocation  states.txt
    stateSelection
  exportTimeSeriesActivity
    exportFile         Input.xml
    timeSeriesSets
      timeSeriesSet (4)
  exportDataSetActivity
    moduleInstanceId   $CATCHMENT$_URBS_Forecast
  exportParameterActivity
    fileName           params.xml
    templateLocationLooping
      locationModelLoop (2)
        | locationId                | locationSetId                          | model
      1 | $CATCHMENT$_$SUBCATCHMENT$ |                                        | Subcatchments
      2 |                            | URBS_matching.$CATCHMENT$_$SUBCATCHMENT$ | Matching
    moduleInstanceId   URBS_Parameters
  exportRunFileActivity
    exportFile         paramAdapter_runinfo.xml
    properties
      string (2)
        | key          | value
      1 | urbsIniFile  | %ROOT_DIR%\model\$SUBCATCHMENT$.ini
      2 | rdfFile      | %ROOT_DIR%\model\$SUBCATCHMENT$.rdf
```

- A dummy state file
- The inputs.xml time series file, holding the input time series for the locations and parameters as defined above.
- The vector, rdf and sub files extracted from the ModuleDataSet.zip file for this sub-catchment
- The parameters.xml, being populated with the model-parameters attributes for the sub-catchment, the matching flags for the relevant matching locations and (if any) the VBF parameters for any reservoir location in the sub-catchment.
- A runinfo.xml file, informing the adapters about directories and files handed over, with specific information of needed.

The example shows an export parameter activity that not only exports the params.xml file, but also replaces location attributes in the params.xml; this is usefull when model parameters are modified in the Delft-FEWS GUI. If no modifiers are used the params.xml file can just be exported with the following code.

**exportParameterActivity**

```
<exportParameterActivity>
       <fileName>params.xml</fileName>
       <moduleInstanceId>URBS_Parameters</moduleInstanceId>
</exportParameterActivity>
```

The exportRunFileActivity cna also be configured in multiple flavours. This activity exports a pi run file in xml format, containing general information about the run. This file is used by the URBS parameter adapter. Sometimes the URBS Bin folder key is also exported as shown below.

**exportRunFileActivity**

