|  |
| --- |
| Glasswall Solutions Ltd |
| Core2 cameras - Creating and Development Rules |
| How to build and maintain Core2 style cameras |

|  |
| --- |
| Steven Warner  7-5-2019 |

## Building a Core2 Camera

### Check BDD

Confirm ‘[Naming conventions](#_Naming_conventions)’ in ‘Policies’ section adhered to

First check is that Miraplacid data viewer works with the BDD and you have run a decent number of test files through BDDLDirectoryReader to confirm the BDD is sound when parsing documents

### Prepare batchfiles

Create a setup\_(cameraname).bat in ..\glasswall.classic\data.definitions.and.rules\BDD\_DVL\_to\_code\_scripts if it does not yet exist

See [sample](#_Setup_Batch_Sample) for expected form

### Generation

New Generator script - generate\_code\_core2.bat

*Usage generate\_code\_core2.bat "tfs\_path" "cameraname" "toolversion" ["optional code params"]*

*where "tfs\_path" is the path to your workspace e.g. c:\tfs2013*

*where "cameraname" is one of the camera types e.g. emf,wmf,msdoc,msppt,msxls,printersetting,rtf,equationnative etc*

*where "toolversion" is bdd-to-cpp tool and C# dvl\_checker tool located in*

*'ucp' - default "\\STORAGE-SERVER\Development Drive\Development Tools\UCP\_ToolChain" directory*

*'ucp\_pending' - "\\STORAGE-SERVER\Development Drive\Development Tools\UCP\_ToolChain\Pending" directory*

*'ucp\_previous' - "\\STORAGE-SERVER\Development Drive\Development Tools\UCP\_ToolChain\Previous" directory*

*'ucp\_local' - "c:\UCP\_ToolChain.Local" directory, for recently rebuild tool output. Usually for development.*

It takes the same parameters as generate\_code.bat but always inserts ‘—core2’ option so that BDDs should still be able to be built in Classic for maintenance.

The output goes to a different temporary directory:

It creates a “temporary\_generate\_output.core2” directory

For each camera two directories are produced e.g.

*mp3*

*mp3.support*

The intent is to keep the root directory (mp3 in this case) for generated output that will be tried to be kept unaltered when copied across to the project.

The support directory, contains code that is partially populated ready for more hand inserted sections. (will try and place markers to help clarify)

If a generated camera is going to be heavily altered for interim work (such as reusing a classic camera) then a cameraname.hybrid directory should be created to mix the generated and hand built parts. Eg

*mp3.hybrid*

### Validation Callback and Issue ID / Sanity ID / Group ID

#### Sample of porting: Tiff

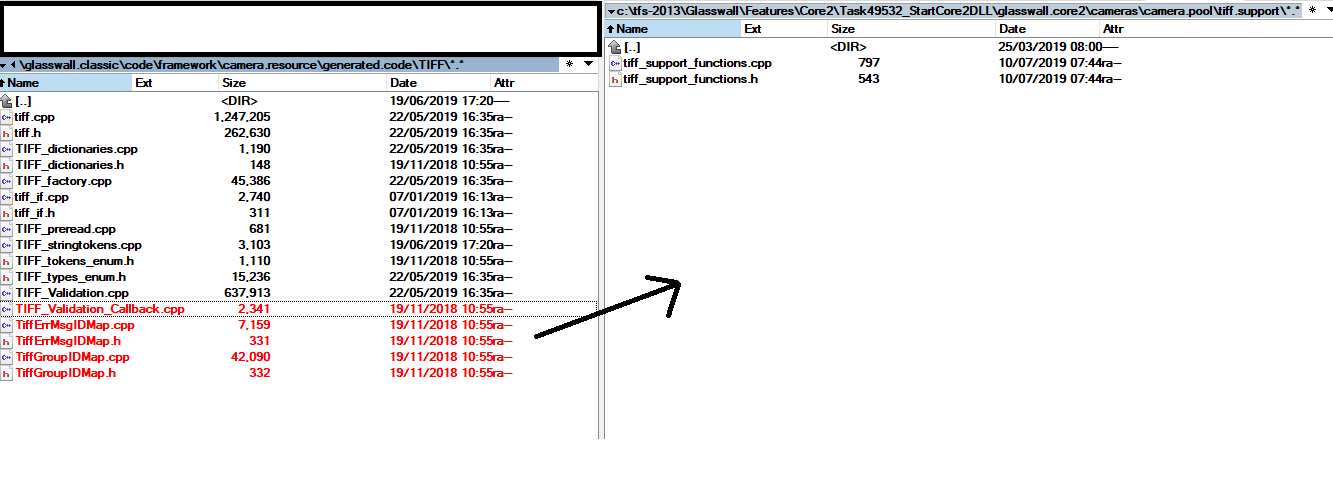
Call back code needs to be wrapped in an object to ensure no global static data is used.

