



**DECIPHERING YOUR DATA.**

**STF - Configuration and usage Guide**

**CONTENT:**

[Introduction 1](#_Toc58605008)

[Overview 1](#_Toc58605009)

[Requirements 2](#_Toc58605010)

[Example configurations 2](#_Toc58605011)

[Example minimum requirements 2](#_Toc58605012)

[Accessing the code and resources 3](#_Toc58605013)

[The STF example app design premise 4](#_Toc58605014)

[Overview 4](#_Toc58605015)

[Designed logical flow 5](#_Toc58605016)

[Code reference details 6](#_Toc58605017)

[Examples of risk considerations in development decisions 8](#_Toc58605018)

[JSON File 9](#_Toc58605019)

[Configuration file 14](#_Toc58605020)

[Log File 16](#_Toc58605021)

[Password encryption 18](#_Toc58605022)

[Version History 19](#_Toc58605023)

## Introduction

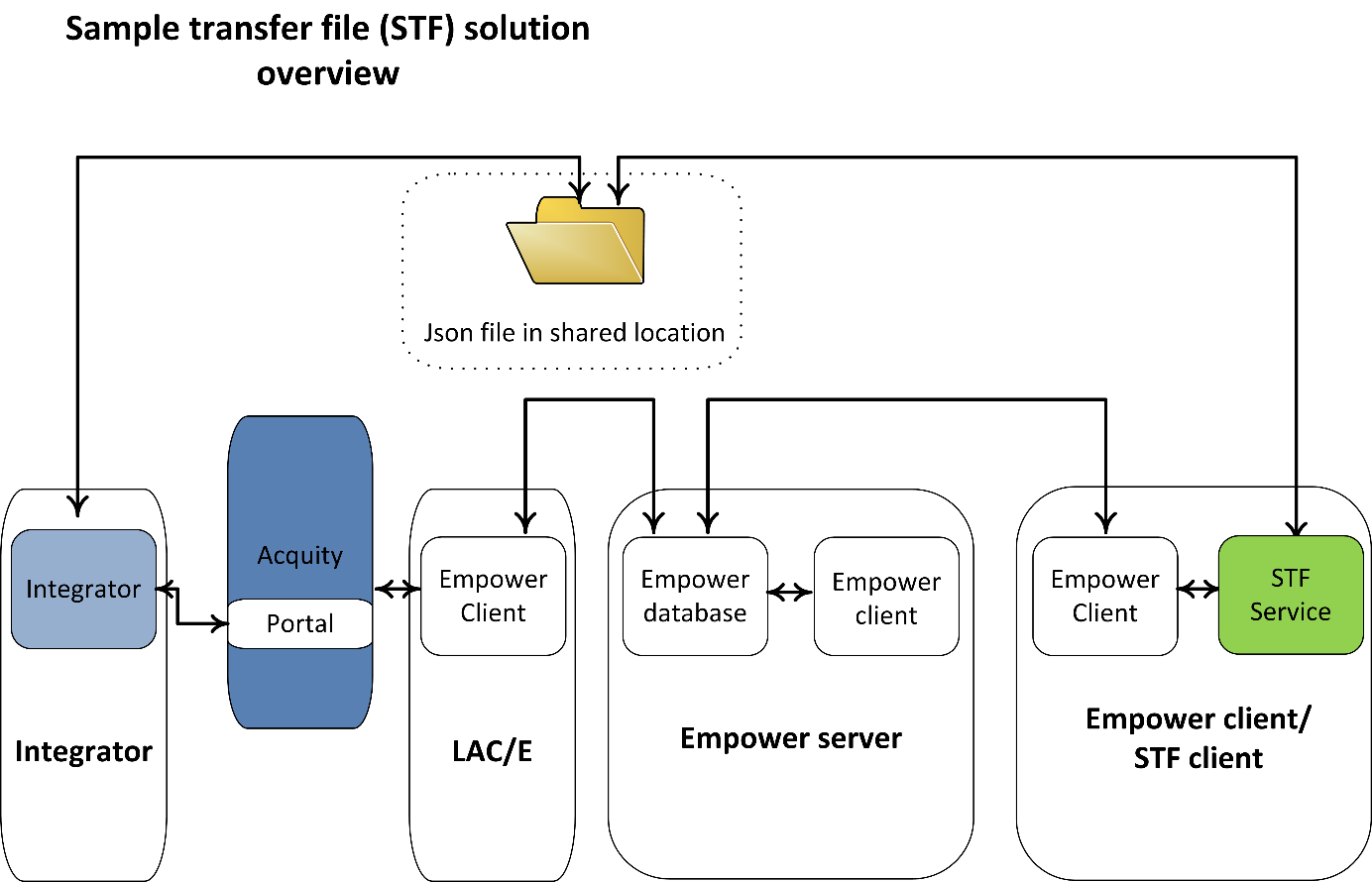
The STF solution is an example of how to create an interface to Waters Empower software via the Empower Toolkit. It is designed and written exclusively for integrators with the Waters portal. The STF provides an example of code that will accept metadata in standard JSON format and use this metadata to create and run sample set methods (SSM) in Empower. It is the responsibility of the integrator’s development team to rework the code and ensure that the resultant code meets their requirements.

