CSSP Desktop for developers

**NOTE: All services, tools and applications have duplicate projects for testing**

All code can be found under <https://github.com/ECCC-CSSP> under the CSSPTools repository

CSSPAzureAppTaskModelServices

Used to send AppTaskModel items to Azure database

CSSPCreateGzFileLocalServices

Used to create .gz local files from local database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBLocal.db). These files are then stored under C:\CSSPDesktop\csspjsonlocal\

CSSPCreateGzFileServices

Used to create .gz files, which are compressed .json files, from the CSSPDB database. These files are then stored under <https://csspstorage.blob.core.windows.net/csspjson/> . Accessing this directory is private and requires a key.

CSSPCultureServices

Used to hold all the text information in (fr and en)

CSSPCustomAttributes

Used to hold all the customized attributes used throughout the code

CSSPDBCommandLogModels

Contains the structure of the CSSPDBCommandLog database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBCommandLog.db). Used with CSSPDBCommandLogServices to managed the storage of all local commands. Mainly used to help find future bugs.

CSSPDBCommandLogServices

Used to managed the storage of all local commands in CSSPDBCommandLog database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBCommandLog.db). Mainly used to help find future bugs.

CSSPDBFilesManagementModels

Contains the structure of the CSSPDBFilesManagement database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBFilesManagement.db). Used with CSSPDBFilesManagementServices to managed the local copies of Azure hosted files.

CSSPDBFilesManagementServices

Used to managed the local copies of Azure hosted files in CSSPDBFilesManagement database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBFilesManagement.db).

CSSPDBLocalServices

Used to store new and changed information locally in CSSPDBLocal database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBLocal.db). The permit the user to work offline. It is then required to Sync the local database with the Azure database once in a while.

CSSPDBModels

Used to hold the data types and database contexts for CSSPDBLocal (Local) and CSSPDB (Azure).

CSSPDBPreferenceModel

Contains the structure of the CSSPDBPreference database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBPreference.db). Used with CSSPDBPreferenceServices to managed local login, internet connection, keys and authorization.

~~CSSPDBPreferenceServices~~

~~Used to managed local login, internet connection, keys and authorization in CSSPDBPreference database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBPreference.db).~~

CSSPDBSearch

Contains the structure of the CSSPDBSearch database (C:\CSSPDesktop\cssplocaldatabases\CSSPDBSearch.db). Used with CSSPDBSearchServices to manage the 2 tables within CSSPDBSearch.db. TVItems and TVItemLanguages are used to be able to quickly do a search even without internet connection.

CSSPDBSearchServices

Used to manage the 2 tables within CSSPDBSearch.db (C:\CSSPDesktop\cssplocaldatabases\CSSPDBSearch.db). Tables TVItems and TVItemLanguages within the database are used to be able to quickly do a search even without internet connection.

CSSPDBServices

Used to manage the Azure database.

CSSPDesktop

Is the actual local application. It can be installed using <https://csspdesktopinstall.azurewebsites.net/>.

CSSPDesktopInstallPostBuild

Internal use only. It uploads necessary files for CSSPDestop update. Which install a local web server on the user machine for offline browsing of CSSP information.

CSSPDesktopServices

Is the main service for the CSSPDesktop application

CSSPDownloadFileServices

Used to download files from Azure files (<https://csspstorage.file.core.windows.net/csspfiles/>) and store them locally (C:\CSSPDesktop\csspfiles\). Accessing the files on Azure is private and requires a key.

CSSPEnums

Used to hold all the Enums and related Enums functions required throughout the code.

CSSPHelperModels

Used to hold data types and classes that are not directly used in databases.

CSSPHelperServices

Used to manage and verify data types and classes that are not directly used in databases.

CSSPLocalTaskRunnerServices

Used to execute local task which might take a long time. Like generating a report or a kml document for example.

CSSPLoggedInServices

Used to manage logged in contact both locally and on Azure.

CSSPReadGzFileServices

Used to read .gz files, which are .json compressed files, into a single complex object. It makes the transferring of information from local server to local web site easier.

CSSPReadGzFileServices

Used to scramble some hashed variables.

CSSPServerTaskRunnerServices

Used to execute server task which might take a long time. Like MIKE scenario for example.

