LESSON 1 INTRODUCTION

Understanding what tools you have, where they are located, and how they are organized is critical to any profession. This lesson introduces the basics of the tools used by RFC professionals.

This lesson describes CHPS Focal Point roles and responsibilities and basic information on configuring CHPS. By the end of this lesson, you should be able to:

- Identify the Configuration Focal Point roles and responsibilities.
- List the main configuration directories.
- Describe the use of XML in CHPS.
- Review XML specifications.
- Describe the CHPS naming convention.
- Apply version control when creating files.
- Recall the CHPS file retention policy.

1.2 ROLES

The responsibilities of CHPS Configuration Focal Points vary by office. The following information provides guidelines and suggestions on managing tasks and communicating the changes made to configuration files.

Division of Tasks - Configuration processes for CHPS involve a large number of files, so you may want to divide the tasks within your RFC. Many offices find it helpful to assign people to specific tasks in order to prevent overwriting each other's files.

Responsibilities - Focal point responsibilities can include, but are not limited to:

- Configuring new sites and associated parameters and units
- Adding data from new sources
- Configuring modules and workflows

Communication Skills - Make an effort to maintain clear lines of communication. This includes steps such as maintaining a log of configuration changes, or adding comment lines when changing code. Work with other members of the staff involved in configuration and develop a documentation strategy.

Support Services - The Development and Operations Hydrologist (DOH) at your office can provide more guidance on the needs in the CHPS program at your RFC.

1.3 CONFIGURATION DIRECTORIES

As a focal point, you may be required to configure the large set of XML files defining the configuration of CHPS.

The CHPS Config directory is typically located in the **/awips/chps_share/** or for a Stand Alone instance, the**/awips/chps_share/sa/YOURFOLDER** directory.

Each of these files is contained in a set of directories, which hold different parts of the configuration.

If the system does not locate the Config directory, an error message appears and the system stops.

Click here to see general information on the CHPS directory structure from the Introduction to CHPS course.

1.4 CONFIGURATION SUBDIRECTORIES

The location of the CHPS Configuration directory may vary depending on the RFC, but the subdirectories are **standard** among **all River Forecast Centers**.

The configuration directory contains subdirectories with logical names.

Commonly Used Subdirectories

Some of the subdirectories you will become familiar with in your role as a configuration focal point include:

- **RegionConfigFiles** form the basis for this RFC-specific configuration, including all locations, parameters, etc.
- **WorkflowFiles** contain definitions of workflows for running sequences of modules.
- **ModuleConfigFiles** contain configurations of all modules used to handle data and run forecast models (ModuleParameters and ModuleDataSetFiles may also be helpful).
- **SystemConfigFiles** include definitions of the functional elements DELFT-FEWS has available (plug-ins). The layout of the time series display in the main GUI is also defined in these files.

Click the reference link at the bottom of the page for a complete list (with brief descriptions) of the configuration files located in the configuration subdirectories.

Root Configuration Files

Root configuration files define Operator Client (OC) and/or Forecasting Shell Server (FSS) behavior. The files may be unique to each OC or FSS.

Warning! DO NOT edit root configuration files without consulting the Hydrologic Services Branch!

Reference: *CHPS XML File Definitions and Locations*

1.5 XML REVIEW

CHPS uses XML to structure the configuration files affecting nearly all aspects for the forecasting system from appearance to data ingest and display.

XML stands for eXtensible Markup Language. An extensible language does not require use of standard tags, so programmers can create their own.

XML is a markup language much like HTML. It does not replace HTML. It was designed to carry data, not to display data. In other words, XML does not DO anything without software to send, receive, or display it.

Basic Syntax

The basic components in XML include elements, attributes, and values.

An **element** has an opening tag **<elementnamehere>** and is the most basic unit of a document. It can contain text, attributes, and other elements. The name should describe the purpose and contents. The same name is used in the closing tag (which is similar to HTML) **</elementnamehere>.**

Elements can further be divided into root elements, and sub elements.

Attributes are description within an element's opening tag. The quotation mark delimited values further describe the purpose and content of a particular element. An element can have as many attributes as needed, as long as each attribute has a unique name. Attributes are expressed in the following way:

<element attribute="value">content</element>

Values further describe the purpose and content of an attribute. Values are denoted using double quotation marks and follow an attribute separated by an equal sign.

XML Syntax Presentation
XML (eXtensible Markup Language) is called a markup language because it "marks up" language. For example, the following XML contains both **text** and **markup**. **<?xml version="1.0" encoding="utf-8"?>** **<nws-sample>** **<message>Feel free to send me:</message>** **<candy>some M&Ms</candy>**

<phone>(123)456-7891</phone> </nws-sample>
When we open this XML code in an IE browser window, it does not look much different. This is what we mean when we say “XML was designed to carry data, not to display data.”
XML documents begin with an XML declaration. The declaration provides (at a minimum) the number of the version of XML in use, but the example in the presentation contains the character encoding used in the document. The line also has code defining the parameter version, schema location, and other processing instructions.
Elements are the most basic unit of a document and can contain text, attributes, and other elements. An element has an opening tag **<elementnamehere>** The name should describe the element’s purpose and contents. Contents may be elements or text. The same name is used in the closing tag, similar to HTML. Special Case: Empty elements do not contain other elements or text. They are “empty.” These elements are only written once, and use a backslash as a closing tag (i.e. - </hr>).
Returning to the original sample code, look at several elements and their opening and closing tags. The element nws-sample contains multiple elements. In this example, nws-sample is acting as the root element. A single root element is required for every XML document. This root (or parent) element contains all other (child) elements. The syntax is the same as above. Child Elements are the sub-elements, which share a relationship with the root element. In XML, you can create nested child elements, establishing a hierarchal relationship between these individual parts. Elements must be properly nested. If you start element A, then start element B, you must first close element B before closing element A.
Attributes are a description within an element’s opening tag. The quotation-mark delimited values further describe the purpose and content of a particular element. An element can have as many attributes as needed, as long as each attribute has a unique name. Attributes are expressed in the following manner:<element attribute = “value ”> content </element>Values further describe the purpose and content of an attribute. Values are denoted using (”) and follow an attribute separated by a (=) character, as is shown above.
Keep the following specifications in mind as you edit the XML configuration files:

- XML is cAsE sENsiTive
- Comments may be added anywhere after XML declaration by using the following syntax <!- - added content - ->
- White space is ignored in XML, so you can add extra white space, including line breaks, around the elements in your code to make it easier to edit and view
- Every XML document must contain one and only one root element
- The only pieces of XML allowed outside the root element are comments, processing instructions, and the declaration.
- Closing tags are required (unless you are using an empty element).
- Elements must be properly nested. (If you start element A, then start element B, you must first close element B before closing element A.)
- Values must be enclosed in quotation marks.

1.6 CHPS XML

Configuration Focal Points must use XML to manage and define most of the configuration information used in CHPS, which control almost all aspects of operations.

At this point, you may be wondering why XML was chosen for CHPS.

There are several benefits to XML. XML is…

- Extensible: programmers can create their own tags.
- Compatible: XML is widely used, making information exchange with other agencies and academia easier. It is an international standard maintained by an independent standards' committee, the World Wide Web Consortium (W3C).
- Inexpensive: XML does not cost anything to use. It can be written with a simple text editor or one of many free XML authoring tools.
- Easy to use: XML is "human legible" and reasonable clear. Descriptive element tags identify content.
- Platform/software independent: XML is both platform and software independent. Most browsers and text editors support it.
- Separates structure from function: XML simply houses information. It relies on separate style sheets for appearance settings. This allows for greater manipulation of data and elements.

A basic understanding of HTML and JavaScript is helpful prior to working with XML.

XML Schemas

XML is the most common tool for data transmissions between all sorts of applications. The XML language (itself) has no predefined tags; however, each XML configuration in CHPS adheres to an XML Schema Definition (XSD).