The example code is provided under a Material Transfer Agreement (MTA). and does not entitle any code ownership or copyright of the example code. The MTA provides terms of use for the example code and the limits of support that is provided by Waters and DataCiph.

### Overview

The example STF solution describes a service that would provide the integration functionality for an Empower enterprise installation. The example is written in C# and is commented to explain how the code works. A set of help files are also provided to assist the developer. The example would run on the same machine as the Empower client and access either a local folder or a network share for the location of the JSON file.

The following diagram shows a simplified overview of an Empower Enterprise installation. The example solution is shown in green (STF Service) and communicates via the Empower Toolkit. This solution could reside on any system where the Empower client software is installed.



## Requirements

### Example configurations

The example STF would run with Empower 3 FR4 and FR5.

### Example minimum requirements

Empower software version

Empower 3 (3471) FR4 client

DotNet framework version

Version 4.8

## Accessing the code and resources

The code will be available from Waters.

You will need to check out the code, open with Microsoft Visual Studio 2017 or equivalent. Then build for x86 systems. It is important to build for x86 as the Empower Toolkit is 32bit.

It is a two level with a wrapper type layer of code that deals with the Toolkit calls and a layer of application code that provides the functionality. This important concept provides a layer of field verification before the call is made to Toolkit and reduces the number of unhandled exceptions due to invalid calls and unhandled errors.

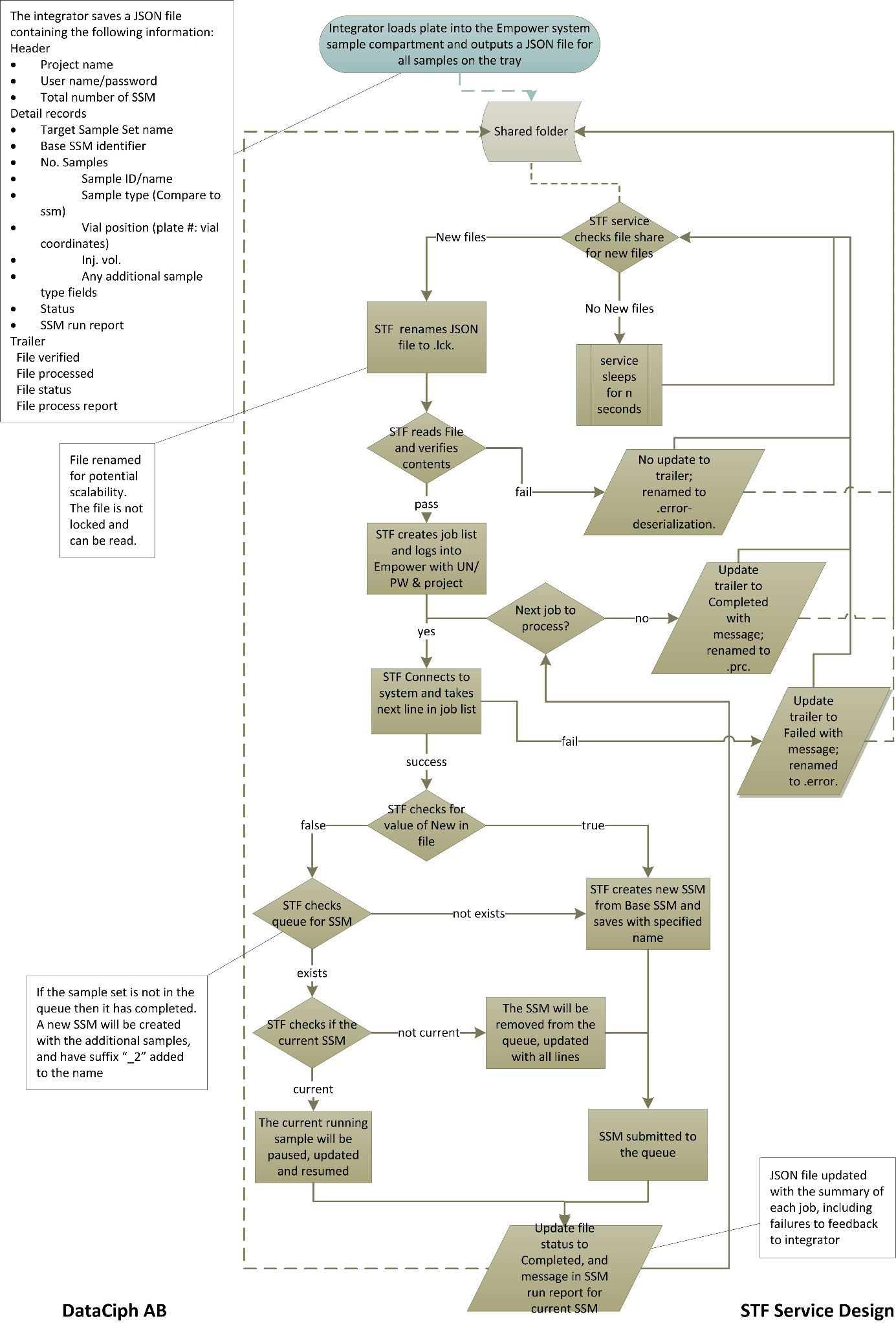
## The STF example app design premise

### Overview

The STF example application would use “new” JSON files as the method for transferring metadata between an integrator application and Empower. The STF was designed to periodically check a shared folder location for new JSON files and then process these files. The STF deserialises the JSON file and parses the metadata. The STF would then log into an Empower project using the Empower Toolkit and would either create new SSMs from base SSMs that exist in Empower, or update existing SSM’s in the Empower System queue. These SSM’s would then be added to the Empower queue to run on the Portal enabled system that the integrator is handling. The STF example would also write back to the JSON file reports which can be read by the integrator software regarding the success or failure of processing the file or each SSM job.

