CFDSource Usage

CFD Source Basics

At the core CFD facilitates importing/exporting a logical Caché/Ensemble project from the Caché database from/to a local file system. In this regard CFD is agnostic to the source control tool used and any can be used on the local file system.

CFD does provide tighter integration into GIT, which is discussed later, but first review how CFD interacts with the file system. In brief this is:

- CFD reads (or creates) a project file that describes the CFD project.
    - This includes its name, the code package(s) that make up the project, the code package(s) that make up the test and the static data and test globals.
- CFD facilitates exporting all the code and data into the file system in the project file.
- CFD also facilitates importing all the code and data from the file system.

File Repository Structure

The following is the file structures that CFD uses (it is based on a Maven project structure):

{BaseDir}\{Namespace}\{Project} …

.cacheproject	(A description of the project and the packages}
{Project}.prj	(The Studio project)
\src\main\cache*	(The code. Directories are created for the packages)
\src\main\resources*	(Any non-code resources)
\src\main\data*	(Any static/reference data)
\src\test\cache*	(Any test code)
\src\test\resources*	(Any test non-code resources)
\src\test\data*	(Any test data)

Where:

- {BaseDir} = A base directory for all source projects (default is Caché install directory\Workspaces)
- {Namespace} = The current namespace
- {Project} = The name of the CFD project (also the name of the project in Studio)

Getting Started

There are a number of ways to get started with CFD, but the following represents a simple route:

- Create a project inside Studio, the name of this is the name of CFDSource system.
- Create classes, routines, add CSP files and resources.
- Ensure CFDSource is setup as the Source Control system for the namespace
    - Restart studio if it not
- Select Confidence … New System from the menu
- Give the project a group (i.e. org.tpspencer), a version (i.e. 0.1) and a brief (1-line) description.

- Enter the prefix to any classes in the project that are test classes/resources
- Select a Repo type
    - File base repositories are supported, you should enter a path that the user running cache has permission to write to, i.e. /opt/isc/Repo
    - GITHub repositories via ssh are supported, but read the instructions on setting up with GITHub before using. Enter in your GIT username, email address and password.

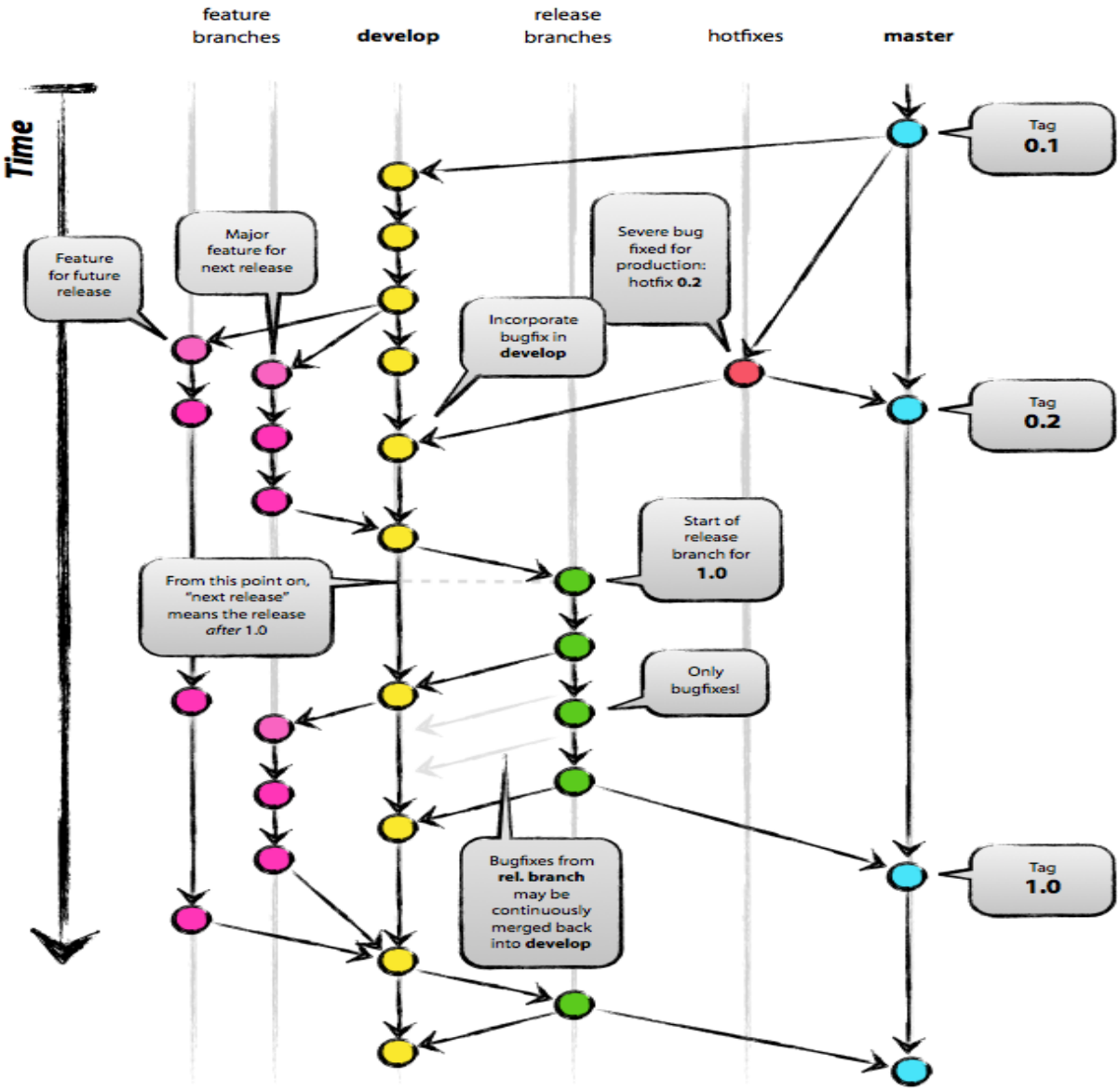
CFDSource can also be used from terminal via an instance of the CFDSource.System class, i.e.:

```
>set sys=##class(CFDSource.System).%New("{ProjectName}")
>do sys.StartFeature("{feature}", 1, .tTrace)
>… make changes
>do sys.Export("Quick describe changes", .tTrace)
>… make other changes
>do sys.EndFeature("Completed feature", 1, .tTrace)
```

Much of CFDSource is documented in the CFDSource.System class. If you can follow this then you understand how CFDSource works.

