HighWire Measurement Integration Recommendations

Table of Contents

Summary

HighWire has the capability to integrate through a vast number of transports, formats, and contents. Please see below for the recommended transportations, file/message/record format, and file/message/record content. These are recommendations not requirements; however, there are user experience impacts. AMP would recommend that the user familiarize the usage of automated delivery and re-run capabilities of HighWire to help decide on integration choices. Before we get into that let AMP share our perspective how the history and future of measurement. Let us also share the best practices of what we have observed from those that execute the job measurement well.

Recommended Transportation (In order of recommendation)

Message Queue (Broker at Amazon or Azure)

Pros

Event based and fire / forget queue-based publishing. When the target system is down the messages will be queue and received when the target system is brought back on-line.

Only one topic required per message type. Proven for years at HighWire customers.

This can scale to the largest of systems. If the asset is larger than 1,000 meters this is highly recommended, larger than 5,000 it is a requirement.

Guaranteed Delivery / Fault tolerance – Allows for target system to process completely before taking message off queue.

Performance – Messaging allows for multi-threaded application systems to scale better.

For HighWire all the benefits of files are kept in place because a file is saved for use through file interface.

HighWire has the capability to send changes as they happen.

Cons

Longer development time, skill of development staff

Infrastructer / cost of queueing system

MQTT

Pros

IOT Brokers are supported 24/7 and secured by Amazon or Azure.

Event based and fire / forget topic-based publishing. When the target system is down the messages will be queue and received when the target system is brought back on-line.

This can scale to the largest of systems. If the asset is larger than 1,000 meters this is highly recommended, larger than 5,000 it is a requirement.

Can deliver to multiple systems the same message.

Performance – Messaging allows for multi-threaded application systems to scale better.

For HighWire all the benefits of files are kept in place because a file is saved for use through file interface.

HighWire has the capability to send changes as they happen.

Cons

Adoption is just starting to get momentum in the industry.

Infrastructure / cost of MQTT/IOT broker.

Files Via Folder Share[1]

Pros

Easier / faster to implement and development

Auditability – The file is a visible / trackable thing from a user perspective.

Usability – Users can utilize the file through the online UI of HighWire to reimport failed interface issues. The files can be changed by a user to fix interface errors.

Cons

Files take up space on the application server and archive managements is required

Users can change files and some consider this to be an Audit issue. However, as long as users have access to just copies this can be mitigated.

The triggering mechanisms to see arrival are depended on Microsoft file API(s). The difficulty is know file writing is complete.

Files Via SFTP[2]

Pros

Same as files above.

Cons

Same as files above.

[1] AMP does not recommend or support a network file share because of MSFT bugs related to file changes on network drives. Unless the file share is from the application server and the source writes to it.

[2] AMP has had great success with SFTP because there are many SFTP tools the customers can use to automatically send new files to the HighWire. This allows for disaster easier disaster recovery via DNS change.

Database

Pros

Easier management from the publisher's perspective. Most developers can write records to a database or the database already exists.

This can scale. If the asset is larger than 1,000 meters this is an ok option. Larger than 5,000 it is a requirement if not using queueing.

Cons

Mostly ends up just simulating what Queueing does. HighWire just queues the result from the polling of a database.

The database must be kept up and retry mechanism must be utilized.

There must be a mechanism to determine whether the target has already seen the data.

Database query performance as the data tables grow must be considered.

Web Services

Pros

HighWire utilizes RESTful web services for its user interface. The API from HighWire's perspective already exist. When HighWire is the source for the integration, many of the integrations are already complete and available.

Cons

HighWire currently does not have measurement integration from others REST API via this transport. Its polling mechanism would have to be enhanced to include REST calls for specific web services.

HighWire must be up during the request from the target.

Transport Format

JSON (Messaging)

Pros

JSON is becoming the most ubiquitous standard for message interchange. It works better in Messages than files. We would recommend CSV when files are involved.

JSON like XML can have parent child relationships.

JSON if kept flattened can be transformed easily into other formats.

Cons

JSON is not as readable by users.

When abused JSON like XML can have needlessly complex parent child relationships.

CSV (Files)[3]

Pros

CSV is the most ubiquitous standard for file interchange. We recommend CSV when files are involved.

CSV with tools like Excel can be the easiest way that a user with less technical capability can deal with interface issues.

Cons

CSV files can be changed by users which could be an Audit issue if they can access the source of the file. We recommend that users only have access to a copy of the file.

Excel (Files)

Pros

Excel is the most ubiquitous standard for file interchange sourced from a user.

Excel is the user interface for creating files it's available on almost every desktop.

Cons

Excel is harder for programmers to create and can require stronger programming skills to achieve.

Excel can change format of some data which can cause problems.

A user can easily change the format to their taste and this can make utilizing in the HighWire interface difficult and sometime has required custom plugin development.

PDF (Files)

Pros

PDF are the most common way that information is sharded amongst counter parties in Oil and Gas

They can be produced when software is creating printed statements and reports.

Users cannot easily change a PDF.

Cons

Hard for software to read and process PDF information. More advanced development practices need to be employed to process these files.

Encrypted Special Binary Format

Pros

No one can read them, and they cannot be changed by users.

Cons

No one can read them, and they cannot be changed by users. There is a better and easier mechanism to achieve the goal of auditability of measurements.

[3] Recommended file naming convention is <MeterReference><YYYYMMDDHHSSMM>.csv example: Meter202404131343.csv (date is time of the file or measurement date) and MM=Milliseconds.

Specialty tools and libraries are required to work with them.

Transport Content

1 record per file / message / database record (Usually delivered as created)

Pros

File not finished writing issues. The larger the file the more likely HighWire mechanisms for making sure the file is finished writing will be defeated.

File management is easier because the user can file the record or records, they need for reprocessing quickly for self-service interface fix!

User rerun through the user interface is much easier to manage.

HighWire's file adapter log viewer viewing and re-run mechanisms works better with individual records per file.

HighWire's multi-threading processing performance will process the files in parallel in the file adapter.

Archiving is applied more accurately as is uses the measurement date if more than one record the measurement date could cross the archive boundary.

Cons

Number of files to manage and archive. HighWire has mostly solved these issues by it's capability to subdivide the files into archive by measurement date.

A files / messages / database records per meter or tank (Delayed delivery)

Pros

AMP strongly recommends this content recommendation when delayed delivery and content is gas hourly, liquid meter interval and tank hourly inventory data.

Almost the same usability as 1 record per file. (See 1 record per file above. **Most** "Pros" apply.)

Performance per meter is increased because the master data is only read once.

Number of files is decreased.

HighWire's multi-threading processing performance will process the files in parallel in the file adapter, polling broker, ticketing, and validations. This does not matter as much on small systems; however, becomes critical on larger systems.

Cons

Archiving is applied less accurately as it uses the measurement date. When more than one record exists the measurement date is determined by the first record.

When the delivery is delayed and the source is something like daily measurements this would end up being 1 record per file per day anyway.

All records per content type (Lab, Tank Gauge, Daily Measurement) of transaction (Delayed Delivery)

Pros

Easier to development.

To locate data user can look at one file.

Easier for a user to reload and entire intervals data. (Weak “Pro” because this should not be required and HighWire can resubmit a large number of records from the “File Adapter Log Viewer.”)

Cons

File finished writing issue is exacerbated.

An issue with one record could cause all records to fail to interface.

Reruns can send a tremendous amount of already processed records.

File size affects memory usage of File Adapter and slows performance.

Determination of archive location by measurement date is less accurate.