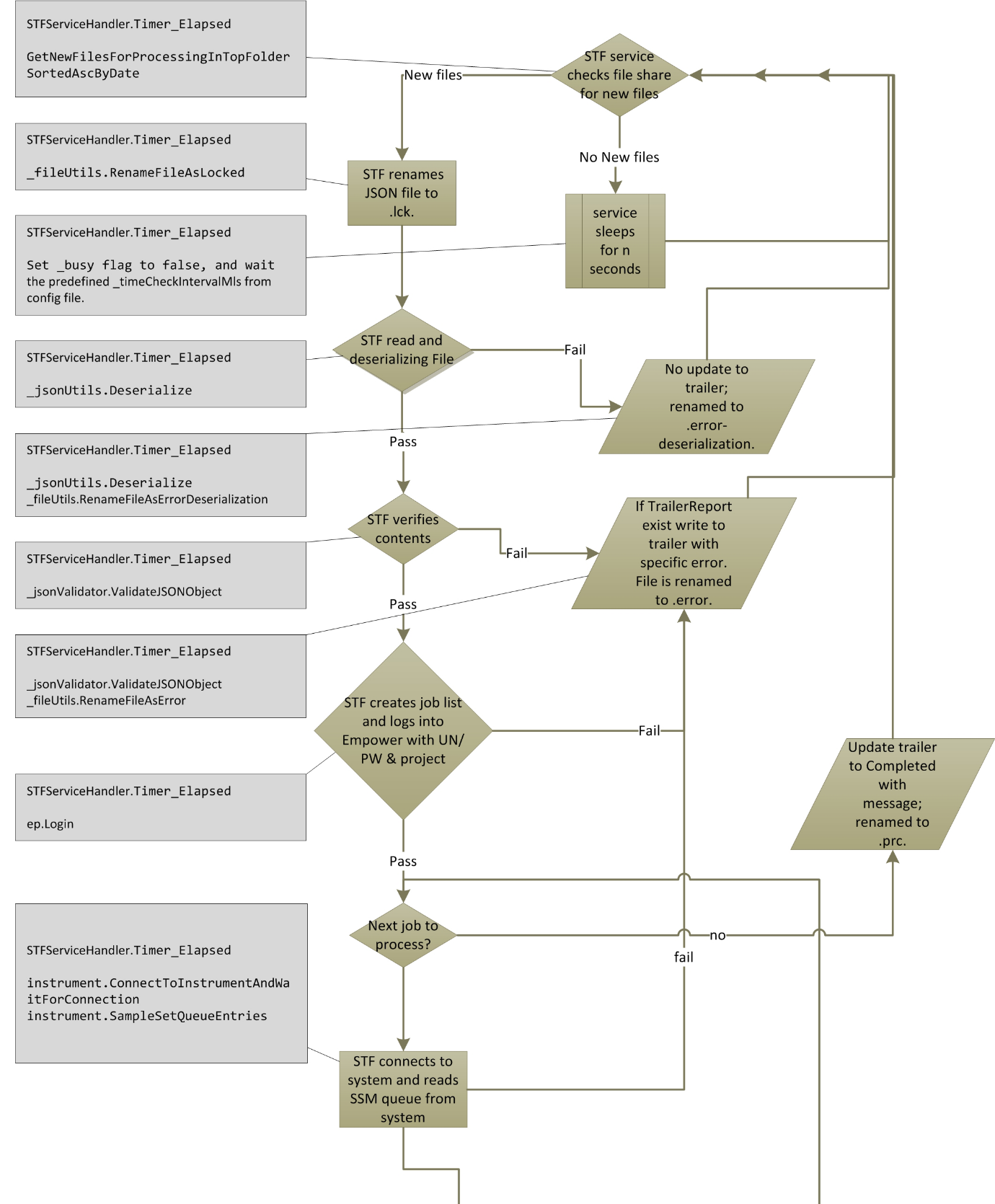
The process can be visualised as the JSON file name progresses from new through lck to prc when successful, or to error when not successful. Additionally, a log file example is provided as a human readable record of the process, and for troubleshooting failures.