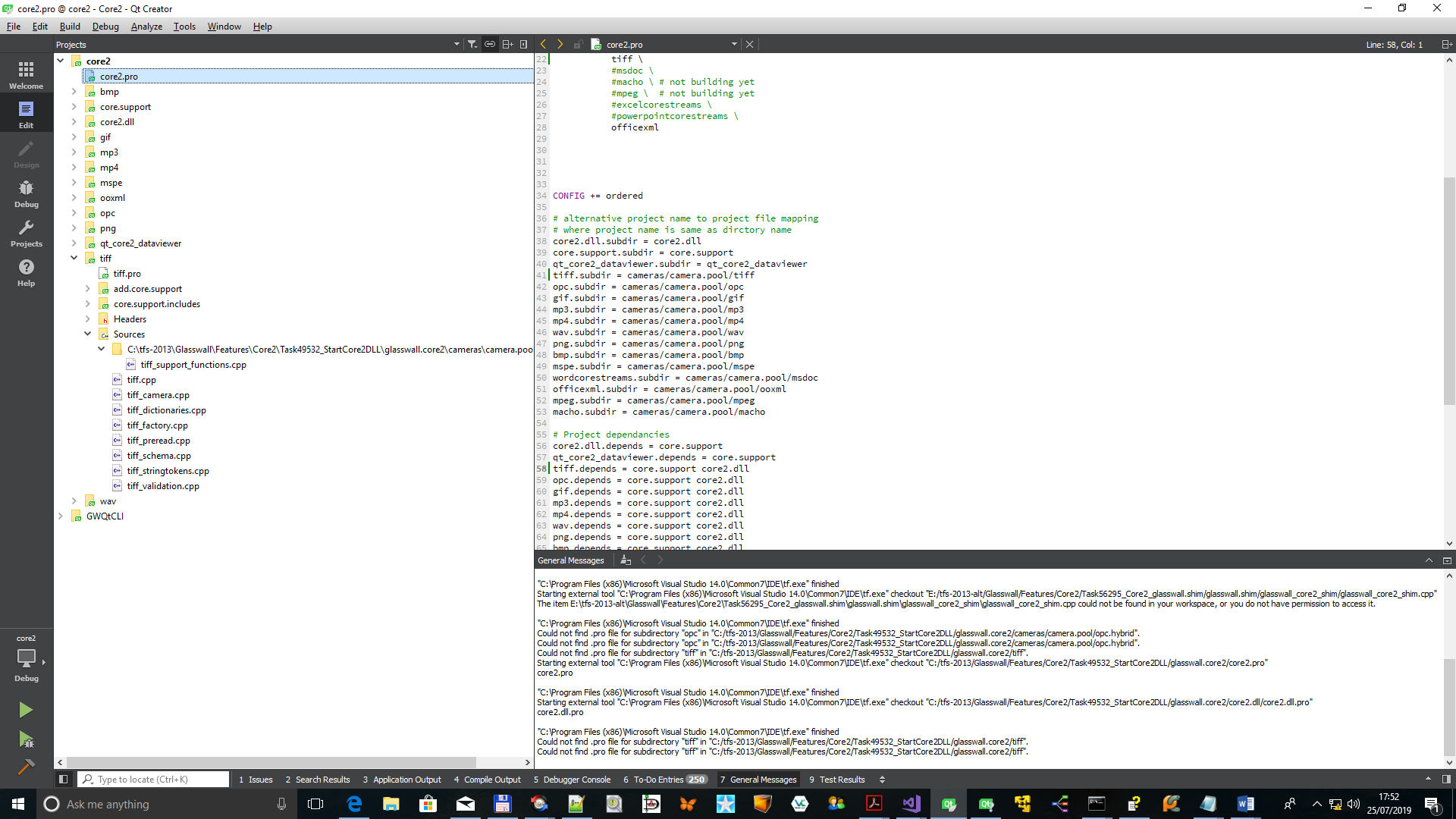
The ID mapping functions from Classic are usable and just need linking to the standard Core2 Camera interface.

The source code for Issue (aka Err) ID / Sanity ID / Group ID needs to be copied from classic generated area to the core2 <cameraname>.support directory[[1]](#footnote-1)

For completeness, a Sanitise map was created for Tiff to show the sequence of construction.

All filenames changed to all lowercase to help removed complaints from Linux when filename case differs.