```
<exportRunFileActivity>
     <exportFile>paramAdapter_runinfo.xml</exportFile>
     <properties>
          <string key="urbsIniFile" value="%ROOT_DIR%\model\$SUBCATCHMENT$.ini"/>
          <string key="rdfFile" value="%ROOT_DIR%\model\$SUBCATCHMENT$.rdf"/>
          <string key="URBS_BIN" value="$MODULES_BIN$\urbs"/>
     </properties>
</exportRunFileActivity>
```

## Execute Activities

Before executing the URBS model, two adapters are called: the parameter adapter to create the model.ini file and the pre-adapter to convert the time series (see Figure above). As can be noted, each adapter creates its own diagnostics file to keep track of success and failure.

**Execute Activity**

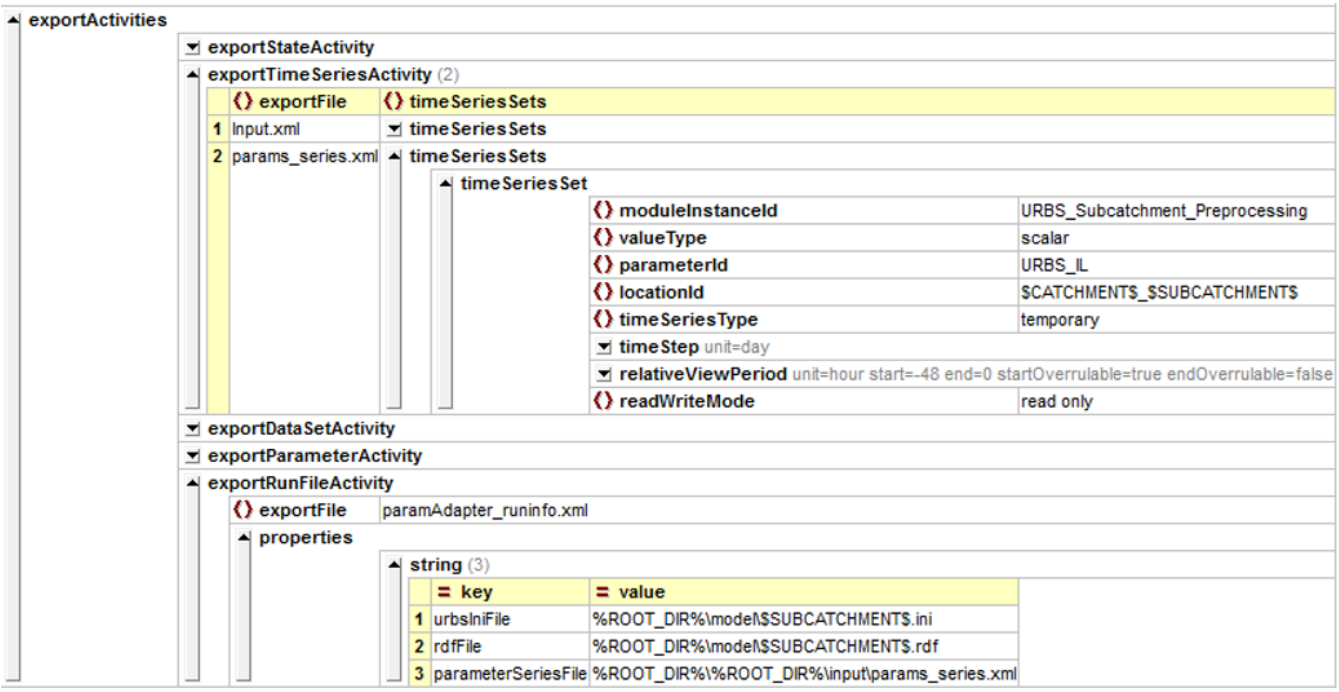
```
<executeActivities>
        <executeActivity>
                <description>Run parameter adapter</description>
                <command>
                        <className>nl.wldelft.urbs.ParameterAdapter</className>
                        <binDir>%ROOT_DIR%\..\..\..\delft-adapters</binDir>
                </command>
                <arguments>
                        <argument>%ROOT_DIR%\input\paramAdapter_runinfo.xml</argument>
                </arguments>
                <timeOut>400000</timeOut>
        </executeActivity>
        <executeActivity>
                <description>Run PreAdapter</description>
                <command>
                        <executable>%ROOT_DIR%\..\..\bin\preadapter.exe</executable>
                </command>
                <arguments>
                        <argument>%ROOT_DIR%\input\Input.xml</argument>
                        <argument>%ROOT_DIR%\model\$SUBCATCHMENT$.ini</argument>
                        <argument>%ROOT_DIR%\logs\urbsdiag_pre.xml</argument>
                        <argument>%ROOT_DIR%\model\$SUBCATCHMENT$.vec</argument>
                        <argument>%ROOT_DIR%\model\$SUBCATCHMENT$.rdf</argument>
                </arguments>
                <timeOut>1200000</timeOut>
                <overrulingDiagnosticFile>%ROOT_DIR%\logs\urbsdiag_pre.xml</overrulingDiagnosticFile>
        </executeActivity>
        <executeActivity>
                <description>Run modelAdapter (using cmd.exe /c to properly terminate)</description>
                <command>
                        <executable>c:\$WINCOMMANDSHELL$</executable>
                </command>
                <arguments>
                        <argument>/c</argument>
                        <argument>%ROOT_DIR%\model\modelAdapter.bat</argument>
                </arguments>
                <timeOut>1200000</timeOut>
        </executeActivity>
        <executeActivity>
                <description>Run PostAdapter</description>
                <command>
                        <executable>%ROOT_DIR%\..\..\bin\postadapter.exe</executable>
                </command>
                <arguments>
                        <argument>%ROOT_DIR%\output\$SUBCATCHMENT$.csv</argument>
                        <argument>%ROOT_DIR%\logs\urbsout.log</argument>
                        <argument>%ROOT_DIR%\logs\urbserr.log</argument>
                        <argument>%ROOT_DIR%\output\Output.xml</argument>
                        <argument>%ROOT_DIR%\logs\urbsdiag_post.xml</argument>
                </arguments>
                <timeOut>1200000</timeOut>
                <overrulingDiagnosticFile>%ROOT_DIR%\logs\urbsdiag_post.xml</overrulingDiagnosticFile>
        </executeActivity>
</executeActivities>
```

## Import Activities

The Import Activities are not different from any other General Adapter Run. The time series from URBS are imported from the Output.xml file produced by the URBS post adapter.

## Special Functions URBS parameter adapter

For the URBS parameter adapter, some optional functionality can be configured that are unique to this adapter. This optional functionality allows editing of a parameter value with a time series; this is implemented to enable copying of URBS loss values from a time series file to the URBS parameter ini file. The time series with values for the URBS parameter file need to be exported in a new PI Time Series XML file; i.e. params_series.xml.




The exported run file requires a reference to this exported PI XML file with a key = parameterSeriesFile. When this property is configured, the URBS parameter adapter will read a PI XML file and extract the first non missing value for each time series and replaces the values in the .ini file that matches the parameterId of that time series; in this example URBS_IL. For example the value 48.0 will be taken from the time series file below and used to write "URBS_IL=48.0" to the urbs.ini file instead of the value taken from the PiParameters.xml file. This function that replaces URBS parameter values with time series values can only be done for the "Initial Dam Volumes and Levels", "Catchment and Channel Routing Parameters" and "Rainfall Runoff Parameters" sections in the URBS .ini file.

**Example parameterSeriesFile.xml**