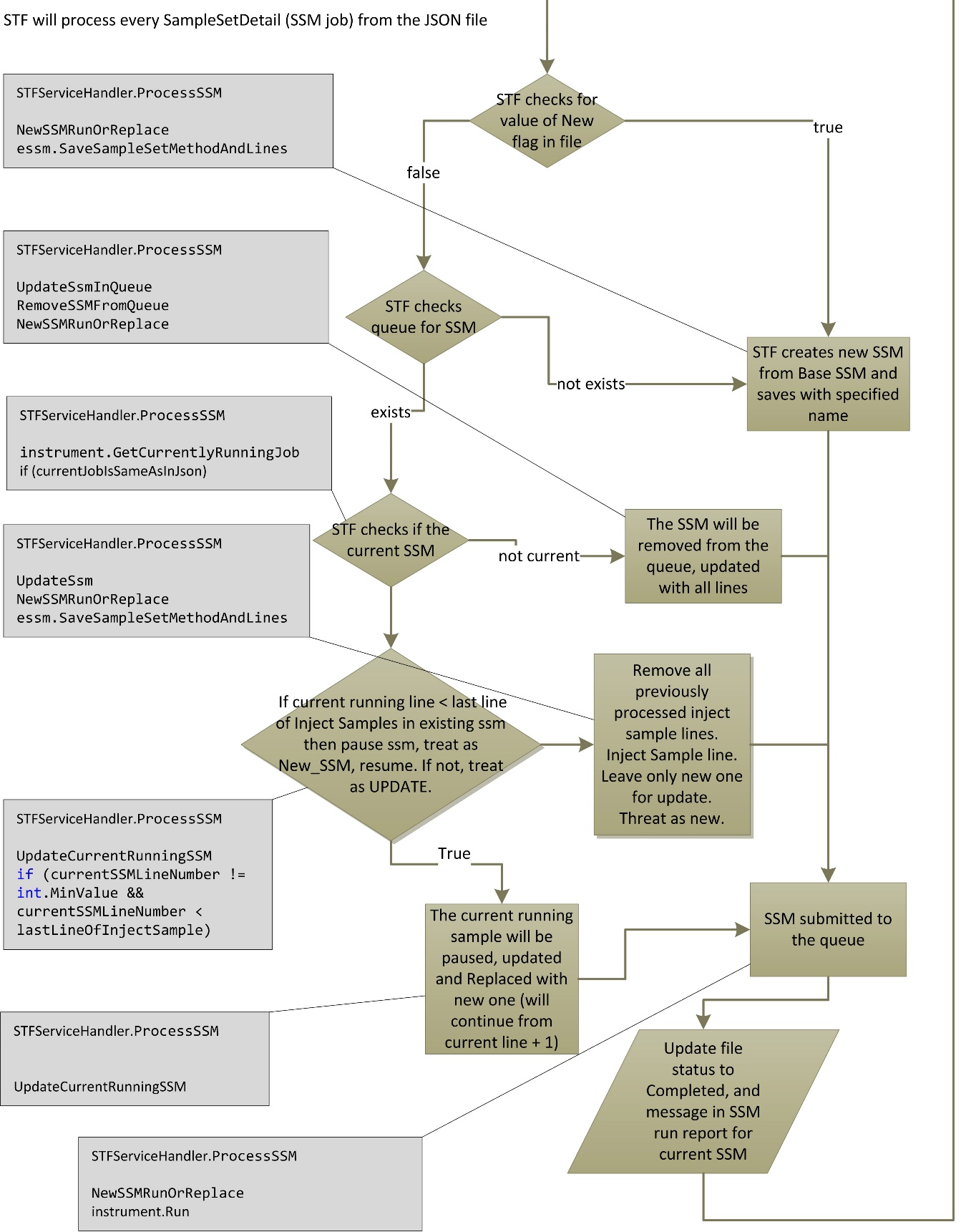
### Designed logical flow

The following diagram provides an overview of the STF process

### Code reference details

The following diagram provides a code reference during the STF process





### Examples of risk considerations in development decisions

|  |  |
| --- | --- |
| **Risks** | **Risk Description** |
| list all steps that need comments, and set default strings. | A step that should require a comment is not configured |
| test pausing an existing run, updating and restarting | Test cannot be paused, updated & restarted |
| managing DLLs to ensure maximum support ability of all apps, minimum carry over of unwanted code | poor lifecycle support |
| Manually create JSON file in format | cannot read JSON file |
| STF service that checks file share on a configurable basis | Service does not check file share/not frequently enough |
| STF service reads json file and verify contents | Verification can only occur on mandatory fields, additional fields cannot be verified and could be incorrect format |
| Read user credentials only from the json file | Cannot read user credentials |
| Set up share and permissions | permissions do not work as expected in 'production' environment |
| Create a separate executable initiated from outside Empower |  |
| error handling and logging for login | Fails to login and not logged |
| the STF service needs to add comments with all saves | A step that should require a comment is not configured |
| Handle issues - The STF service needs to alter a sample set in the queue or a running sample set | Missing error handling in any of multiple steps - connecting to instrument, pausing sample, resuming SSM, updating SSM, saving SSM, removing from queue, adding to queue, updating log files, updating trailer. |
| The STF service needs to send the SSM to the system | service fails to send SSM to the queue without handling |
| The STF service needs to write to the json file | service fails to write to the JSON file |
| The STF service needs to write log files | Log file fails to write |
| The STF service needs to support all run types | Run type set in json file is not compatible |

### JSON File

#### Json file overview

The JSON file contains the information for Empower and Empower login, the specific system used by the integrator and the metadata for the SSM(s).

The following page shows an example “new” JSON file as expected from a proposed integrator application. The naming convention for the JSON file is *<*Integrator\_ID*string>\_<*STF\_ID*string>\_<*creation\_Timestamp*YYMMDD\_hhmm>.*new.json*.* The minimum fields have been used in the samples array in this example. The example STF is designed to handle the samples array where each object transposes to a sample line in the SSM. The samples objects consist of string value pairs that must be the same as the sample type fields in Empower.