CSSPSQLiteServices

Used to create and setup local SQLite databases (CSSPDBCommandLog.db, CSSPDBFilesManagement.db, CSSPDBLocal.db, CSSPDBPreference.db, CSSPDBSearch.db) which are stored under C:\CSSPDesktop\cssplocaldatabases\. These are created once during the installation and update of CSSP Desktop.

CSSPSyncDBsServices

Used to Sync local and Azure databases and report back any issues that could have occurred back to the user.

CSSPUploadFileServices

Used to upload a file to the local files archiving under C:\CSSPDesktop\csspfiles\n\, where n is the TVItemID which is unique and represent an item within the database. It could be a subsector, MWQM site, pollution source site, etc.

CSSPWebAPIs

Is the main web application programming interface on Azure. It’s actually being hosted on Azure and contains many functions to manage Azure database and other tasks.

CSSPWebAPIsLocal

Is the main web application programming interface on the user computer. It’s actually being hosted on the client/user and contains many functions to manage local database and other tasks. It is also responsible to send .gz files, which are compressed .json files, to the Angular client. Angular is a javascript driven web client viewer developed by Google.

CSSPWebModels

Used to hold data structures/types for easy transfer between applications. Its mainly used by CSSPWebAPIsLocal and Angular client.

~~PolSourceGroupingExcelFileReadServices – used to read the C:\CSSPTools\src\assets\PolSourceGrouping.xlsm file. This dll is temporary until a more appropriate solution is found which will use the CSSPDB to hold all the Groupings information.~~

Azure Components (<https://portal.azure.com/#home>)

Charles LeBlanc Azure Account

@charlesleblancecgc.onmicrosoft.com

App Services

CSSPDesktopInstall

Used to install CSSPDesktop. It also takes care of updates.

CSSPWebAPIs

Used to access CSSPTemporaryDB. It also stores contact, authorization and AppTask the server needs to accomplish. The CSSPTemporaryDB is just a sub portion of the CSSPDB database which will be replacing the CSSPTemporaryDB once everything is hosted by Azure.

CSSPLabSheet

Used with the Desktop application (CSSPWQInputTool) (this tool will be replaced by the new CSSPDesktop)

Databases

CSSP

Used with CSSPLabSheet (app services) and CSSPWQInputTool (desktop client) (temporary, will be removed in the future once CSSPDesktop is completed)

CSSPTemporaryDB

Used with CSSPWebAPIs (app services) and CSSPDesktop (desktop client). This DB will be renamed to CSSPDB in the future version

Storage

csspstorage

Used to store static information that could have been pre-generated or uploaded

csspwebapislocal (blob container)

Used to store apps related information

csspclient.zip

Contains all the required files to run the angular client app (this file should be unzipped under C:\CSSPDesktop\csspwebapislocal\csspclient\)

csspotherfiles.zip

Contains all the HTML help docs, Google Material files and Google Maps javascripts used with the angular client app and CSSPWebAPIsLocal (this file should be unzipped under C:\CSSPDesktop\csspotherfiles\)

csspwebapislocal.zip

Contains all the required files to run CSSPWebAPIsLocal, a local web server (this file should be unzipped under C:\CSSPDesktop\csspwebapislocal\)

csspjson (blob container)

Used to store all the pre-generated compressed JSON files (.gz). These files will be regenerated when required by the CSSPServerTaskRunner running on the server. All compressed JSON files (in total ~ 24000 files) are stored using various names like shown below where nnn is the CSSPDB TVItems unique number (TVItemID):

WebAllAddresses.gz (Contains all addresses)

WebAllContacts.gz (Contains all contacts)

WebAllCountries.gz (Contains all countries)

WebAllEmails.gz (Contains all email addresses)

WebAllHelpDocs.gz (Contains all help docs)

WebAllMunicipalities.gz (Contains all municipalities)

WebAllMWQMLookupMPN.gz (Contains all MWQM Lookup MPN)

WebAllPolSourceGroupings.gz (Contains all pollution source groupings)

WebAllPolSourceSiteEffectTerms.gz (Contains all pollution source site effect terms)

WebAllProvinces.gz (Contains all provinces)

WebAllReportTypes.gz (Contains all report types)

WebAllTels.gz (Contains all telephone numbers)

WebAllTideLocations.gz (Contains all tide locations)

WebAllTVItemLanguages.gz (Contains all TVItemLanguages)

WebAllTVItems.gz (Contains all TVItems)

WebArea\_nnn.gz (Contains all area related information)