Process with GIT

The process adopted/encouraged by CFD is shown in the following diagram:



Specifically your source for the project will be help in a repository with a main branch called 'develop'. Feature branches are created to support the development of individual features. Once complete the feature branch is merged back into the main 'develop' branch.

Note: This is not the only way or right way of interacting with source control, just *a* way!!

CFD & Release Managers

Sorry, CFD does not assist release managers creating release branches, master tags or hotfixes – this must be done with GIT/other tools directly.

CFD & Developers

CFD assists you working in the model as described above when using Caché as follows:

- Supports exporting code from Caché to create a GIT repository

- Supports cloning an existing git repository
- Supports creating a new feature branch
    - (Code in the project is set as read-only when not developing a feature)
- Supports committing changes in the feature within the branch
- Supports end the feature and merging the code back into the develop branch.

CFD and Caché Studio

When inside studio, and in a CFD Source enabled namespace (see Installation), CFD Sources aims to be relatively unobtrusive. There is menu called Confidence added, the function of which is depending on the current studio project.

The studio plugin will look at the name of the current Studio project to determine if it is an active CFD project. An active CFD project has a .cacheproject file inside the {BaseDir}\{Instance}\{Namespace}\{Project} directory.

The options below will change based on whether the current studio is an active CFD project or not.

Create a GIT Repository

Available only if working in a Studio project that is not CFD project. After exporting the source as described above (and updating/creating the .cacheproject) file it runs the following git commands:

- >git init
- >git config user.name “{^CFDSource(“User”)}”
- >git config user.email “{^CFDSource(“Email”, ^CFDSource(“User”))}”
- >git add .
- >git commit –m “Initial project creation”
- >git checkout –b develop master
- >git status

In addition if you also selected to create a central repository, this is added to the project as a remote and a sync of the develop branch is made to it.

Import a GIT Repository

Available only if working in a Studio project that is not CFD project. It will import a project from GIT and will then import it into the current namespace. The git commands are:

- >git clone –b develop {RepoUrl} {ProjectName}
- >git config user.name “{^CFDSource(“User”)}”
- >git config user.email “{^CFDSource(“Email”, ^CFDSource(“User”))}”
- >git status

This should result in a local version of the repository and we will have imported any source into the current namespace.

There are a few limitations on the repository url. Essentially we cannot pass passwords on the GIT command line, so you must either use a non-password GIT URL, or specify the password in the GIT URL. For instance:

- file://path/to/repo
- http://username@password:server/repo/path

The second method means the password is saved in the GIT config file by GIT. I clone from file based locations within a trusted network. I do not recommend the latter approach, but it should work.

Project Settings

Available only if working in a Studio project that is a CFD project and allows you to effectively edit the .cacheproject file if in an active feature.

Fetch

Available only if working in a Studio project that is a CFD project.

Fetch will (re)import all the source into Caché from the file system. It will optionally fetch from the remote GIT repository (the one originally cloned from). This is useful if you have not worked on a project for some time.

The GIT commands performed are:

- >git pull
- Optionally, if a branch is specified
    - >git checkout {Branch}
    - >git status

Start Feature

Available only if working in a Studio project that is a CFD project and further where there is no current feature development.

Start will start a new feature. The name of the feature given will be the name of the branch in GIT (no spaces). The GIT repository must be sitting in the [develop] branch with no outstanding changes. After performing this operation the GIT repository will be on a branch named after the feature. Also all code in the project will be editable in Studio.

The GIT commands performed are:

- >git status
- Optionally if pulling from remote
    - >git pull
    - >git status
- >git checkout –b {Feature} develop
- >git status

Export (or Commit)

Available only if working in a Studio project that is a CFD project and further where there is a current feature open.

Export will export the source from the current project to the file system and optionally commit it in the current branch. It does not end the current feature branch.

The GIT commands performed are:

- >git status
- >git add .
- >git commit m “{Message}”
- >git status

End Feature

Available only if working in a Studio project that is a CFD project and further where there is a current feature open.

Ending the feature will cease any changes in the feature branch. It will then update the develop branch and cleverly replay all the changes made on develop. This shows up very nicely inside GIT and will allow the feature to be easily included or not included in any actual release.

Using this feature will end the current feature and move back on to the develop branch.

The GIT commands performed are:

- >git status
- >git checkout develop
- >git merge –no-ff {Feature}
- >git status
- Optionally, if push
    - >git push origin develop

Package

Available only if working in a Studio project that is a CFD project. This writes out the source into a single export file for installation on a new system (instead of individual files for each class)

CFD and Management Portal Development

It should be clear that development inside the Management Portal will just continue as before. The only point is that you must go to either Studio or the Terminal (see below) to actually export your changes into the local GIT repository. If you make changes outside of a feature, then you must start a feature before exporting.

CFD and Terminal

It is not absolutely necessary to use Studio projects can be exported at the terminal as described in the getting started above.