{

"HeaderFields": {

"EmpowerProject": "a\_project\\Defaults\_OAI1",

"EmpowerDatabase": "TEST",

"EmpowerUn": "system",

"EmpowerPw": "manager",

"System": "UPLC\_2",

"Node": "LACE\_1",

"SampleSets": 2

},

"SampleSetDetails": [

{

"BaseSampleSetMethodName": "STF\_base\_SSM1",

"SampleSetName": "test\_2610202\_16",

"ExperimentId": 567,

"NumberOfSamples": 3,

"New": true,

"Samples": [

{

"LineNumber": 1,

"Function": "Inject Standards",

"Vial": "1:A,1",

"SampleName": "abc1",

"InjVol": 1.0,

"RunTime": 1.0

},

{

"LineNumber": 2,

"Function": "Inject Samples",

"Vial": "1:A,2",

"SampleName": "abc2",

"InjVol": 2.0,

"RunTime": 2.0

},

{

"LineNumber": 3,

"Function": "Inject Samples",

"Vial": "1:A,3",

"SampleName": "abc3",

"InjVol": 3.0,

"RunTime": 3.0

}

],

"Status": null,

"SampleSetMethodRunReport": null

}

]

"TrailerReport": {

"FileVerified": false,

"FileProcessed": false,

"FileStatus": "Not started",

"FileProcessReport": null

}

}

#### JSON file details

This section steps through the arrays, objects and string value pairs in the JSON file. Please refer to the code to understand how each is used in detail, or if additional data and validation are required.

HeaderFields object:

The object contains the file processing information to log into Empower and the number of sample sets to process.

EmpowerProject: the value is a string and must contain the full path to the Empower project where the base SSM exists and the resultant SSM will be saved and run. Case and spelling sensitive, and only validated on login.

EmpowerDatabase: the value is a string and is the database instance alias used for Empower login

EmpowerUn: the value is a string and is the username for the Empower user required to run the SSM. This user must have suitable privileges and access to perform the required tasks, but also be restricted significantly for security.

EmpowerPw: the value is a string and can be plain text or encrypted. Encryption is determined by the key “EmpowerJSONpwEncrypted” in the configuration file.

System: the value is a string and is the Empower chromatographic system to which the integrator is connected.

Node: the value is a string and is the Empower LACE or acquisition server to which the chromatographic system loaded by the integrator is connected.

SampleSets: the value is an integer and describes the number of sample sets to be created and run in Empower. This number must equal the number of SSM’s described in the SampleSetDetails array.

SampleSetDetails array:

The array contains the details that would be required to update or create a SSM in Empower.

BaseSampleSetMethodName: the value is a string and must equal the name of the base sample set method that exists in Empower. Verified when searched.

SampleSetName: the value is a string and is used as the new SSM created in Empower. If the name already exists in the Empower project, a suffix of \_2 is added to the name. Empower can handle the same name and new versions of SSM’s but this is not designed in the code example. This is also the name used for the resultant sample set. Empower will support different naming for the sample set but this is not designed in the code example.

NumberOfSamples: the value is an integer and must equal the number of samples in the samples array.

New : the value is Boolean (true/false) and determines if the process will follow the flow of a new SSM (true) or update an existing SSM (false). When the value is true, all injected sample rows would be required in the order they are to be injected. When the value is false, the additional injected sample rows would be required in the order they are to be injected; the additional rows would be added to the end of a SSM that could either be in the Empower system queue, or be currently running on the system. The example includes a check that the running sample set has not passed the last inject row. If the SSM would have passed the last row, or be completed (and no longer present in the queue) then a new SSM would be created using the completed SSM as the base SSM, saved with the incremental suffix and sent to the system. It is important to understand what checks are required in the code to prevent incorrect handling under “exception” circumstances, for example, the first SSM failed to run rather than had completed.

Samples array:

This is a nested array within the SampleSetDetails array, describing each sample and containing the metadata for each line of the new SSM. The minimum fields that are required are LineNumber, Function, Vial, SampleName, InjVol and RunTime. Additional fields that are required for the resultant SSM must be added to the samples array in the exact spelling and case as in Empower, or populated in the base SSM. Validation of both field name and format for additional fields are not completed in the STF example and will require careful design consideration to prevent failure.

LineNumber : the value is an integer and an arbitrary field.

Function : the value is a string and must equal the function that is required in Empower. The STF example is designed to handle Inject Standards and Inject Samples, and retain the positions in the final SSM. Non-inject rows at the start and end of the base SSM would also be preserved. Consideration will need to be given to support all functions allowed in Empower to handle exceptions and prevent failure.

Vial : the value is a string and must have the correct format for the plate type used. The plate type must also be present in the base SSM in Empower. Validation back to Empower must be considered to avoid a failure due to incorrect plate type, format and position.

SampleName : the value is a string and is the sample name.

InjVol : the value is a number (float) and is the injection volume in µL for the sample.

RunTime : the value is a number (float) and is the run time in minutes for the sample.

Status : the value is a string and would be either null, or strings Failed or Completed. The status would be null when the file is new, or when the file fails initial steps such as file verification, deserialization or connection to Empower project or system. The status would be Failed when there is an issue with the base SSM or when the SSM could not be saved for any reason. The status would be Completed when the SSM is successfully saved and sent to the system. If there would be a failure due to deserialisation or reserialision of the file, this value would be unchanged.