WebClimateDataValue\_nnn.gz (Contains all climate data value for a particular climate site)

WebClimateSite\_nnn.gz (Contains all climate sites under a province)

WebCountry\_nnn.gz (Contains all country related information for a particular country)

WebDrogueRun\_nnn.gz (Contains all drogue runs related information for a particular subsector)

WebHydrometricDataValue\_nnn.gz (Contains all hydrometric data value for a particular hydrometric site)

WebHydrometricSite\_nnn.gz (Contains all hydrometric sites under a province)

WebMikeScenario\_nnn.gz (Contains all MIKE scenarios related information for a particular MIKE scenario)

WebMunicipalities\_nnn.gz (Contains all municipalities under a particular province)

WebMunicipality\_nnn.gz (Contains all municipality related information for a particular municipality)

WebMWQMRun\_nnn.gz (Contains all MWQM runs related information for a particular subsector)

WebMWQMSample\_nnn\_nnn2\_nnn3.gz (Contains all MWQM samples related information for a particular subsector between years “nnn2” and “nnn3” inclusively.

WebMWQMSite\_nnn.gz (Contains all MWQM sites related information for a particular subsector)

WebPolSourceSite\_nnn.gz (Contains all pollution source site related information for a particular subsector)

WebProvince\_nnn.gz (Contains all province related information for a particular province)

WebRoot.gz (Contains all root related information)

WebSamplingPlan\_nnn.gz (Contains all sampling plan related information for a particular sampling plan)

WebSector\_nnn.gz (Contains all sector related information for a particular sector)

WebSubsector\_nnn.gz (Contains all subsector related information for a particular subsector)

WebTideDataValue\_nnn.gz (Contains all tide data value for a particular tide site)

WebTideSite\_nnn.gz (Contains all tide sites under a province)

csspfiles (file shares)

Used to store all the CSSP files (.docx, .xlsx, .gif, .png, .dfsu, etc…) All files are stored under <https://csspstorage.file.core.windows.net/csspfiles/> with the TVItemID as the subdirectory. So for example all files stored under the <https://csspstorage.file.core.windows.net/csspfiles/>1/ are related to Root while subdirectory ending with …\5\ are all related to Canada and so on. File name under a particular subdirectory needs to have unique names.

Local Components (C:\CSSPDesktop\)

Directories

csspfiles

Used to hold a copy of the Azure storage csspfiles (the content of this directory is managed by CSSPWebAPIsLocal and the use of SQLite database CSSPDBFilesManagement.db)

csspjson

Used to hold a copy of the Azure storage csspjson (the content of this directory is managed by CSSPWebAPIsLocal and the use of SQLite database CSSPDBFilesManagement.db)

csspjsonlocal

Used to hold .gz files created using the local CSSPDBLocal database. It’s use with the same name under csspjson directory to assemble all information old, new and changed for viewing with Angular client.

cssplocaldatabases

Used to hold local database which are used by CSSPDesktop and CSSPWebAPIsLocal. All the databases listed below will be created the first time CSSPDesktop is install and verified after each update. If the structure of the CSSPDB change then it could be that some of them needs to be recreated. The users will be asked to sync all their changes to Azure before updating is permitted.

CSSPDBCommandLog.db (SQLite)

Used to help store a history of commands sent to CSSPWebAPIsLocal. Mainly used to help with the debugging

CSSPDBFilesManagement.db (SQLite)

Used to help manage the sync of local and server csspfiles

CSSPDBLocal.db (SQLite)

Used to store changes locally until the user decides to upload all its local changes to Azure. This is comparable to CSSPWQInputTool and CSSPPolSouceInputTool local storage (.txt files) but in a database format. This database has identical tables and fields as CSSPDB.

CSSPDBPreference.db (SQLite)

Used to managed local login, internet connection, keys and authorization. When the users is not connected to the internet, they still require to identify themselves through login. When not connected to the internet, it will use this database to login. When connected to the internet it will use the Azure CSSPTemporaryDB database. The first time the users install the CSSPDesktop application, they will be asked to login with the Azure CSSPTemporaryDB. A local copy of the information required is then stored locally for later use.