XSD is the current standard schema language for all XML documents and data. XSD enables programmers to define the structure and data types for XML documents.

XML Schema Presentation
CHPS adheres to an XML Schema Definition (XSD), the current standard schema language for all XML documents and data. A **schema** defines the structure, content, and to some extent, the semantics of XML documents.

This goes beyond the basic XML specifications you learned about earlier.

An XML schema puts files into a **baseline format**, which can be ingested into the system for processing.

If an XML file correctly adheres to a schema, then it is considered a **valid instance** of the schema.

Recall if your XML adheres to the regular XML specification, then it is "well-formed". Now, you know if it adheres to the various Deltares schema, then it is "valid".

XML schema defines:

- which elements appear in a document
- which attributes appear in a document
- which elements are child elements
- the order of child elements
- the number of child elements
- whether an element is empty or can include text
- data types for elements and attributes
- default and fixed values for elements and attributes

An XML schema defines what each element must or may contain.

Do you recall the XML page begins with an XML declaration and other processing instructions?

Locating the schema is part of the **other processing instructions** statement.

- The xml schemas referenced in the CHPS XML configuration files are contained in the:
  Deltares website
  RFC local file system

A professional XML editor lets you view the XML schema to which the XML configuration file refers.

If you are not using a professional XML editor, copy the url from the configuration file into a web browser to view it there.

1.7 NAMING CONVENTIONS

Currently there is not a common or standard naming convention for CHPS configuration files; therefore, each RFC may use a different system.

Naming files according to an established pattern makes them easier to locate. Define a naming convention for your RFC's CHPS configuration files and use it consistently.

Remember, configuration of CHPS involves several staff members, and inconsistencies can result in miscommunications and errors.

Additionally, consistent use of a naming convention helps with continuity between current CHPS Focal Points and makes it easier for a new focal point to become proficient.

Here are some suggestions on keeping various versions of the same configuration file in an understandable format.

1. Add **version numbers** to XML file names to track the various copies. For example, use the following pattern to name files: **ImportGrids_3.xml**

2. Add a **default** label to differentiate between the copy in use and altered versions of a configuration file. For example: **SpatialDisplay_default.xml**

3. Combine the two previous methods: **SpatialDisplay_3_default.xml**

Other Suggestions/Techniques

Make a copy of the entire current **/Config** directory and place it in a uniquely named folder on the CHPS application directory. If disk space is an issue, zip and tar the directory to save space.

Add the date in year-month-day format at the beginning of the archived file names. This allows you to find files quickly by sorting them in numerical order.

1.8 FILE POLICY

Keeping unnecessary files makes locating files more difficult. Conversely, deleting files and asking questions later is not a sound policy either.

The first step in managing configuration files is finding out the local policy on file retention. If your office does not have a policy, contact your regional office. In the event no regional policy exists, follow the national recommendation:

According to the Hydrologic Support Branch, offices should keep patches and releases for a year, at least for now. The tar files for the builds and patches will also be available on the ftp site for about a year. Currently, the policy **only** covers install files.

Keep the patches and builds in the **/awips/chps_share/install** directory, with subdirectories named for the patch or the build (such as 35025_201102_patch.jar or OHD-CORE-CHPS-2.1.c).

Advantages of Removing Files

Removing clutter from the system has advantages for the people who maintain and troubleshoot the system and for the system itself. Getting rid of the clutter **increases** the processing speed of the system and **decreases** the frustration level of a focal point.

Personnel Advantages

CHPS Configuration Focal Points can locate the files they need more easily. CHPS support personnel can investigate FogBugz tickets and identify solutions to the problems without sorting through unnecessary files.

System Advantages

The system runs more efficiently. Tasks complete quicker with fewer files to go through.

For example, if you retain **tarred** directories of all of the patches and installs (assuming two full builds and two patches per year), you are adding over 100 megabytes (MB) every year. Not a huge amount of space taken, but what if the practice of keeping every patch and build continues for five years?

Caution! Removing some files equates to a CHPS "show stopper" - the system will NOT function!

When in doubt, do not delete a file! Ask a focal point at another office or submit a FogBugz ticket before removing a file if you are uncertain!

LESSON 2 CONFIGURATION BASICS

Configuration files affect not only the appearance of CHPS, but the entire operation of the system. Follow the standard XML editing procedure to avoid creating more work for yourself.

This lesson describes the basic configuration steps used in CHPS. By the end of this lesson, you should be able to:

- Identify configuration prep work.
- List basic configuration steps.
- Describe the process of acquiring files.
- Define the steps of editing a configuration file.
- Validate configurations changes.
- Explain how to submit files.
- Delete old configuration files.

2.2 PREPARATION

The process of configuring CHPS is straightforward; however, it does involve programming. As every good programmer knows, programming begins with a clear head and solid plan. So stop and consider...

Who	Are these changes for one use, your entire site, or could it be useful nationally? How will you communicated changes for more than one user?
What	What specific changes are needed? Which parameters are involved? What else is involved?
Where	Where can these items be located? What files and directories need to be created or modified to make the change?
When	How soon do you need these changes? How often do these changes impact your users? Does your timeline (for making updates) impact system outage?
Why	Even if it seems straightforward to you, it may be helpful to explain why changes were made in note to your team **and** in comments within the code. So, answer this question ahead of time. Why are you making the changes?
How	How do you plan to edit, validate, and update changes? Do you plan to use your SA for testing, why or why not?

2.3 CONFIGURATION STEPS

Basic configuration consists of a series of logical steps. Use the Configuration Manager to complete these steps. The Configuration Manager is an interface between your RFC's configuration on the local CHPS Master Controller and the local datastore. Basic Configuration Steps include:

1. Download the most recent configuration from the database.
2. Export the configuration from the OC to your local system.
3. Edit the files using and XML or text editor.
4. Validate changes.
5. Import the configuration from your local file system.
6. Upload the configuration to the Central Database.

The Configuration Manager is a graphical user interface (GUI) designed to simplify your role as focal point. This GUI allows you to download the current configuration from the Central Database and upload them again once you have edited them. The local datastore in your OC directory will have a copy of these configuration files and the data needed in FEWS.

Configuration Manager Profile Transcript
We talked about the configuration process… and broke down this diagram. Now... let's talk about how to use the Configuration Manager to edit and upload configuration files.
The Configuration Manager is an interface designed to help you complete the steps required in editing a configuration file. The Configuration Manager not only provides a GUI to complete multiple steps in this process... it also connects to the Central Database and your OC local datastore.
The Configuration Manager functions include: • exports/imports • tracks files • validates files • logs errors • helps with version management
Start the Configuration Manager by opening a terminal window. Use the job sheet provided at the bottom of the lesson page for detailed instructions.
Interface Overview On the left is the file tree, which shows the various groups of configuration files. The set active button is in the button bar, along with the upload and download buttons. These buttons allow you to connect to the Central Database and synchronize configuration files. Configuration files are displayed in the top portion of the main screen and log files are displayed below.

Use the following job sheet to learn how to use the Configuration Manager.

Job Sheet: Using the Configuration Manager

2.4 DOWNLOAD

Remember: You have both an OC and SA your RFC. It is highly recommended you use the SA system to create and test all changes prior to modifying your OC. The SAs are designed for testing. All information is stored locally on the SA and changes made will not affect the live system.

You have multiple methods for acquiring files for your individual instances of the SA and OC. Open a terminal window and use the simple Linux copy command, cp, to transfer the files from another directory to yours.

Acquiring Files for SAs:

You can acquire configuration files for your SA by copying files from a couple different locations:

1. From another user's base configuration files.
2. From the main copy of the office base configuration files. Note: The location of these files may be different from office to office.

   Some offices keep their base configuration files in the SA directory; some have a specific directory in the main chps_share directory, while other offices may have another location.

You should put your new files in the following directory: **/awips/chps_share/sa/YOURFOLDER**

Acquiring Files for OCs:

You can copy another user's OC directory for initial set up, but when making changes to an OC configuration, you **must** use the Configuration Manager.