SampleSetMethodRunReport : the value is a string and would be either null, or a message string. The message string would indicate the success or failure of the creation, saving and submitting of the SSM. The name of the SSM would also be reported. If there would be a failure due to deserialisation or reserialision of the file, this value would be unchanged.

TrailerReport object:

The object contains the processed file information. If there would be a failure due to deserialisation or reserialision of the file, this object would be unchanged.

FileVerified : the value is Boolean (true/false) and would report if the verified fields are correct.

FileProcessed : the value is Boolean (true/false) and would report if the file was processed or not. The STF example design would report true even if it was processed and the SSM failed.

FileStatus : the value is a string and would be strings Not started, Failed, or Completed. The STF example design would report Failed when the file had a failure during processing. The STF example design would report Completed when the file was processed without failure, regardless of whether the individual SSM completed or failed.

FileProcessReport : the value is a string and would be either null, or a message string. The message string would indicate the success or failure of file processing. If the file would have completed, the string would include a summary of number of SSM processed and the number succeeded or failed.

### Configuration file

The configuration file (SFTTest.exe.config) contains the non-hardcoded information that would be required to run the example STF. The following page shows the application settings section of the configuration file and then each key value pair will be described.

<appSettings>

<add key="EventLogName" value="Application" />

<add key="EventLogSourceName" value="STF Service" />

<add key="LogFileNamePath" value="c:\STF\stfService.log" />

<add key="TimerInterval" value="15000" />

<!--5min=1000\*60\*5 (ms)-->

<add key="ServiceMonitorApp" value="STFServiceMonitor" />

<add key="InstrumentConnectionTimeOut" value="4" />

<!-- in seconds -->

<add key="ServiceId" value="01" />

<add key="STF\_JsonDirectoryPath" value="C:\STF" />

<add key="STF\_JsonDirectoryUserName" value="Administrator" />

<add key="STF\_JsonDirectoryPassword" value="EAAAABrr9+MostHC4aldXRZl7DtgTQlxHzNUeCgRlNt4dweN" />

<add key="STF\_JsonDirectoryDomain" value="server08a" />

<add key="EmpowerJSONpwEncrypted" value="false" />

</appSettings>

Here is a step through of the key value pairs in the config file. Please refer to the code to understand how each is used in detail.

#### Event logging

These two keys describe the logging to the Windows event logs.

<add key="EventLogName" value="Application" />

<add key="EventLogSourceName" value="STF Service" />

#### Log file location

To set the location of the log file

<add key="LogFileNamePath" value="c:\STF\stfService.log" />

#### Sleep interval

To set the timer interval in milliseconds for the application to sleep between checking for new JSON files. With automation applications it is important to make sure that iterations do not hang or are long running and then collide or cause a permanent inactive state. The example STF would use a simple busy flag set to true when it starts a job. This flag would then only set to false when the job completes. If the flag remains as busy when the next iteration should start (according to the TimerInterval value) then the STF Example is designed to check the start time of the previous job and determine if the job should continue or has hung.

<add key="TimerInterval" value="15000" />

<!--5min=1000\*60\*5 (ms)-->

ServiceMonitor

<add key="ServiceMonitorApp" value="STFServiceMonitor" />

Not used.

#### Instrument connection timeout

Used to set the instrument connection timeout for connections to systems in Empower. It is not uncommon for systems to be unavailable on the Empower network, either due to exceptions or planned events. The latency of system connections is dependent upon the installed environment, and can vary from one second to one minute. Any resultant code will need to handle system connections that have low or high latency, or do not complete.

<add key="InstrumentConnectionTimeOut" value="4" />

<!-- in seconds -->

#### STF service identifier

Used to identify an STF service and can be used for scalability

<add key="ServiceId" value="01" />

#### JSON file shared folder location and credentials

These keys describe the shared folder where the JSON file would be placed by the integrator application. This could be a local or remote network share. The password is encrypted using the Rijndael algorithm. A domain user with suitable access will be required to read and modify the JSON file.

<add key="STF\_JsonDirectoryPath" value="C:\STF" />

<!--<add key="SFT\_JsonDirectoryPath" value="\\192.168.0.123\test123" />-->

<add key="STF\_JsonDirectoryUserName" value="Administrator" />

<add key="STF\_JsonDirectoryPassword" value="EAAAABrr9+MostHC4aldXRZl7DtgTQlxHzNUeCgRlNt4dweN" />