CSSPDBSearch.db (SQLite)

Used to be able to quickly do a search even without internet connection.

csspotherfiles

Contains all the HTML help docs, Google Material files and Google Maps javascripts used with the angular client app and CSSPWebAPIsLocal (this file should be unzipped under C:\CSSPDesktop\csspotherfiles\)

csspwebapislocal

Contains all the required files (csspclient and CSSPWebAPIs.exe and related/required files). The CSSPDesktop tool will run the CSSPWebAPIsLocal. The Angular (.js) and Index.html is also stored within this subdirectory under csspwebapislocal/csspclient

**Purpose and details of CSSP Web APIs**

This CSSPWebAPIs mini web server located at C:\CSSPDesktop\csspwebapis\csspwebapis.exe is automatically started when the Start button is clicked.

The main purpose of this local mini web server is to be able to provide information requested from the user interface (Local Web Site). It’s also the brain of all commands coming from the local web site. It is responsible in managing all request and knowing when information need to be pulled down from Azure or use the local information stored in C:\CSSPDesktop\csspjson and C:\CSSPDesktop\csspfiles.

It’s also responsible in orchestrating and sending specific tasks to external software (single purpose application) which most of them will be run locally.

* TaskRunnerCSV – used to generate .csv files
* TaskRunnerDOCX – used to generate .docx files (Word Documents)
* TaskRunnerKML – used to generate .kml or .kmz files (Google Earth)
* TaskRunnerMIKE – used to read, parse and interpret .m21fm, .m3fm, .dfs0, dfs1 and .dfsu
* TaskRunnerTXT – used to generate .txt files
* TaskRunnerXLSX – used to generate .xlsx files (Excel Documents)
* TaskRunnerJSON – used to generate .json files (compressed JSON files could also be .zip or .gz

**How CSSPWebAPIs works**

CSSPWebAPIs was developed using asp.net core (Microsoft). It is built with both the local version and the server version (Azure) within the same code.

CSSPWebAPIs (local version) is responsible to send JSON and other files (.png, .jpg, .docx, .xlsx, .txt, etc..) type information to the user interface (web site). It first starts by downloading required JSON and/or other files from Azure and storing them on the local computer, if these files does not already exist on the local computer. It only downloads and store information needed and gradually. It then read the local version of the files and send the information to the web site that requested it.

Note: For the first-time use, you will need internet connection to at least start downloading required files to view them on the web site (UI).

When users make changes on the web site or uploads a new file, the local server will first upload or change the information locally using local version of the database and store the file under C:\CSSPDesktop\csspfiles. Once in a while, when the user has internet connections, the changed and/or new local information and files are sent to Azure in the background. Reports of the background upload/sync will be available for users to see and analyze. Giving them the opportunity to reload the information.

Note: Regarding updating the web component of CSSP Desktop. The user will be asked to make sure all changed information (stored locally) be sent the Azure before being able to accept/do the update. It could happen that some of the local database needs to change in structure and emptying them before changing their structure is important so no information is lost.

Example of logic when the web site requests a web page.

With internet connection:

* Web site request a web page
* CSSPWebAPIs accept the request
* It collects all the .json file required to fulfill the request (could be more than one .json files)
* It then checks all required .json files in the database CSSPDBFilesManagement.db to see if these files are already stored locally or not and if they need an update using Microsoft Azure Store ETag,
* For the .json files not stored locally yet, these files are downloaded from the Azure Store csspjson and stored locally under C:\CSSPDesktop\csspjson\
* They are then read, parsed and stored in an in-memory database using the same structure as CSSPDB
* The JSON information is then read from the in-memory database (might use LINQ to only send partial/filtered information to the client)
* The same process is done for normal files like (.xlsx, .docx, .dfsu etc)
* The name of these files to be downloaded are within the .json files
* The files are then downloaded from Azure Store csspfiles and stored under C:\CSSPDesktop\csspfiles\

Example of logic when information is changed or added from the web site (client UI)

* When user change or add new stuff using the web site, the information is temporary stored locally using C:\CSSPDesktop\cssplocaldatabases\CSSPDBLocal.db (SQLite)
* CSSPDBLocal.db has the same database structure as CSSPDB located on Azure
* Changed information is stored in the database with the same identifiers used in CSSPDB (Azure)
* New information however is stored in the database with very high values as identifiers
* The local CSSPDBLocal does not have any relationship between tables in order to be able to use the correct identifies without uploading all necessary tables and information within the local database (CSSPDBLocal)
* CSSPWebAPIs has to maintain all these relationships itself.
* When .json files are parsed the information is placed into an in-memory database with the same structure as the CSSPDB (Azure) and CSSPDBLocal.
* This in-memory database can then be use to verify all necessary relationships in order to accept the changes or additions to the CSSPDB/CSSPDBLocal
* Once the information is changed the in-memory has to reflect the new changes or additions.
* In-memory database has to be reloaded every time CSSPDesktop is restarted