Once the initial configuration files are in your Stand Alone instance, test and make changes directly to the files in your Stand Alone.

Changing the base configuration files (located in your SA directory) will NOT affect the other instances of the Stand Alone or the Operator Client as long as you only edit the configuration files located in your directory.

Note: If others make configuration changes in another Stand Alone, obtain the newer configuration files periodically to keep your Stand Alone updated.

2.5 EDIT

CHPS configuration files are written in XML. Using an XML editor is recommended, but you may also use a text editor.

The steps involved in editing an XML file include:

1. Identify and locate the files you need to modify.
2. Close CHPS before making the required changes.*
3. Make all necessary changes to the XML files.
4. SAVE your files using the established file naming convention.
5. If you are not using an XML editor, manually ensure your code is “well formed” and “valid”.
6. Restart the CHPS application.
7. Inspect the results. If you did not get the expected results, check the log files for errors. Troubleshooting is discussed in Lesson 6.

Once you are satisfied with your configuration changes and they work in the SA, upload the change to the Central Database using the Configuration Manager. This will allow the OC to receive the new configuration through its local datastore.

*If you elect to skip Step 2, you must reload the system to see your changes. Hit the F5 key while CHPS is running to reload the configuration. You may still need to restart the CHPS application.

Attention! Any edits to files in the **SystemConfigFiles** directory require a complete close and restart of the CHPS application.

2.6 VALIDATE

Remember, your XML must be **well-formed** AND **valid** in order to avoid errors in CHPS.

- "Well-formed" code complies with XML specifications, which are defined in the Extensible Markup Language (XML) 1.0 documentation maintained by w3.
- "Valid" code correctly adheres to the Deltares schema.
- Use a professional XML editor, such as oXygen, which color-codes errors.

Always test changes on a Stand Alone instance of CHPS **before** attempting to upload the files to the Central Database!

Configuration Manager Validation

The Configuration Manager checks to see if the code is "well formed" and "valid".

When the Configuration Manager finds a violation, it provides an appropriate message indicating the type of violation and the file(s) causing the violation.

If the file is validated, the icon next to the file will turn green. See the example to the right.

2.7 SUBMIT

Recall we **strongly recommended** you create and edit changes on your SA **prior** to modifying your OC. You must then follow the import procedures as described below after transferring these modified files to your OC.

The Import/Upload Process

Importing a configuration file allows the file to be saved to the local datastore. The Configuration Manager sets the filename to **NameVersionStatus.xml**.

Where:

- **Name** is the configuration file ID.
- **Version** is the version number of the configuration file.
- **Status** is default if it is the active file, a version number if it is inactive.

Synchronizing

Uploading a configuration (file) requires synchronizing all modified and added configuration files with the Central Database. The Configuration Manager provides each configuration file with a unique Master Controller ID.

Caution! Only upload files after testing them on the SA in order to prevent corrupting the Central Database!

Optimistic Locking

A procedure called optimistic locking stores the latest changes made to a configuration. If other people are working on configuration, it might not save **your** changes!

During the period between downloading a configuration and uploading a modified configuration, someone else could have made a change to the same configuration. The Configuration Manager will not deal with this possibility.

Communicate with other members of the RFC to ensure only one person is changing a configuration file, as only the last change is saved.

If an error occurs while uploading the new configuration, the configuration rolls back to the previous version.

2.8 FILE CLEANUP

The number of configuration files can accumulate quickly. Having fewer configuration files to manage makes focal point duties easier. Configuration focal points should:

- **Identify** files and directories that are candidates for deletion. Some candidates for deletion are OFS legacy products/processes, old releases and patches, unused segments, and unused forecast/model data.
- **Discuss** file deletion with other office members that help with configuration. Work with the CHPS System Manager at your office to determine what information can be removed and when. For example, consider how long you want to keep QPF grids or forecast data.
- **Follow** the office policy on archiving old files, or develop a policy if one does not exist.

Caveat: Forecasters may wish to do a case study on an event – deleting the forecast and/or model data would be a problem.

Removing Segments

Why would you want to delete segments? Imagine your office had a location, which was no longer a real-time site, but kept in the system. Years later, a gage is installed **near** the old location, so a segment for the new site was developed and eventually became a new forecast point. You would need to get rid of the old site's information so it is not confused with the new site that has data. Consider the following when deleting a segment:

- Each segment has a directory of xml files for each location.
- Segments are included in workflows, so deleting a segment is not as straightforward as deleting the segment directory!
- Additionally you need to remove the workflow mapping of the segment in the Administration Interface.

Best Practice

Start a document containing information on which files/directories were deleted, who deleted them, and when. Having this documentation is helpful whether you work alone or on a team.

Click here for a refresher of the Linux commands needed to delete unused/old files and directories (this links to an outside resource).

LESSON 3 CONFIGURE INPUT DATA

Working with input or static configurations allows you to control what data the system uses, where the data is processed and viewed, the values and units given to the data, and how various portions of data are filtered.

This lesson provides information configuring static and dynamic input data. By the end of this lesson, you should be able to:

- List types of input data.
- Configure location data.
- Configure parameter data.
- Configure units and unit conversions.
- Configure filters.
- Configure grids.
- Describe decimal options.
- Set data expiry times.
- Configure data synch levels.

3.2 DATA TYPES

CHPS uses two types of data – static and dynamic. CHPS is a location-oriented system, which means CHPS uses the static input data to orient its dynamic data. CHPS pulls in dynamic data in order to make forecasts.

Static Data

- **Location** configuration files assign segments with x, y, z coordinates along river basins where data can be processed and viewed. These locations are placed in logical groups sharing common forms of data via location configuration.
- The **grid** configuration file handles the sizes, types, and locations of grids throughout forecast basins.
  The **parameter** configuration file assigns which specific parameters the CHPS system uses. Parameters are the model variables such as stage and discharge. Other river parameters such as temperature, precipitation, reservoir storage, and elevations are also defined in this configuration.
- **Units and unit conversion** configuration files assign specified units to each of the parameters defined and potential conversions of those units between metric and English units.
- **Filter** configuration files are a way to sort out necessary and unnecessary data used or displayed at given locations.

Dynamic Data

As the name implies, dynamic data changes. Examples of dynamic data used in hydrologic forecasting are river stage, precipitation, and pool elevation. See the other sections in this lesson for information on setting how long the data resides in the system, how much precision you store the data in, and how the data is synchronized on the system.

3.3 LOCATION

CHPS is a location-oriented system. All of the time series data must be referenced to a geographic location. CHPS uses **Locations.xml** and**LocationSets.xml** to accomplish this task.

Locations.xml

Locations.xml is located in the **RegionConfigFiles** directory.

This file houses all location information. Comment lines for each forecast group usually separate the information about the sites. The data for each segment must include:

- a unique location Id,
- a shortname (5-character site Id),
- and geodatum (x, y, z coordinates).

Note: Some locations have "0" for the x, y, z coordinates. Those are placeholders, because they do not represent a single location.

Try it out! Practice Exercise: Adding Locations and Location Sets

LocationSets.xml

LocationSets.xml is also located in the **RegionConfigFiles** directory.

This file houses location set information. Location Sets define logical groups of locations.

Since there are so many locations, it is helpful to define location sets. The location sets are not arranged by region, but grouped by similar processes. This makes it easier if an action needs to be taken on a whole set of locations (such as validation). You only need to define a location set once, and then CHPS evaluates a location set as a list of locations.

Use the following job sheet to learn how to add a gage and catchment to the system.

Job Sheet: Adding a Location and a Location Set

3.4 PARAMETERS

Parameters are the forecast variables used in the CHPS system such as discharge and stage. Parameters also include information about river basins and land data such as reservoir storage, elevations, temperature, precipitation, etc.