Add existing files menu

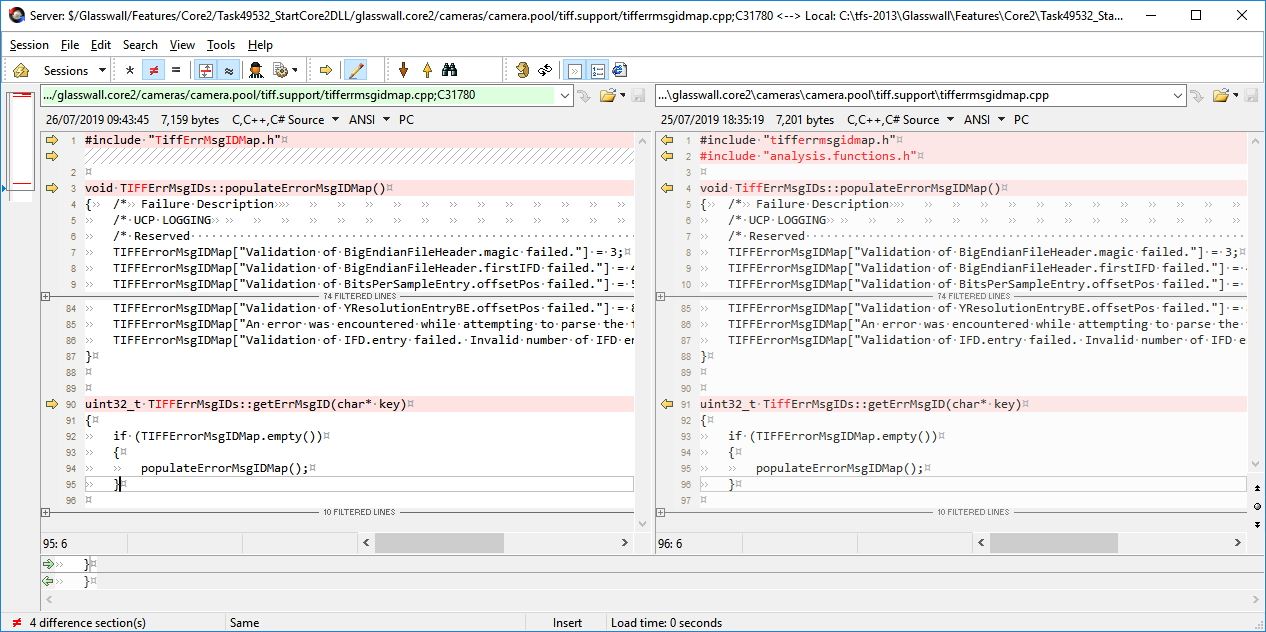
Add the new files

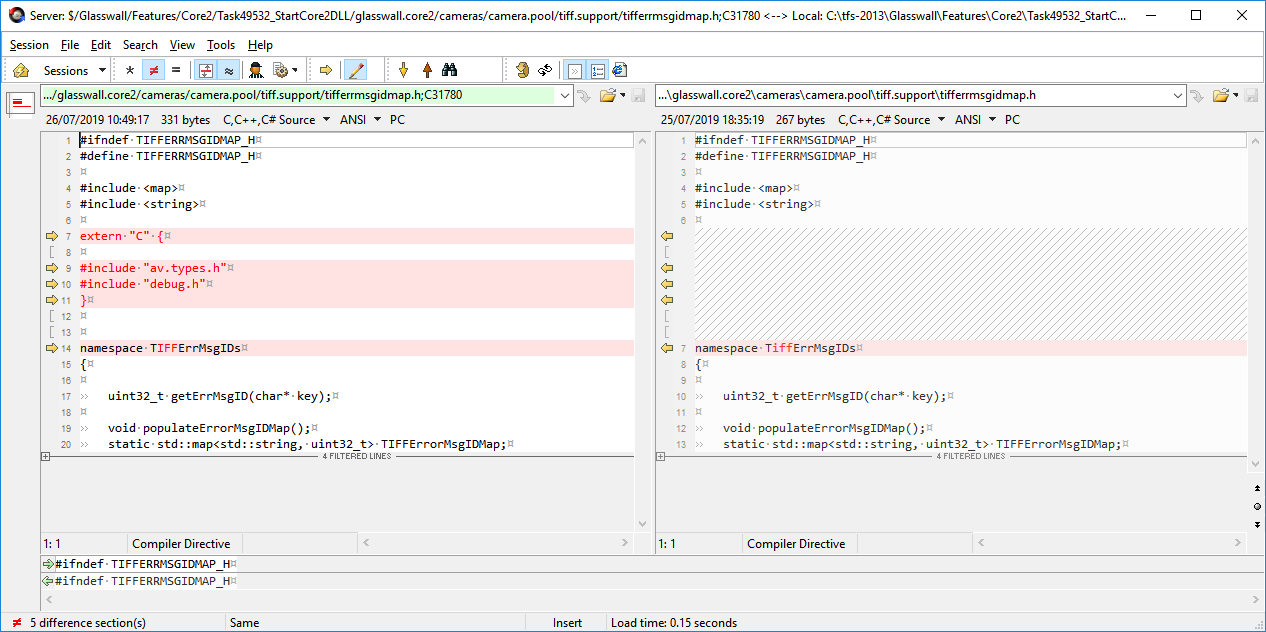
New files then part of project:



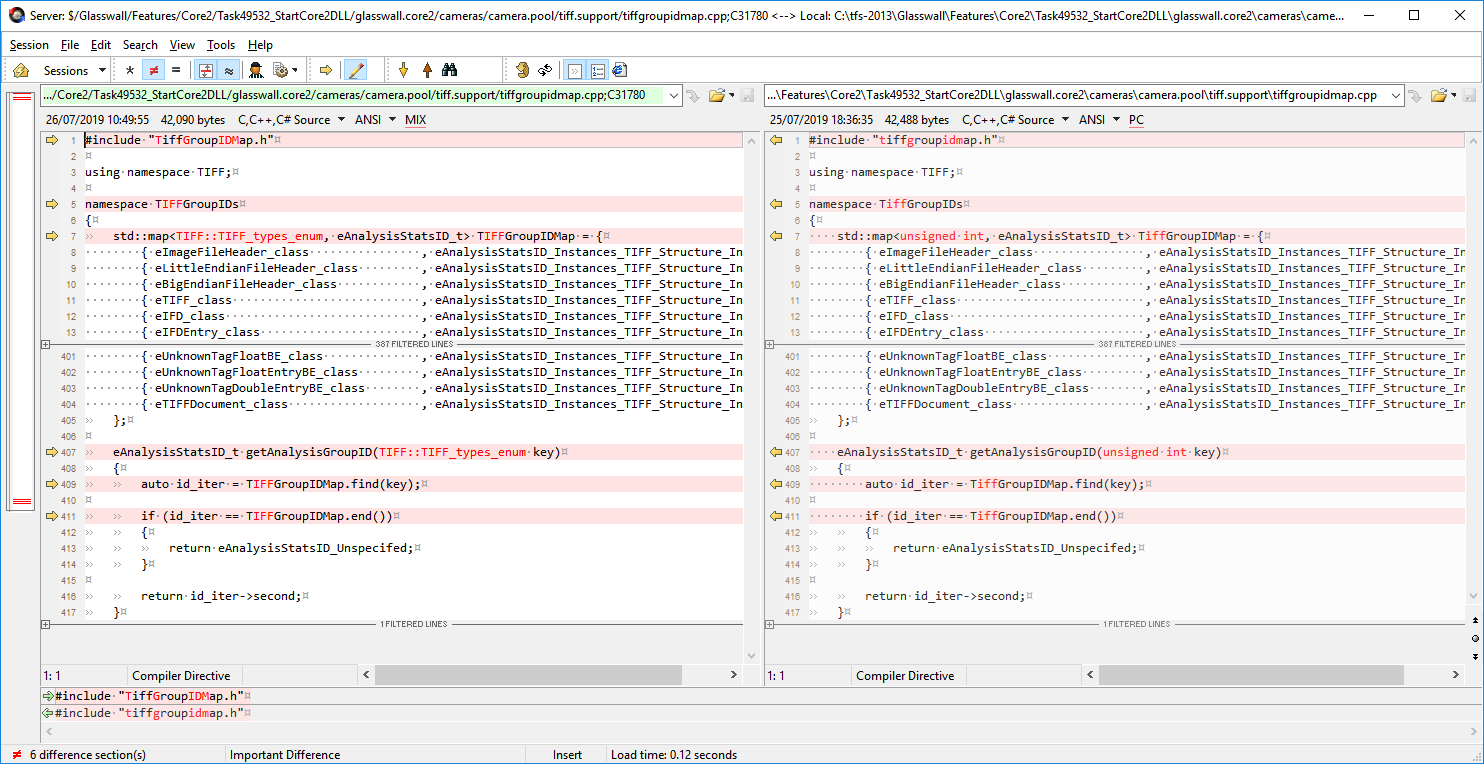
Next step is to make a few changes to the code to remove ‘C’ and old classic headers, then connect the functions to <camera>\_support\_functions.cpp