Example of logic when change or new information is sent to Azure

* User uses the upload to Azure page to manage when to upload
* Information is gathered and sent to Azure parts at a time
* CSSPWebAPIs (server version) receives the information from the user, this could take many back and forth (sending multiple .json file and other types of files (.xlsx, .docx, .dfsu…)
* It then stores the work job id in a table within the CSSPDB database located on the server
* Once all information has been uploaded, parsed, entered in the CSSPDB database and all the files (images, documents etc…) are also uploaded to the server and stored properly using CSSPWebAPIs server
* It then need to recreate all the .json for which some information has changed.
* It then sends the status of the upload to the client in order to inform the client if everything was ok or indicate where some error occurred so the user can fix it
* If everything is ok and all the changed .json file are recreated, an ok message is sent to the CSSPWebAPIs (client)
* The CSSPWebAPIs (client) then needs to remove all information from the C:\CSSPDesktop\cssplocaldatabases\CSSPDBLocal.db that was properly sent and stored in CSSPDB Azure

Purpose of each CSSP Desktop databases

* CSSPDB (Azure SQL Server) is the master database and is considered the correct and final storage of information for the CSSP Desktop
  + It is used to pre-generate .json files which are stored on Azure Store csspjson
  + Every time these are changes made to CSSPDB, some of these .json file might have to be regenerated
  + It is also used to generate other types of files which will be used for Arc GIS and other tools
* C:\Desktop\cssplocaldatabases\CSSPDBLocal.db (local) is the main local database which has the same structure as the CSSPDB (Azure)
  + It stores the user changes or additions temporary until the user decides to upload the changes to Azure
  + It also allows the user to work offline with the .json files (the user has to make sure they download all necessary information before however)
* C:\Desktop\cssplocaldatabases\CSSPDBFilesManagement.db (local) is use to manage the downloaded files
  + Keeps the names, date created and Etag for each file downloaded from Azure csspjson and csspfiles
  + It can then verify the Etag to see if the file has changed on Azure since the last download
* C:\Desktop\cssplocaldatabases\CSSPDBLogin.db (local) is use to store relevant information for the user to be able to login without being connected to the internet
  + Used mainly to be able to login to the web site (client UI) when the user is not connected to the internet
  + It also saves some basic user preferences
* In-Memory database (local) is use for quick access to information which the user might request via the web site (client UI)
  + It’s main purpose is for speed
  + It’s also used to make sure the user changes and additions are properly verified with relational tables
  + It’s very much dynamic following the users needs and location on the web site
  + It only store what is needed and removes the rest, so if a user goes to another subsector it will remove the information from the previous subsector and add the information from the new subsector

Advantage and purpose of using Angular Client

* Angular framework, developed by Google, has the advantage of pre-compiling all javascript and html template into some single javascript files which can and is downloaded only once.
* CSSPDesktop is responsible to download/update the csspclient (Angular) into C:\CSSPDesktop\csspwebapis\csspclient\
* Once downloaded, it does not have to be downloaded again until a new update is available
* This pre-download and pre-compile save a lot of unnecessary downloads from Azure
* Adding to this that most of the information from CSSPDB is pre-generated into .json file which are compressed at > 95%.
* These .json files are also just downloaded once and stored on the user computer under C:\CSSPDesktop\csspjson\
* Files like (Excel, Word, etc) are also only downloaded once and stored on the user computer C:\CSSPDesktop\csspfiles\
* These are only re-downloaded when they change on Azure

Purpose of storing CSSPDB (Azure) content in pre-generated .json files

* In the old CSSPWebTools every time the user requested a web page the information had to be taken from the CSSPDB using SQL Server, Filtered and interpolated into something like a template which created information of .html type (web page). Some of these web pages could start to be big (> 1 MB). So every time one user would hit that page the all of these process had to be done.
* With the new way of doing things, having .json files pre-generated and zipped (often zipping at >95%), the information being transferred is reduced considerably
* It also allow the user to work offline (no internet connection)