All of the parameters used in CHPS are registered, defined, and grouped together logically in the **Parameters.xml** file located in the **RegionConfigFiles** directory.

You typically will not be doing much modification of this file, since all of the necessary parameters used by the forecasters have already been registered. It is important to be familiar with the general contents of files as all available parameters in the CHPS system are referenced.

3.5 UNITS

CHPS uses metric units. However, units can be converted while importing, exporting, or displaying.

The conversion of the units occurs due to coding in xml files within the **UnitConversionsFiles** directory.

Conversions use basic coding. For example...

A typical unit conversion when going from English units to metric units is feet (FT) to meters (M). In order to do this conversion, apply a simple dimensional analysis: 1 inch = 0.0254 meters, and there are 12 inches in 1 foot; therefore, 1 foot = 0.3048 meters.

To do the same conversion in CHPS, specify the input unit type (i.e., feet, FT), the output unit type (i.e., meters, M), and the multiplier (i.e., 0.3048). Also try adding an incrementer if necessary (in this example, there is no need for an incrementer).

3.6 FILTERS

Filtering the available information in CHPS makes it easier to find what you need to make forecasts.

As a focal point, try limiting and defining the locations displayed on the main map display and the locations and parameters displayed in the list box using filters.

Filters are organized in folders with child filters. Filters arrange the locations with associated parameters in logical groups. Each filter is defined as a collection of times series sets.

Configure filters in the **Filters.xml** file located in the **RegionConfigFiles** directory.

You may never need to edit **Filters.xml**. The important thing to remember is it exists and you have the ability to add or remove filters should the need arise.

3.7 GRIDS

Since CHPS is a location-oriented system, all time series data must be referenced to a geographic location. For gridded time series data, you must reference each point in the grid to a location in a grid structure.

Grids may be **regular** or **irregular**.

- **Regular** grids have cells with the same width, height, and area within the coordinate system specifying it.
- **Irregular** grids have a fixed number of rows and columns, but the cell height and width is not equal in each row and column. Additional information is required on the location of the center of each cell in order to display the grids and for use in the spatial interpolation routine, which interpolates data in between cells.

Grids.xml is located in **RegionConfigFiles** directory.

3.8 DECIMAL

After you add a new data source, it is a good idea to check the precision of **all** the data stored in your system.

By default, data is stored to eight decimal places, but observed data is typically imported out to two or three decimal places!

Hint: Reducing number of decimal places significantly reduces the size of the Central Database.

Storing Decimal Values

Navigate to the **RegionConfigFiles** directory and locate the**Parameters.xml** file. Check the valueResolution tag for each type of data (QINE, temperature, etc.). The document has tags for both the format in which it is **stored** and the format in which it is **displayed**. The document also contains a conversion factor.

This tag is just an "FYI" to the system. It does not change the display characteristics. See the next section about the **TimeSeriesDisplayConfig.xml** for display characteristics.

In this example, QINE is stored in CMS, but will display in CFS.

Suggestion: Compress precipitation grids instead of temperature grids. Many of the precipitation grid points have the same value (think 0.00). Compressing temperature grids causes data loss because there is more variability among the grid points.

Displaying Decimal Values

The **Parameters.xml** file does not control the way it appears in the displays – change the precision in the **TimeSeriesDisplayConfig.xml** (located in the **SystemConfigFiles** directory).

Use the following job sheets to change the way decimal values are stored and displayed in your system.

Job Sheets: Reducing Decimal Values Stored in Database | Reducing Displayed Decimal Values

Reference: *See the Deltares wiki for more information on data compression.*

3.9 EXPIRY

In addition to changing the precision of data stored in the system, another option for limiting the amount of data is changing the expiry times. Expiry times establish database retention times. Longer expiry times mean the data will be kept in the system longer. Setting longer expiry times leads to larger local datastores and longer data processing times. Expiry times are defined as followed:

- **Unit:** Second, Minute, Hour, Day, Week
- **Multiplier:** Defines the number of units given above
- **Divider:** Defines fraction of units

A process, called the MarkedRecordManager, labels the data that needs to be removed by the Rolling Barrel. The Rolling Barrel task removes the data once it reaches its expiry time.

Set expiry times for workflows by editing **WorkflowDescriptors.xml** in the **RegionConfigFiles** directory.

Use DbVis to search by expiry times and see what is marked for deletion at a certain time. Open the DbVis interface (click here for more information on DbVis) and use the SQL query: **Select <table> WHERE expirytime='YYYY-MM-DD HH:mm:SS'**

Options for Optimizing

The Configuration Focal Point can decide which information can be assigned shorter expiry times. The default time is set at 30 days. Many time series, however, are only needed for a few model runs. Work with your RFC to identify data stored in the database too long.

Example: ESP (Extended Streamflow Prediction) runs generate a large quantity of data, most of which is only needed for the forecast runs. Try marking the ESP data for deletion soon after the forecast period. Schedule ESP runs early in the morning before the forecast period, keep the data for 5 or 6 hours, and mark the expiry time for shortly before the Rolling Barrel task. The Rolling Barrel then marks the data for deletion, and it does not needlessly synch.

For details on setting expiry times, use the job sheets below.

Job Sheets: Changing Expiry Times Using XML Editor | Changing Expiry Times for Scheduled Tasks

3.10 SYNCH

Data is shared throughout the system in a process called synchronizing, or synching as it is more commonly called.

Synching prevents performing duplicate actions and matches data throughout the CHPS system. The synching task can be very resource-intensive, especially when there is a longer time between synchs.

What are synch levels? Synch levels are integers assigned to each task to denote the frequency they are synched. The synch levels are defined in the chart to the right.

How do synch levels work?

Synch levels categorize the data going in and out of FEWS. When a synchronization task runs, the types of data synchronized depends on the level given to each time series in the database. Therefore, a time series with a synch level of 4 will not synch as often as a time series with synch level 0. Synch level 4 is used for astronomical and climatological data where level 0 is used for all scalar data from a forecast run.

Where are synch levels defined?

Synch levels are established when time series are set up in the system. Synch levels are defined in some of the module configuration files and in as the **synchProfiles.xml** files. The**synchProfiles.xml** files are found at the following paths:

/awips/chps_share/oc/xxrfc_oc/

/awips/chps_local/fss/rfc_name/FSS##/FewsShell/rfc_name

Caution! Do **NOT** edit the **synchProfiles.xml** file without consulting with the Hydrologic Support Branch.

The **synchProfiles.xml** files are created by Deltares and cannot be changed. If your RFC is synching unused data, it is possible to request a custom **synchProfiles.xml** file or request edits to the original file. Making unauthorized edits cannot only cause errors in the system; the changes may be lost with the installation of a new patch or build.

LESSON 4 MODULES

Refining the forecast process makes the system run quicker and makes it easier for the forecaster to create accurate forecasts. Modules are the activities in CHPS you can configure to achieve those goals.

This lesson provides details on configuring CHPS modules. By the end of this lesson, you should be able to:

- Define modules.
- Identify specific modules.
- Edit module configuration files.
- Configure modules using a multi-file process.
- List opportunities for optimizing modules.

4.2 MODULE DEFINITION

Modules are the activities or operations performed in CHPS. Modules appear within workflows (which are discussed in Lesson 5) and have a standard layout.

The Standard Layout
Most modules have a standard structure similar to the one illustrated below.

1. The processing begins with a request of a specific set of data from the database as one or more input time series sets. Click here for more information on time series sets.
2. The data is transformed using the module's functional items (i.e. interpolation, import, etc.).
3. One or more output time series sets are used to store the data in the database under a unique id.

Note: The unique id, a unique module instance name, is defined by an instance of a module.

Module Descriptors - Module Descriptors tell CHPS the module name and how it will run. Java classes implement the workflow plug-in interface to CHPS. The Module Descriptors **define** what Java classes will run each module and **assign** a recognizable name for the module. Define the modules in the **ModuleDescriptors.xml** file in the **SystemConfigFiles** directory.