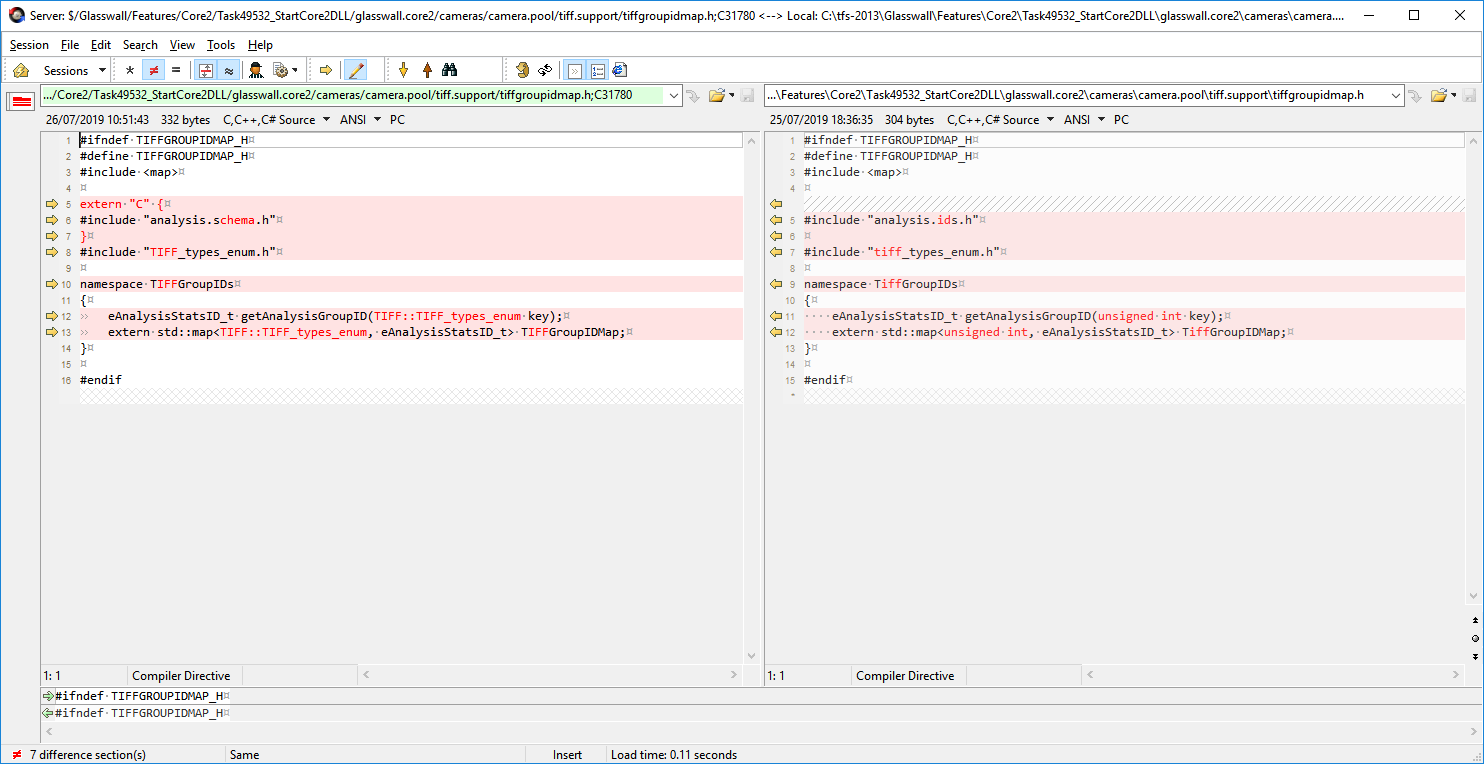
ErrMsg ID changes



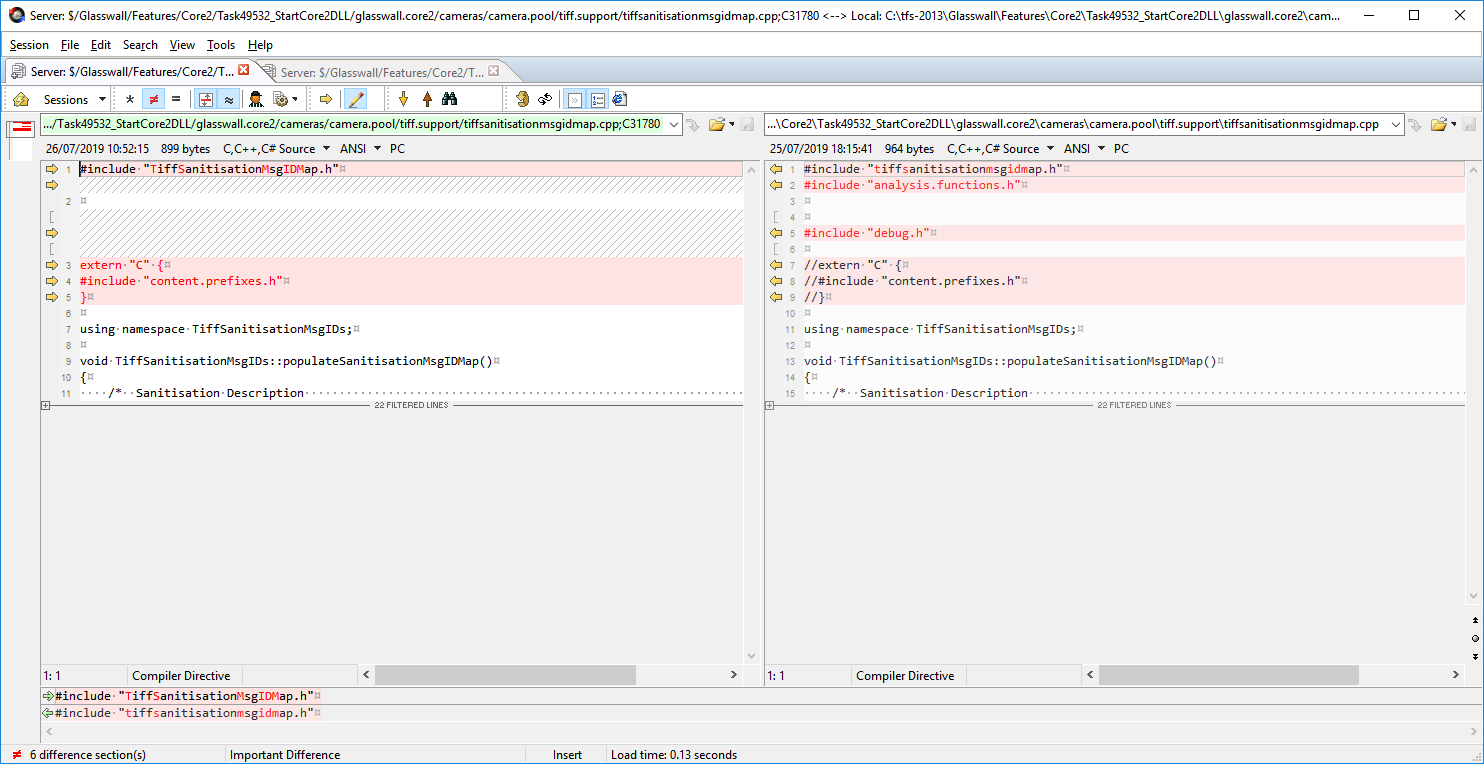


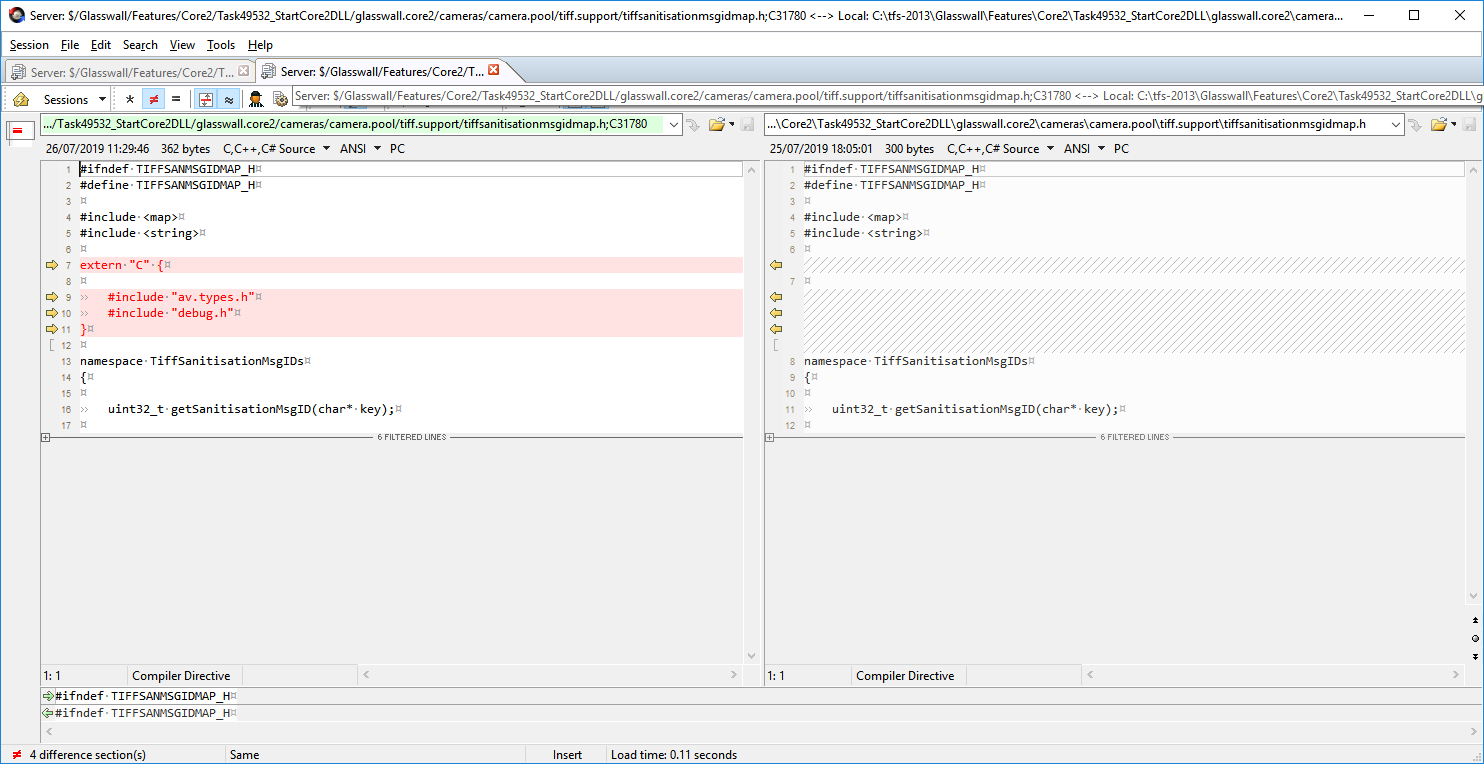
GroupMsg ID changes

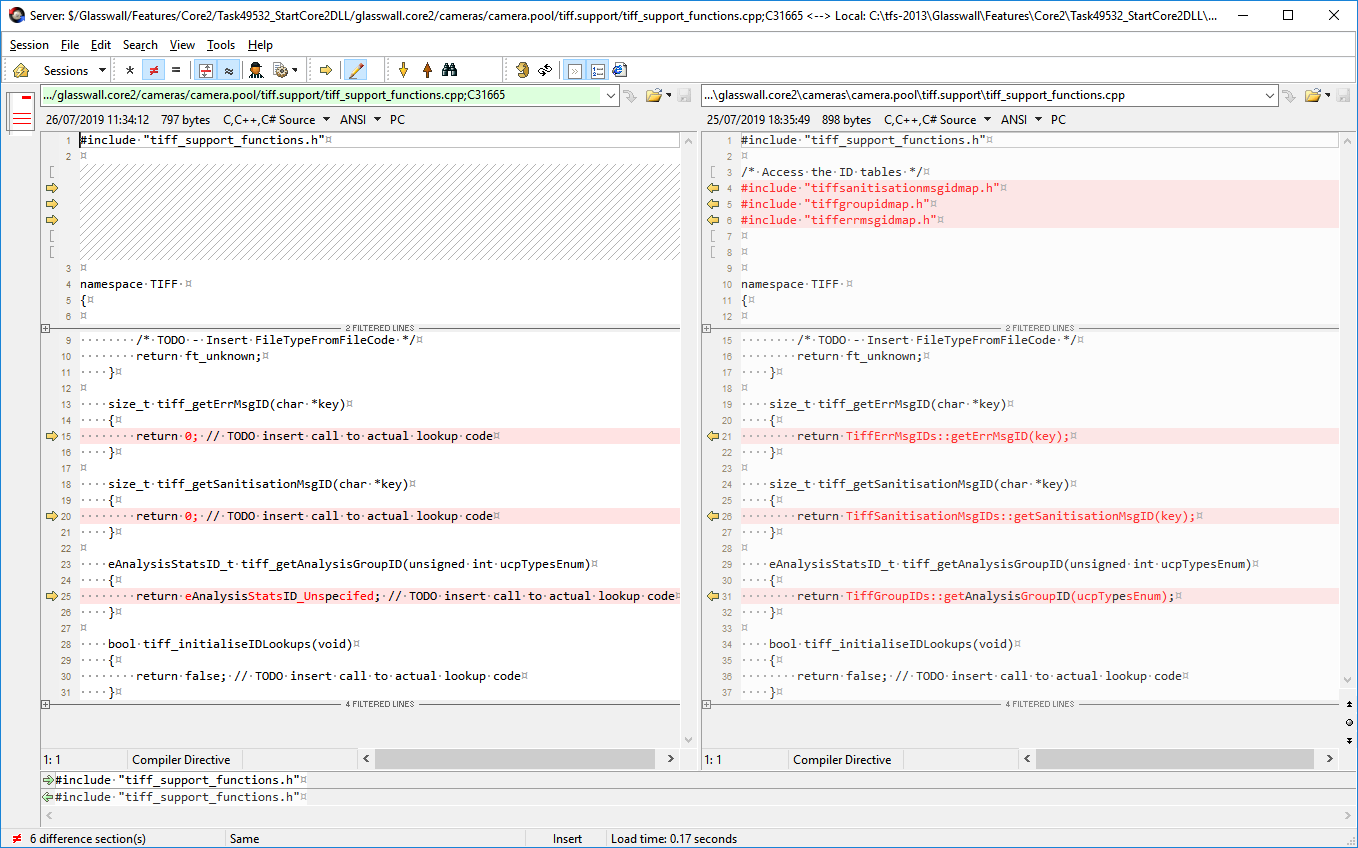




SanitisationMsg ID changes





Support Functions Changes

### Adding New Camera to Core2 Project

Project (PRO files)

There are several PRO (project) files that control the construction of the code:

Adding new cameras.

…\glasswall.core2\core2.pro

TEMPLATE = subdirs

#project name to project file mapping where name and directory are different

ooxml.file = cameras/camera.pool/ooxml/officexml.pro

bmp.file = cameras/camera.pool/bmp/BMP.pro

…

#Active subproject list : comment these out to remove from current build

SUBDIRS = core.support \

core2.dll \

qt\_core2\_dataviewer \

opc \

mp3 \

mp4 \

wav \

…

CONFIG += ordered

# alternative project name to project file mapping

# where project name is same as dirctory name

core2.dll.subdir = core2.dll