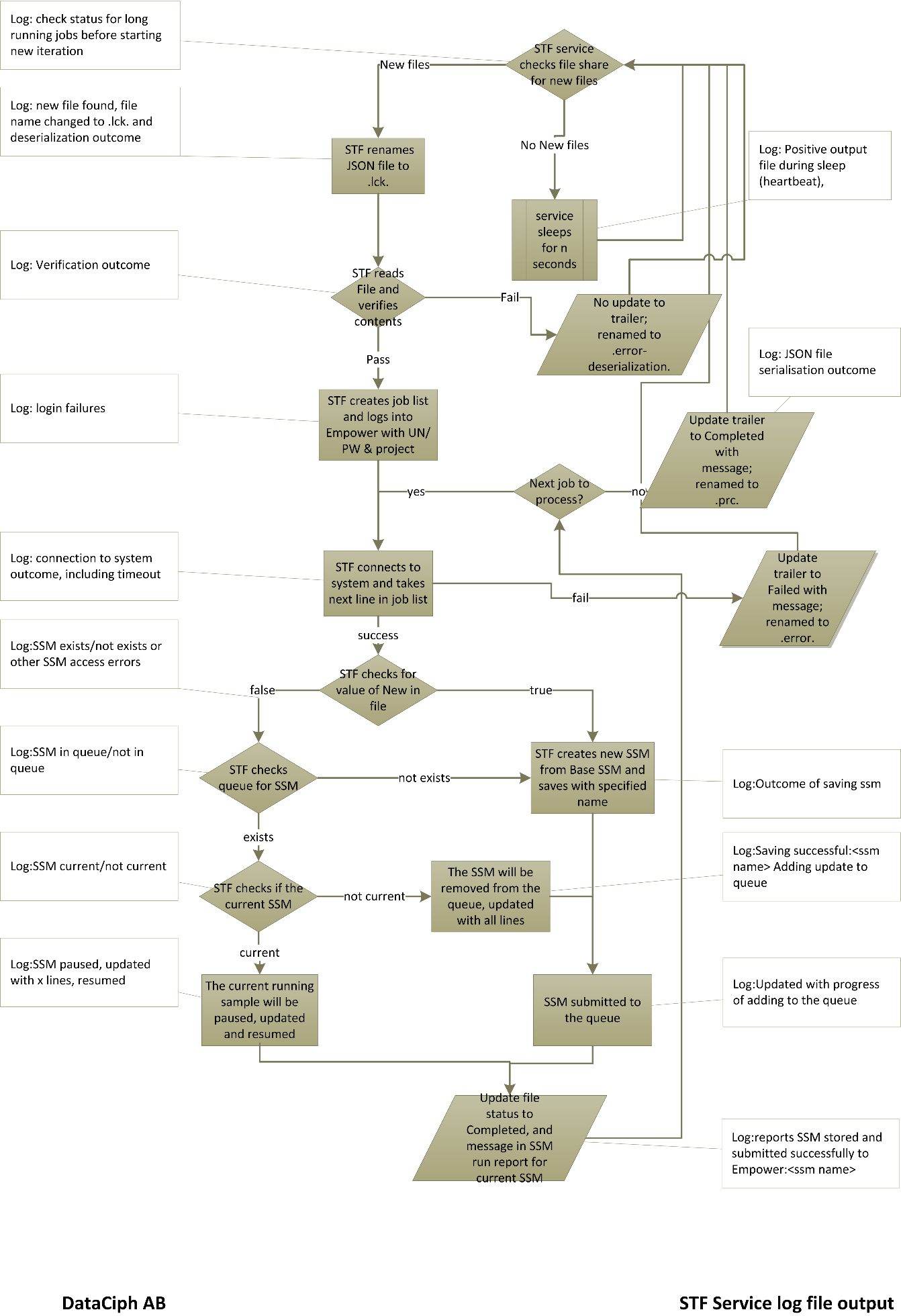
<add key="STF\_JsonDirectoryDomain" value="server08a" />

#### Encryption of the Empower password in the JSON file

Security for the Empower password in the JSON file should also be considered. Passing the password in plain text may not be acceptable. In the STF example, the password could be used in plain text with value set to false or encrypted with the value set to true.

<add key="EmpowerJSONpwEncrypted" value="false" />

### Log File

The log file will form an important component of any file transfer solution, providing user visibility and evidence of interface activity, and reporting of exceptions. Log4net is used in the STF example and the log configuration settings are set in STFTest.exe.config file. The following flow diagram indicates the logging examples provided in the STF:

Example of the log output following successful JSON file processing:

2020-11-11 17:05:03,445 [6] INFO Common.Tools.Loggers.Log4NetLogger - Timer\_Elapsed: service is not busy, there is no need to compare last iteration start time to current time... going straight to STF file process..

2020-11-11 17:05:03,445 [6] INFO Common.Tools.Loggers.Log4NetLogger - Timer\_Elapsed: starting new iteration at..11/11/2020 5:05:03 PM

2020-11-11 17:05:03,445 [6] INFO Common.Tools.Loggers.Log4NetLogger - Timer\_Elapsed: starting... setting iteration busy flag to True...

2020-11-11 17:05:03,445 [6] INFO Common.Tools.Loggers.Log4NetLogger - Starting file deserialization C:\STF\int1234\_standard\_200929\_1429.Tlck.json.

2020-11-11 17:05:03,445 [6] INFO Common.Tools.Loggers.Log4NetLogger - File deserialization C:\STF\int1234\_standard\_200929\_1429.Tlck.json succeeded.

2020-11-11 17:05:03,445 [6] INFO Common.Tools.Loggers.Log4NetLogger - Validating file: C:\STF\int1234\_standard\_200929\_1429.Tlck.json

2020-11-11 17:05:03,445 [6] INFO Common.Tools.Loggers.Log4NetLogger - File verification passed: C:\STF\int1234\_standard\_200929\_1429.Tlck.json!