Module Instances

Now that you assigned a name and defined the Java class to run the module, you need to further specify what you want the module to do and what data to use. The module instance files are stored in the **ModuleConfigFiles** directory. Module Instances must include: input data, output data, data processing steps, and the unique module instance name (from the **ModuleDescriptors.xml** file).

The Link - All modules must be present (registered) in the **ModuleInstanceDescriptors.xml** in order to run! The file, located in the **RegionConfigFiles** directory, links instances of the module in a workflow to a registered module type.

Module Instance Sets (optional) - Reduce the number of calls to module instances by grouping similar module instances into **Module Instance Sets.** Group Module Instances either by location or module type. The **ModuleInstanceSets.xml** file is located in the **RegionConfigFiles** directory.

Reference*: For more details on configuring modules, please refer to the Deltares DELFT-FEWS Configuration Guide Configuring Available DELFT-FEWS Modules.*

4.3 MODULE TYPES

While there are several modules built into CHPS, there are five commonly used modules.

Import - The import module allows the import of data from external sources into CHPS. Import modules are listed in the ModuleDescriptors.xml file in the SystemConfigFiles directory. The modules are named for what they do, for instance, a TimeSeriesImportRun would import time series from various formats into a CHPS-readable format. Data is imported from specified directories in the ModuleConfigFiles directory. Data conforming to the expected format is imported. Data not in a readable form moves to a directory for failed imports.

Export - In order to support the "C" in CHPS (community), the export module is configured to format time series data for use in other systems. For instance, the export module corresponding to the import module in the previous slide, is called TimeSeriesExportRun and would export CHPS data to various formats. The configuration specifies the following parts:

- General – specify the file name, data type, etc.
- Metadata – export specific settings
- Time series set – actual data to export

Transformation - Transformation is a general-purpose module allowing for generic transformation and manipulation of time series data. Examples include simple arithmetic manipulation, applying stage-discharge relations, and shifting time series in time (CHANGET).

Example: Stage observations are in one-hour time steps, but RFC models typically use a six-hour time step. The CHANGET function transforms input time series data to a specified interval.

Interpolation - CHPS uses the Interpolation module to alleviate the problems with data sparse areas and lack of observational data at needed time steps. The two types of interpolation with this module are **spatial** and **serial**. **Spatial** interpolation fills in gaps and derives spatially distributed data by utilizing information from neighboring locations. **Serial** interpolation takes the available observations for a location and fills in the missing observations.

Example: The 12 UTC observations are missing, but the 09 and 15 UTC observations are available. The interpolation module makes an "educated guess" for 12 UTC.

General Adapter - The General Adapter (GA), or "translator" module, allows CHPS to run external models. The GA is responsible for data exchange with the modules and for executing the modules and their adapters.

4.4 CONFIGURATION PROCESS

Now that you know what type of information is defined in module configuration files, it is time to learn how to configure a module from start to finish. Module configuration involves more than one file and more than one directory. The module configuration process includes:

1. Create a module instance configuration file in the correct segment subdirectory located in the ModuleConfigFiles directory.
2. Register the module instance configuration file by editing the ModuleInstanceDescriptors.xml file in the RegionConfigFiles directory.
3. If the module has external parameter values, place a copy of the XML configuration file in the ModuleParFiles directory.
4. If the module has external data sets, place a copy of the ZIP files in the ModuleDataSets directory.
5. Ensure the module is registered in the ModuleDescriptors.xml file in the SystemConfigFiles directory.

Note: if you want to group module instances, the module instances must to be added to the specific module instance set to which it pertains.

Time Saver Hint: Often you can locate a module configuration, parameter, or data set file similar to what you need in the **ModuleConfigFiles**,**ModuleParFiles**, or **ModuleDataSets** directories.

Save yourself some time and work by copying and modifying these files instead of writing your new file from scratch.

Rename it and put it in the correct subdirectory when you are done.

Try it out! Practice Exercise: Adding a Transformation Module

4.5 MODULE OPTIMIZATION

What if you have a situation where you only want to manipulate a time series in a given situation? Rather than making two workflows and only running one of them, you have the option to apply a conditional module instance in the original workflow.

The use of conditional modules can optimize the forecast experience, as it allows for more accurate dissemination of data for that scenario. While other types of modules could be applied conditionally, the best example is the use of a transformation module.

Examples:
1. Discharge from locks and power-generating reservoirs
2. Seasonal variation (Example: winter months November through March)

Conditional Transformation by Range

Conditional transformations use one of two types of triggers – **Range** or **Periodic**. Use the range trigger (because of the variability of the flow) for conditional transformations of discharge from locks or power-generating reservoirs. Using a range as a trigger means a variable has to reach a certain value within the range defined.

Conditional Transformation by Period

Periodic transformations are triggered by a set date and/or time. Once the current date or time meets the condition, the transformation will be applied. The module will continue to be applied until the date or time has passed the ending date set in the workflow. Winter is an example of when you might apply a periodic conditional transformation.

Think Tank

Are there other ways you could apply a module in a conditional situation? Chat with some of the team members in your office and come up with ideas of when to apply conditional modules and what type. Keep in mind that you must have a way to trigger these modules during that scenario. Use the module schemas found on **CHPS 1** in the **/awips/chps_local/schemas** directory or look up the schemas on the Deltares Wiki for guidance.

Use the job sheet below to set up a conditional transformation.

Job Sheet: Creating a Conditional Transformation

LESSON 5 WORKFLOWS

While modules are the activities and operations, workflows are specific tasks (often containing modules) executed for the forecast group. You must configure workflows properly in order to complete a forecast.

This lesson provides information on configuring CHPS workflows. By the end of this lesson, you should be able to:

- Define workflows and workflow components.
- List the workflow directories.
- Describe the process of configuring workflows.
- Add a module to a workflow.
- Use the Workflow Navigator.
- Clean up code in existing workflow configuration files.
- Optimize user experience with workflows.

5.2 WORKFLOW DEFINITION

Workflows identify specific tasks and outline the order in which they are completed. Modules are the activities or operations. Together, workflows and modules provide an operational outline of which operations to complete.

All functionality used by CHPS in processing dynamic data and running external forecasting models is configured in module instances. These instances execute in a logical sequence, which is defined in a workflow.

Activities

Each task in a defined workflow file is called an activity. Single or multiple activities may be contained in the workflow. Different types of activities include sequences of modules, nested workflows, and/or ensemble runs.

Files

All workflows must be registered in the**WorkflowDescriptors.xml** file in the RegionConfigFiles directory in order to process the activities. Entries in the file define the specific behavior of the workflow. Each workflowDescriptor id defined in **WorkflowDescriptors.xml** has a corresponding XML configuration file in the WorkflowFiles directory.

Sequences

Workflows activities are processed in the order in which they are listed in the configuration. Since the activities process in a logical sequence, workflows can assist with troubleshooting if the results are not as expected. Workflows contain calls to module instances and other workflows.

Water Cooler Break! Take a few minutes to explore a workflow configuration file. Learning to configure new or existing workflows requires practice and exploration of the files!

Reference: For more details on configuring workflows, please refer to the *Deltares Delft-FEWS Configuration Guide*.

5.3 WORKFLOW DIRECTORIES

The **WorkflowFiles** directory organizes workflow configuration files by the processes they handle. This directory contains a subdirectory for **System + Preprocessing** and a **separate** subdirectory for **forecast groups.** Segment level items are included in the forecast group subdirectory.

The **System + Preprocessing** subdirectory contains workflow files to handle the sequencing of tasks:

- system wide processes such as Rolling Barrel
- importing/exporting such as scalars, grids, and ratings
- river forecast area preprocessing
- forecast group workflows

The names are very straightforward. For example, a file designed to sequence preprocessing activities includes "preprocessing" in the name.

Click here if you would like to view a list and description of **System + Preprocessing** Directory contents.