core.support.subdir = core.support

qt\_core2\_dataviewer.subdir = qt\_core2\_dataviewer

tiff.subdir = cameras/camera.pool/tiff

opc.subdir = cameras/camera.pool/opc

mp3.subdir = cameras/camera.pool/mp3

…

# Project dependancies

core2.dll.depends = core.support

qt\_core2\_dataviewer.depends = core.support

tiff.depends = core.support core2.dll

opc.depends = core.support core2.dll

mp3.depends = core.support core2.dll

…

## Pitfalls

Code generator has removed and tidied a lot of boiler-plate code in the generated output. So expect subtle differences from Classic output

Also some methods of grabbing data have changed to more compact forms.

It’s still a work in progress, so some new bugs have been introduced while trying to improve its reinterpretation of BDDL expressions.

## Policies

### Naming conventions

#### Include files

Intent where possible the generated files will stick to lowercase for filenames to avoid Linux complaints about different files when case is changed.

#### Root file names/directories

You must make sure there is a simple and consist naming of the key directory and root objects

##### BDD Sample MP3

Main BDD[[2]](#endnote-1) same name as the camera and in same named directory.

…\glasswall.classic\data.definitions.and.rules\AUDIO\bddl\**MP3**\**mp3**.bdd

BDD contains a Document object for Miraplacid to open the file :

*[extension="mp3", mime\_type="glasswall/mp3",* ***ft\_t="ft\_mp3"****]*

*document MP3Root {*

***MP3*** *mp3;*

*}*

But this is not used as kick off point for the Code Generator, instead it will use a root class as specified in the camera batch file – but for convention that should also be the same name.

*struct MP3 {*

*ID3V2Tag id3v2Tag optional;*

*Frame [\*] frames;*

*ID3V1Tag id3v1Tag optional;*

*APETag apeTag optional;*

*}*