2020-11-11 17:05:04,131 [6] INFO Common.Tools.Loggers.Log4NetLogger - Common.ConnectToInstrumentAndWaitForConnection(acqServerNode=GrandicM, instrumenName=fake\_2, timeoutSeconds=4)... step in...

2020-11-11 17:05:04,147 [6] INFO Common.Tools.Loggers.Log4NetLogger - Common.ConnectToInstrumentAndWaitForConnection()... loops before timeout: 20

2020-11-11 17:05:04,973 [6] INFO Common.Tools.Loggers.Log4NetLogger - Common.ConnectToInstrumentAndWaitForConnection()... Connection Succeeded

2020-11-11 17:05:07,548 [6] INFO Common.Tools.Loggers.Log4NetLogger - Saving successful:test\_2610202\_16. Adding update SSM to queue.

2020-11-11 17:05:08,905 [6] INFO Common.Tools.Loggers.Log4NetLogger - SSM stored and submitted successfully to Empower: test\_2610202\_16.

2020-11-11 17:05:08,905 [6] INFO Common.Tools.Loggers.Log4NetLogger - Starting file object Serialization for C:\STF\int1234\_standard\_200929\_1429.Tlck.json.

2020-11-11 17:05:08,905 [6] INFO Common.Tools.Loggers.Log4NetLogger - Object serialization succeeded for C:\STF\int1234\_standard\_200929\_1429.Tlck.json.

2020-11-11 17:05:09,030 [6] INFO Common.Tools.Loggers.Log4NetLogger - Timer\_Elapsed: finally... setting busy flag to False

#### Logging of JSON file Process Issues

Specific issues with the process from deserialisation of the JSON file, through SSM creation and running in Empower to reserialising the JSON file, will need to be handled and logged. The STF shows some examples of capturing these errors but does not attempt to create full verbose messages. It is expected that the level and verbosity will be created according to specific requirements.

Here are some typical examples of failure mode logging from the STF example:

* JSON file validation failure due to missing or incorrect header field:

2020-11-11 17:03:48,297 [4] ERROR Common.Tools.Loggers.Log4NetLogger - JsonUtils.cs(75).Deserialize: Error occurred while Deserializing file: , ex=Object reference not set to an instance of an object.

2020-11-11 17:03:48,297 [4] ERROR Common.Tools.Loggers.Log4NetLogger - File not processed - deserialzation failure: Error occurred while Deserializing file: Object reference not set to an instance of an object.

* Incorrect Oracle database alias

2020-11-11 17:03:20,576 [4] ERROR Common.Tools.Loggers.Log4NetLogger - Login() : Login failed.

* Missing or incorrect project name in Empower:

2020-11-11 17:03:21,293 [4] ERROR Common.Tools.Loggers.Log4NetLogger - Login() : Project - Login: Invalid project name.

* Incorrect number of samples during SSM validation

2020-11-11 17:03:21,293 [4] ERROR Common.Tools.Loggers.Log4NetLogger - NumberOfSamples=9 in SampleSetDetailsData is not equal to jSONobject.SampleSetDetails.Samples.Count=8

* Missing or incorrect base SSM method name in the Empower project

2020-11-11 17:03:42,728 [4] ERROR Common.Tools.Loggers.Log4NetLogger - SampleSetMethod 'bad\_base\_SSM1' not exist!

2020-11-11 17:03:42,728 [4] ERROR Common.Tools.Loggers.Log4NetLogger - BaseSampleSetMethodName 'bad\_base\_SSM1' does not exist in the project.

* Locked base SSM in Empower

2020-11-11 17:03:48,095 [4] ERROR Common.Tools.Loggers.Log4NetLogger - EmpowerSampleSetMethod SaveSampleSetMethodLines: Method is permanently locked.

2020-11-11 17:03:48,095 [4] ERROR Common.Tools.Loggers.Log4NetLogger - Cannot save SSM: STF\_locked\_base\_SSM1. Error: Method is permanently locked..

* Missing or incorrect system name in Empower

2020-11-11 17:03:49,062 [4] ERROR Common.Tools.Loggers.Log4NetLogger - Connect : Invalid value.

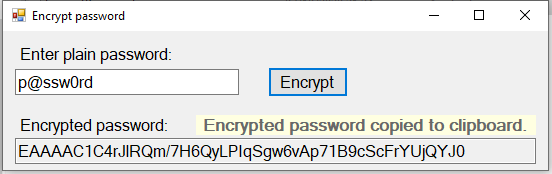
2020-11-11 17:03:49,062 [4] INFO Common.Tools.Loggers.Log4NetLogger - Common.ConnectToInstrumentAndWaitForConnection()... loops before timeout: 20

2020-11-11 17:03:49,062 [4] ERROR Common.Tools.Loggers.Log4NetLogger - Connection status done : Instrument.Connect required.

2020-11-11 17:03:53,134 [4] ERROR Common.Tools.Loggers.Log4NetLogger - fake\_3@Node2: Instrument connection timeout.

### Password encryption

The STF example is designed to handle password encryption as described in the previous sections. To enable the manual encryption of the plain text password, and simple encryption application can be provided. Below is an example of the encryption application:



## Version History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Reason for Revision** |
| 0.1 | 27-Nov-2020 | New draft document |
| 1.0 | 11-Dec-2020 | Review revisions and first version created |
|  |  |  |