A **forecast group** subdirectory contains forecast group and segment level workflow files to handle the sequencing:

- forecast group update states
- forecast group preprocessing
- segment workflows
- module activities within a segment
- segment update states
- segment preprocessing

Again, naming conventions are straightforward with "ForecastGroup" and "Segment" used as distinguishing identifiers.

Click here if you would like to view a list and description of forecast group directory contents.

Attention! Each segment in the forecast group has EACH of the following files: **Segment_UpdateStates.xml, Segment_Flow_Forecast.xml**, and**Segment_Forecast.xml.**

5.4 CONFIGURE WORKFLOWS

Workflow configuration files are named for their location and the process the module handles.

Example: CHRN6_Flow_Forecast is a workflow file for the site with a 5-character id of CHRN6 with a segment level activity of a flow forecast.

Workflow Contents

Workflow configuration files can contain the following elements:

- A **moduleInstanceId** is required only if the activity is in a module.
- **workflowId** is required if the activity is a workflow.
- **fallbackActivity** is an optional activity to run in the event the primary activity fails.
- An **ensemble id** is required if the activity is an ensemble.
- **runIndependent** is a flag indicating whether the activity is independent of other workflow activities.
    - If set to “true”, and the item fails, the workflow will not fail.
    - If the flag is “false” and the activity fails, the workflow will fail.

Basic Steps

1. Create a workflow file in the Workflows directory.
2. Add the workflow to the **WorkflowDescriptors.xml** file in the **RegionConfigFiles** directory.
3. Inspect the results in the Workflow Navigator.

Use the following job sheet for instructions on how to add a workflow.

Job Sheet: Creating a New Workflow

5.5 ADD A MODULE TO A WORKFLOW

Earlier in this lesson, you learned the process for creating a new module for an existing segment. In order for CHPS to process the added module, add the module to a workflow configuration file.

Recall the actual workflow configuration files are contained in the **WorkflowFiles** directory.

Therefore, when creating new modules, you will typically need to complete the following steps.

Steps	Actions
1	Create the module instance in the **ModuleConfigFiles** directory.
2	Register the module instance in **/RegionConfigFiles/ModuleInstanceDescriptors.xml** file.
3	Verify the module is listed in **/SystemConfigFiles/ModuleDescriptors.xml**.
4	Add the new module to a workflow file.
5	Add the model display for the forecast group to the **/SystemConfigFiles/DisplayGroups.xml**.

5.6 WORKFLOW NAVIGATOR

The Workflow Navigator shows all of the registered workflows (set to "visible") for the entire forecast area and its associated structure. The workflows and nested activities follow the hierarchal sequence mentioned previously.

Workflow Navigator Tour

The Workflow Navigator shows all of the registered workflows (set to "visible") for the entire forecast area and its associated structure. The workflows and nested activities follow the hierarchal sequence mentioned previously.

The Workflow Navigator window within the Interactive Forecast Display GUI shows all of the registered workflows for the entire forecast area and its associated structure. The workflows and nested activities follow a hierarchal sequence ensuring an optimal pattern.

The highest level of sequencing is at the System and Preprocessing level. Here the ABRFC_Forecast is the main workflow. Its activities include: nested workflows for importing data, then preprocessing activities at the RFC level, followed by each forecast group workflow. The workflow configuration files associated with these sections are found in the “System + Preprocessing” directory.

The next level down in the workflow sequencing is the forecast group level. Here the forecast group nested workflows contain all of the activities for each segment in the particular forecast group. The forecast group level sequencing follows a similar pattern to the system and preprocessing level. The Forecast group activities for the NMWTX_Forecast workflow begins with the preprocessing nested workflow at the forecast group level followed by the ordered segment nested workflows. The workflow configuration files associated with these sections are found in the individual forecast group subdirectories.

The final level in the workflow sequencing is the segment level. Here each segment has its own forecast workflow file where the activities include the ordering of the segment level nested workflows. Each segment has a flow_forecast workflow file with activities including the modules to be run at that segment. The workflow configuration files associated with these sections are found in the individual forecast group subdirectories in which the segments are grouped.

Updatestates workflow files at the system and preprocessing level, the forecast group level, and the segment level are not shown in the Workflow Navigator. However, you will see these configuration files in the WorkflowFiles subdirectories. These updatestates workflow files are used to gather the latest warm state values for time series data.

When it comes to Configuring workflows, you as a focal point, will most often be completing the following tasks: Workflow Optimization or rearranging activities in workflows to speed up the system, adding and/or removing modules to existing workflows, and adding/removing workflows in accordance with optimization activities.

Maintaining the CHPS workflow configuration files is key to keeping CHPS running as efficiently possible and the system up to date as new sources of data become available.

Use the following job sheet to learn how to use the Workflow Navigator to look into workflows and view associated graphs.

Job Sheet: Using the Workflow Navigator

5.7 WORKFLOW DESIGN

Designing workflows more efficiently helps CHPS complete tasks more quickly, and with less strain on the CHPS hardware. The presentation below provides information on designing workflows for optimal performance.

Workflow Design
As you create configuration files, keep the code clean. It is also an opportunity to check existing configuration files.
Cleaning up and compacting code can make the system work more efficiently. Having clean, elegant code also makes it easier for other editors to make changes and spot errors.
Start with the files with the biggest cost-benefit ratio. You will get the best return for your effort in preprocess workflows (and its module instances). Could you reduce the number of module files? Are there unnecessary data processing steps?
No standard list of steps exists to clean up code within configuration files. Work with others at your office to determine what aspects would provide the most benefit.
XML editors are helpful when cleaning up code. It will automatically add the end tags, display the code in colors, and validate the code against the schema. Remember, if you make changes to a configuration file, you need to determine what other configuration components use these files. Also, if you choose to remove an old segment, un-map the associated workflows.
Think of the way your RFC processes information, executes workflows, etc. Consider different ways of organizing the information. Is there any way to make the process more efficient?
In NWSRFS, each segment had its own “instruction file”, which made the transition to CHPS easier, but it was not efficient. In addition to each segment having its own instruction file, everything was organized by basin, allowing forecasts to be run segment by segment.
Hierarchical processing is more efficient. This option defines the operations belonging to a basin and nests groups of basins. In other words, it processes the information at each of the following levels – RFC, forecast group, and segment.
Database size depends on how many grids and time series are saved, how many forecasts are issued per day, and how long the data is kept. Think about decreasing the volume of data by evaluating the expiry time of less important grids or time series, and reducing the number of decimals for data values. Some RFCs have saved some processing time and ended up with cleaner code by evaluating how data is processed.
Calculate the amount of processing time saved by optimizing the configuration code by doing the following: 1. Run the task and write down the processing timestamp in the log file (Time A). 2. Make the changes to your task. 3. Remove the local datastore. 4. Re-import the data and check the processing time (Time B). 5. Subtract Time B from Time A to obtain the time savings (in milliseconds).
Mistakes in configuration can lead to unforeseen problems. Check the log messages when validating the code, the feedback is typically verbose enough

for you to locate the problem.
Suggestions Do not do “batch” changes to configuration files. Make changes one at a time and check the result. **Hint:** Comment out the existing code and make the changes. If you need to revert to the previous code, it is still there. **Reminder:** Document the changes you made in a log, and within the document. Use the option of adding comments in XML to explain the changes.

Use the following job sheet as a checklist guide for structuring modules and workflows to run more efficiently.

Checklist: Designing Workflows

5.8 CONDITIONAL WORKFLOWS

CHPS can be configured to help the forecasters in many ways. One way to improve user experience is to create conditional workflows. Conditional workflows run when triggered and could provide data for various forecast scenarios.

Conditional workflows run if triggered by a threshold or window around a peak moment of a long time series. Conditional workflows are very effective when hindcasting, performing model comparison, or for more efficient forecasting of atypical situations.