```
<TimeSeries xmlns="http://www.wldelft.nl/fews/PI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
schemaLocation="http://www.wldelft.nl/fews/PI http://fews.wldelft.nl/schemas/version1.0/pi-schemas
/pi_timeseries.xsd" version="1.10">
    <timeZone>10.0</timeZone>
    <series>
        <header>
            <type>instantaneous</type>
            <locationId>tweed_uki</locationId>
            <parameterId>URBS_IL</parameterId>
            <ensembleId>IFD</ensembleId>
            <ensembleMemberIndex>0</ensembleMemberIndex>
            <timeStep unit="second" multiplier="86400"/>
            <startDate date="2018-05-21" time="10:00:00"/>
            <endDate date="2018-05-24" time="10:00:00"/>
            <missVal>-99.0</missVal>
            <stationName>tweed_uki</stationName>
            <lat>-28.46995606962717</lat>
            <lon>153.2671548515222</lon>
            <x>153.2671548515222</x>
            <y>-28.46995606962717</y>
            <z>0.0</z>
            <units>mm</units>
        </header>
        <event date="2018-05-21" time="10:00:00" value="-99.0" flag="8"/>
        <event date="2018-05-22" time="10:00:00" value="48" flag="0"/>
        <event date="2018-05-23" time="10:00:00" value="40" flag="0"/>
        <event date="2018-05-24" time="10:00:00" value="-99.0" flag="8"/>
    </series>
</TimeSeries>
```

## vecFile

a vec file can be configured which will get the "URBS_VEC_TAGS" group from the Parameters.xml and will replace the corresponding tags is the specified vec file with the values of the parameter.

When "URBS_ASTERISKS" group is defined in Parameters.xml all %ASTERISK_<x>% tags will be replaced with an * where the <x> corresponds to one of the values in the string value of "URBS_ASTERISKS" parameter, separated by /

For this to work, the user also needs to define the “vecFile” key to the exportRunFileActivity.

**Example URBS_VEC_TAGS parameters.xml**