##### DVL Sample MP3

c:\tfs-2013\Glasswall\Features\Core2\Task49532\_StartCore2DLL\glasswall.classic\data.definitions.and.rules\AUDIO\validation\MP3\mp3.dvl

*validation MP3 {*

*id3v2Tag := NO\_REQUIREMENT;*

*}*

##### Setup Batch Sample MP3

c:\tfs-2013\Glasswall\Features\Core2\Task49532\_StartCore2DLL\glasswall.classic\data.definitions.and.rules\BDD\_DVL\_to\_code\_scripts\setup\_mp3.bat

*pushd ..*

*set CD\_PARENT=%CD%*

*popd*

*set "PATH\_TO\_BDD=%CD\_PARENT%\Audio\bddl\****MP3****\****mp3****.bdd"*

*set "PATH\_TO\_DVL=%CD\_PARENT%\Audio\validation\****MP3****"*

*set "BDD\_ROOT=****MP3****"*

*set "CODEOPTIONS= --enable\_dataheap --core2"*

*set "DVLOPTIONS= --core2"*

1. <cameraname>.support directory is usually expected to need careful merging with updated support template code when you regenerate the camera from BDD [↑](#footnote-ref-1)
2. Note BDDs are still in glasswall.classic directory … probably should be moved up level at some point [↑](#endnote-ref-1)