Example: A segment is at bank full so any additional precipitation or upstream flow will cause overland flooding. Set up a workflow to run after the segment reaches flood stage. The workflow could update the forecast more often or pull in data for the segment more frequently.

Set Up - Setting up conditional workflows is very similar to setting up daily workflows. The basic steps include:

1. Create additional workflows for conditional situations.
2. Register the workflow.
3. Define the thresholds or window to activate the workflow.

Steps one and two follow standard workflow creation procedure. The hierarchical arrangement of workflows creates opportunities for system optimization. Rearranging the workflows speeds up the system.

Try it out! Practice Exercise: Conditional Workflows

Activate the Workflow - The last step in the procedure creates a file to activate the workflow. The file, called **WorkflowLoopRunner.xml** is located in the **RegionConfigFiles** directory. Define the following parameters:

- Trigger option
- Trigger time series
- Relative view period
- Value option
- Value
- Relative run window

The **WorkflowLoopRunner.xml** file follows the schema workflowLoopRunner.xsd. Use the schema to determine useful options for your RFC and create conditional workflows for various situations your forecasters encounter.

Use the following job sheet to learn how to develop a conditional workflow.

Job Sheet: Creating a Conditional Workflow

LESSON 6 TROUBLESHOOTING

A basic understanding of troubleshooting techniques helps you narrow down potential issues when CHPS is not functioning properly.

This lesson provides information on troubleshooting and report CHPS issues. By the end of this lesson, you should be able to:

- Follow general troubleshooting techniques.
- List types of errors.
- Describe troubleshooting steps.
- List examples of debugging an issue.
- Set log files to debug mode.
- Locate and interpret log files.
- Use CHPS applications to troubleshoot.
- Report issues to FogBugz.

6.2 TECHNIQUES

Troubleshooting is a **logical** and **systematic** search for the source of a problem.

Following steps helps eliminate more common errors before beginning more complex investigations.

Work smarter, not harder!

This becomes even more critical when we consider there are many complex relationships between the user, the computer, the program, and the servers, AND multiple failures and causes are possible.

This page identifies three techniques, which may assist you before and during troubleshooting.

Technique 1: Maintenance

Everything begins with maintenance. As a CHPS Focal Point, regular maintenance should include:

- Reading release notes after each install to be familiar with software change.
- Ensuring no specific user has unneeded or illegal files.
- Removing obsolete or unnecessary files and obsolete entries from configuration files.

Technique 2: Communication Strategy

The second critical area is communications.

If more than one person in your office performs any of the regular maintenance listed above, create a plan to document changes made to the system. Documenting system changes are also useful if you work alone. You might not always be able to recall your latest change when you are troubleshooting.

This type of preparation may significantly shorten the time spent troubleshooting and correcting errors.

Technique 3: Work from Common to Rare Possibilities

Once an issue has been identified and isolated, begin diagnosing the situation.

Start with the most probable and common issue, and progress to the least probable and uncommon issue.

6.3 ERRORS

Recall the chart from the last page, notice there is more than one type of error. You may need to troubleshoot user errors, machine errors, or server and program errors. Here are some tips for handling the various types of errors.

User Errors

The most common and probable issue with computers is generally a user error. Indicators of user errors include but are NOT limited to:

- configuration changes not taking affect
- missing files or changes

In most cases, typographic errors, incorrect syntax, or misplaced code, cause configuration errors. To fix the problem, **fix the configuration file!**

Machine Errors

As you are already aware, computers do not live forever. At times, machines begin to fail. Machine failures are easy to discover; an error occurs on only one workstation. All of the other workstations are functioning properly.

Try recreating the issue on multiple workstations.

- If all others are working correctly, the problem is most likely related to the machine and not more pervasive.
- Reminder: Use the Stand Alone to perform the operations before using in Operator Client mode.

Server and Program Errors

Like most software, CHPS is updated periodically. These installs can range from a small patch, to a large install requiring several hours of down time.

The main indicator of a software problem is you are encountering errors **when no configuration changes have been made**. Most of the time, the error is in non-configurable software.
Note: Software installs are typically **schedule-driven** rather than on-demand, so offices may have to deal with issues caused by a software bug for days or even weeks.

6.4 TROUBLESHOOTING STEPS

The previous section outlined general troubleshooting steps. Here are some Graphics Generator specific steps.

1. Ask questions to identify or describe a problem.
    - What is the problem affecting; the interface, output, etc.?
    - What are the symptoms or indicators? Look in the Log panel for information.
    - Can the problem be recreated?
    - Is it an intermittent or constant problem?
2. Use the following questions to isolate the issue.
    - When did the problem begin?
    - Were there any software changes to the system recently?
    - Were template changes uploaded recently?
    - Could the issue be Pi-Service related?
3. Diagnose the situation.
    - Is it a known issue? Check the FogBugz site and search for similar problems.
    - Research known issues on RFC listservs.
    - Use information from this course.
    - Locate the Graphics Generator documentation.
4. Determine a course of action.
    - Document the steps you take (optional, but helpful).
    - Restart the application, is needed.
    - Report the problem on FogBugz.
    - Share what you learned with others.

6.5 CONFIGURATION DEBUGGING

You will occasionally encounter errors as you modify configuration files with CHPS. Errors may occur even if modifications are well formed and valid against the file's XSD schema.

These types of errors typically will **NOT** prevent the CHPS application from starting. Rather, logical errors may prevent the modification from executing the desired change.

Example

Scenario: You used a professional XML editor (so we know the code is**well formed** and **valid**) to add a time series set for a particular segment in the **DisplayGroups.xml** file in the correct location. You included all the necessary required elements but happen to **misspell** the**LocationId"CRXY4"** when it is meant to be **"CRYM4"**.

When you reopen the CHPS application and go to see your new time series plot, you notice the **data is missing**.

Solution: You check the log panel to discover there is an **error**. The log panel entry says there is an error in DisplayGroups.xml and the **specified** LocationId is **invalid** and **time series data cannot be found**.

You reopen the **DisplayGroups.xml** file and discover the misspelling. You fix the error, save the file, reopen, or reload the CHPS application to find the error has been addressed (e.g., there is no longer an error in the log panel, and the time series plot is displaying properly).

As a focal point, it is your responsibility to seek out and fix these types of errors. Thankfully, CHPS has multiple built in tools for debugging these errors.

6.6 LOG FILES

CHPS automatically generates log files. These log files document user actions, and program/server interactions.

Log files are displayed in the IFD, but available in user-configured locations.

Log files are also located in each instance of the OCs, SAs, and FSSs and can be accessed through the Admin Interface.

Look through these log files when re-running a configuration to detect errors.

When no errors are reported (or all errors are acknowledged), the icon in the lower right is green.

When there is an error, a message is displayed in the log panel providing some information as to what the error is and where the error is occurring.

Use the following job sheet to learn how to debug your configuration using logfiles.

Job Sheet: Debugging Using Logfiles

6.7 DEBUG MODE

While log files give some indication as to what the error is and where it is occurring, the amount of information is limited.

Configure Debug Logging

Debug mode produces more verbose log messages than the default mode. Anyone can use debug mode once it is configured by the focal point. The debug mode setting is described in the **Log4jConfig.xml** file, located at the following path:

/awips/chps_share/oc/ /xxrfc_oc

The option for priority value can be set as either "INFO" or "DEBUG". Info is the standard setting and produces the default style of log messages.

Caution! Set the value back to "INFO" after locating the problem. Leaving the logging in "DEBUG" mode results in HUGE files!

Running a Workflow in Debug Mode

Debug mode is good for troubleshooting, but may not be the best setting for daily use. When using debug mode, the log window in the Interactive Forecast Display (IFD) can quickly become difficult to read. To change the output in the log text menu, edit the Log4J Additivity section of the **Log4jConfig.xml** file.

Note: Debug mode log messages use available log memory more quickly. Debug mode can also slow down the system because of the extra time to make detailed messages.