```
<group id="URBS_VEC_TAGS" name="URBS Vec tags" readonly="false">
   <model>Subcatchments</model>
   <parameter id="VEC_B0" name="VEC_B0">
      <description>URBS_VEC_B0</description>
      <stringValue>27.5</stringValue>
   </parameter>
   <parameter id="VEC_BR" name="VEC_BR">
      <description>URBS_VEC_BR</description>
      <stringValue>0.875</stringValue>
   </parameter>
   <parameter id="VEC_BC" name="VEC_BC">
      <description>VEC_BC</description>
      <stringValue>0.23</stringValue>
   </parameter>
   <parameter id="VEC_BM" name="VEC_BM">
      <description>VEC_BM</description>
      <stringValue>1</stringValue>
   </parameter>
</group>
<group id="URBS_ASTERISKS" name="URBS_ASTERISKS">
   <model>Subcatchments</model>
   <parameter id="URBS_ASTERISKS" name="URBS_ASTERISKS">
      <description>URBS_ASTERISKS</description>
      <stringValue>1/3</stringValue>
   </parameter>
</group>
```

**Example URBS_VEC_TAGS exportRunFileActivity**

```
<exportRunFileActivity>
     <exportFile>paramAdapter_runinfo.xml</exportFile>
     <properties>
          <string key="urbsIniFile" value="%ROOT_DIR%\model\$SUBCATCHMENT$.ini"/>
          <string key="rdfFile" value="%ROOT_DIR%\model\$SUBCATCHMENT$.rdf"/>
          <string key="vecFile" value="%ROOT_DIR%\model\$SUBCATCHMENT$.vec"/>
                 <string key="URBS_BIN" value="$MODULES_BIN$\urbs"/>
 </properties>
</exportRunFileActivity>
```

**Example VEC with tags**

```
River KNOCKLOFTY
MODEL: SPLIT
{Developed by Terry Malone on 08/10/2009}
USES: L, E*1, U*1, I*0.1
{The default parameters are for information only}
DEFAULT PARAMETERS: alpha = 0.2 m = 0.8 beta = 2.5 n = 1 x = 0
DEFAULT PARAMETERS: K24 = 0.993
CATCHMENT DATA FILE = knocklofty.dat
Factor = 1
```

```
INPUT. Newcastle_br
PRINT.Newcastle_br_G: B0=12.5 {12} Br=0.875 {0.825} BC= 0.25  BM = 1.0
STORE.


INPUT. nire
PRINT.Nire_G:B0=5.0  Br=0.875 BC= 0.2 BM = 1.0


GET.

ROUTE THRU #14 L = 2.63 Sc = 0.0005
ADD RAIN #14 L = 2.63 Sc = 0.0005



{PRINT.Ballydonagh: Br=0.9 BC= 0.2 BM = 1.0}

STORE.
RAIN #109 L = 5.84 Sc = 0.0273
GET.
ROUTE THRU #110 L = 1.22 Sc = 0.0019
ADD RAIN #110 L = 1.22 Sc = 0.0019
ROUTE THRU #15 L = 1.58 Sc = 0.0005
ADD RAIN #15 L = 1.58 Sc = 0.0005



PRINT.KNOCKLOFTY*
PRINT.KNOCKLOFTY_G : B0=%VEC_B0% {12} Br=%VEC_BR% {0.85} BC=%VEC_BC% {0.25} BM =%VEC_BM%


end of catchment data file.

4 PLUVIOGRAPHS:


LOCATION. suir014
1 SUBAREAS: 14
LOCATION. suir015
1 SUBAREAS: 15
LOCATION. suir109
1 SUBAREAS: 109
LOCATION. suir110
1 SUBAREAS: 110


END OF PLUVIOGRAPH DATA.

{3 rating station:}
{location. newcastle_br_G*1.1}
{location. nire_G*0.7}
{location. KNOCKLOFTY_G*1.1}
{end of rating stations.}

3 gauging station:
location.  newcastle_br_G%ASTERISK_1%
location.  nire_G%ASTERISK_2%
location.  KNOCKLOFTY_G%ASTERISK_3%
end of gauging station.
```