For issues with a single workflow not completing, use debug mode for that workflow's log messages. When in the FEWS interface, click the F-12.

For details on setting log messages to debug mode, setting the Log4J additivity, and using the F-12 key to debug, see the job sheets below.

Job Sheets: Setting Logs to Debug Mode | Changing Message Appearance in IFD

6.8 TOOLS

CHPS has several built-in tools to help you troubleshoot errors.

Admin Interface
The Administration Interface (AI) provides options for viewing, purging, and downloading log files. Contact your office's CHPS System Manager for information about the AI usage policy.

System Monitor
Recall from Introduction to CHPS, the System Monitor is available through the IFD. This interface also displays log files and breaks down the components into columns. System Monitor LogsTabs on the interface allow access to information on system status, scheduled forecasts, synchronization status, import status, and more. Focal Points can use the interface to monitor logs and see if tasks are failing (in some cases, a sign of a configuration issue).

Workflow Navigator
An additional interface available through the IFD is the Workflow Navigator. In Lesson 5, you learned about how the Workflow Navigator can show you the modules and segments in a workflow.

You can also use this interface as a troubleshooting tool. The Workflow Navigator can highlight configuration errors and provide access to time series data produced in a run when analyzing problems in your configuration.

Detailed information on how to use the Workflow Navigator to troubleshoot is in the job sheet below.

Job Sheet: Using the Workflow Navigator

6.9 REPORTING

You located the problem in the system, checked resources for a solution, but could not find one. Now what do you do? Report the problem on FogBugz.

FogBugz

FogBugz is a bug tracking system built by Fog Creek Software. You can learn more about the company at their web site, Fogcreek.com.

FogBugz provides a platform to report problems and receive consistent feedback with the RFC Support Group and other RFCs. You can also browse the cases to see if another RFC had a similar problem.

The image below is an example of what the new case form looks like on the web site. Using FogBugz could save you time and provide quick solutions to issues.

For more information on how to report a problem using FogBugz, use the instructions in the job sheet below. The FogBugz site also has links to OHD-created materials.

Job Sheet: Reporting Problems on FogBugz

COURSE SUMMARY

Introduction

- The CHPS Configuration Focal Point has many responsibilities that includes configuring new sites, adding data from new sources, configuring modules and workflows, and overall maintenance of CHPS configuration.
- CHPS configuration files are in XML format. XML stands for eXtensible Markup Language.
- XML contains elements, root elements, sub (child) elements, values, and attributes.
- The main set of CHPS configuration files are located in the **/Config** directory. All of these directories are synchronized.
- While there is no required naming standard, as a configuration focal point, you should follow a consistent naming convention to make managing configuration files easier.
- As with the naming convention, there is not a required file retention policy. The Hydrologic Support Branch, however, suggests keeping patches and releases in the **/install** directory (located in **chps_share**) for at least one year after the release.

Configuration Basics

The Basic Steps to configurations are:

1. Download the most recent configuration from the database.
2. Export the configuration from the OC to your local file system.
3. Edit the files using an XML or text editor and validate changes.
4. Import the configuration from your local file system.
5. Upload the configuration to the Central Database.

To acquire a copy of the latest configuration files for your SA, you can copy another user's up-to-date files using simple Linux commands or copy the office's main copy of the base configuration files. To acquire a copy of the latest configuration for your OC, download the configuration into your local datastore.

Once you acquire files to edit, make sure to close CHPS before making changes. Once the changes are made, be sure that the code is "well formed" and "valid". Then restart CHPS.

Once your changes are made you upload your file(s) to the Central Database using the Configuration Manager. Make sure to set the correct version active and to use a consistent naming scheme for version control.

As creating and maintain configuration files is part of the focal point duties, deleting files is part of the job as well. Focal points should identify candidates for deletion, discuss with office members, and follow/create office policy on archive and deleting files.

Configure Input Data

CHPS uses two types of data -- static and dynamic data. Static data includes location, parameter, unit and unit conversions, filters, and grid data. Dynamic data includes temperature and precipitation.

Static

- Location data references the time series data to a geographic location. All of this information is stored **Locations.xml** and **LocationSets.xml**.
- Some static data require parameters, or forecast variables, to be defined, such as discharge or stage. This information is located in the **Parameters.xml** file.

- Some units may need to be converted as it is imported. The conversion of units occurs due to coding within the **UnitsConversionFiles** directory. For example, the**ImportSHEF.xml** file.
- Filters make it easier to find what you need to make a forecast. You can organize filters in the **Filters.xml** file located in the **RegionConfigFiles** directory.
- Use the **Grids.xml** file to reference each point in a gridded time series to a location in a grid structure.

Dynamic

- When dealing with dynamic data, use the **Parameters.xml** file to define how many decimal places to store the data in and **TimeSeriesDisplayConfig.xml** for display characteristics. Reducing the number of decimal places reduces the size of the database.
- Expiry times determine how long the data remains in the database. Only keep data as long as it is needed. Smaller databases mean faster processing times and a more efficient CHPS system.
- Synch levels are assigned to data to determine how often it is synchronized. Some synch levels are synchronized more frequently than others. Some synch levels are not synchronized at all.

Modules

Modules are the activities or operations performed in CHPS. There are several modules built into CHPS, but some of the most commonly used (and arguably the most important) include: Import, Export, Transformation, Interpolation and General Adapter Modules.

The standard structure of modules are:

1. Request data from the data store
2. Do something...
3. Write data to the data store

The basic steps to configuring a module include:

1. Create a module instance configuration file in the ModuleConfigFiles directory.
2. Register the module in the ModuleInstanceDescriptors.xml file.
3. Register parameters in ModuleParFiles (if needed).
4. Register external datasets in ModuleDataSets (if needed).
5. Ensure module is registered in ModuleDescriptors.xml file.

One way you can optimize CHPS using modules is to create conditional modules such as a conditional transformation module. These modules only run given a specific incident that would trigger the module.

Workflows

Workflows identify specific tasks and outline the order in which they are completed.

Workflow configuration files are located in the **WorkflowConfigFiles** directory and are broken up into System+Preprocessing and forecast group subdirectories.

Configuring a workflow includes:

1. Create a workflow file in the Workflows directory.
2. Add the workflow to the **WorkflowDescriptors.xml** file in the **RegionConfigFiles**directory.

3. Inspect the results in the Workflow Navigator.

Adding a module to workflow requires following the same steps outlined in Lesson 4 to creating new module. Follow those steps with:

1. Add the new module to a workflow file.
2. Add the model display for the forecast group to the **DisplayGroups.xml** file located in **/SystemConfigFiles**.

The Workflow Navigator shows all of the registered workflows for the entire forecast are and its associated structure.

As a configuration focal point, you can design and clean up the code within workflows to optimize performance within CHPS.

Another way to optimize performance is to create conditional workflows that only run when triggered by a situation.

Troubleshooting

When troubleshooting CHPS issues, use three techniques:

- Technique 1: Maintenance
- Technique 2: Communication Strategy
- Technique 3: Working from Common to Rare Possibilities

When troubleshooting, be aware of three types of errors: user, machine, and server/program errors.

Follow troubleshooting steps when tackling each issue.

1. Identify the Problem
2. Isolate the Issues
3. Gather Data
4. Take Action

CHPS provides access to log files using many interfaces. You can use the log.txt file, the Administration Interface, System Monitor, Log Panel in the FEWS IFD, and the Workflow Navigator to view the log messages. These applications may also help you troubleshoot your issue.

To receive a more verbose log message, edit the **Log4jConfig.xml** file to change the priority value to DEBUG. Just make sure to change this back to INFO when you resolve your issue as this change results in huge files.

If you cannot resolve your issue or want to share with others how you found a solution, use FogBugz. FogBugz is the reporting forum used by the Hydrologic Support Branch and other RFCS to support CHPS and its users. This is also a good place to search when at the beginning of troubleshooting an issue.

Congratulations. You reached the end of the course material!