FEWS Verification

Architecture and Components

## A diagram of a software project Description automatically generated with medium confidence

Figure 1- System Architecture

Table of Contents

[FEWS / Verification DB / Model Definition and Creation 2](#_Toc163807202)

[Verification WAR 2](#_Toc163807203)

[Code Generation 2](#_Toc163807204)

[Acquire 2](#_Toc163807205)

[Transform 2](#_Toc163807206)

[Model 3](#_Toc163807207)

[Cube 3](#_Toc163807208)

[Deploy 3](#_Toc163807209)

[Data Access Api 3](#_Toc163807210)

[GraphQL 3](#_Toc163807211)

[Web User Interface 3](#_Toc163807212)

[Vue.js 3](#_Toc163807213)

[MongoDB 4](#_Toc163807214)

[Verification 4](#_Toc163807215)

[Study Definitions 4](#_Toc163807216)

[Configuration 5](#_Toc163807217)

[External sources 5](#_Toc163807218)

[Output 6](#_Toc163807219)

[Template 6](#_Toc163807220)

[External Sources 6](#_Toc163807221)

[Open API 6](#_Toc163807222)

[Machine Environment Variables 6](#_Toc163807223)

[FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_AUTHORIZATION\_URI 6](#_Toc163807224)

[FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_CLIENT\_ID 6](#_Toc163807225)

[FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_CLIENT\_SECRET 6](#_Toc163807226)

[FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_ISSUER\_URI 6](#_Toc163807227)

[FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_KEYS\_URI 6](#_Toc163807228)

[FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_REDIRECT\_URI 7](#_Toc163807229)

[FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_SCOPE 7](#_Toc163807230)

[FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_TOKEN\_URI 7](#_Toc163807231)

[FEWS\_VERIFICATION\_DB\_CONNECTION 7](#_Toc163807232)

[FEWS\_VERIFICATION\_DB\_AES\_PASSWORD 7](#_Toc163807233)

[FEWS\_VERIFICATION\_DB\_USERNAME 7](#_Toc163807234)

[FEWS\_VERIFICATION\_SETTINGS\_TYPE 7](#_Toc163807235)

[OAuth2 7](#_Toc163807236)

[MongoDB 7](#_Toc163807237)

[FEWS Archive 7](#_Toc163807238)

[Verification 7](#_Toc163807239)

[BI Connector 7](#_Toc163807240)

[Drdl.yaml 7](#_Toc163807241)

[Drdl.yaml.config 8](#_Toc163807242)

[Diagnostic Model 8](#_Toc163807243)

[Power Query 8](#_Toc163807244)

[Joining 8](#_Toc163807245)

[Deduplication 8](#_Toc163807246)

[Sorting 8](#_Toc163807247)

[Unioning 8](#_Toc163807248)

[Tabular Model 8](#_Toc163807249)

[SSIS 9](#_Toc163807250)

[Bim 9](#_Toc163807251)

[Config 9](#_Toc163807252)

[Timing 9](#_Toc163807253)

[Analytics 9](#_Toc163807254)

[AI/ML 9](#_Toc163807255)

[Power BI, Excel, SSRS, Tableau 9](#_Toc163807256)

[Bokeh, Plotly, Seaborn, MatPlotLib 9](#_Toc163807257)

## FEWS / Verification DB / Model Definition and Creation

This section is composed of components that provide overall configuration, verification study definitions, external and FEWS data acquisition, authentication and authorization, and web-based user interactivity. It is the heart of the system which generates the entirety of code that defines forecast verification studies. The products of this processing are:

* External data acquisition and transformation into the Verification DB.
* Initial transformation definitions from MongoDB to SQL tables.
* Final transformation definitions from SQL tables to Tabular Model tables.
* Overall Tabular Model definitions – including measures and relationships.
* BI Connector and SSIS cube processing configurations.

### Verification WAR

The verification WAR (Web Archive) consists of three major subcomponents - Code Generation, Data Access API, and a Web User Interface.

#### Code Generation

These component subsets are pluggable interfaces that are dependency aware – allowing them to produce a directed acyclic processing graph that defines a parallel batch processing order. Each batch is separated with a barrier requiring its completion before moving on to further dependent processing. Within each batch, a semaphore dynamically limits the parallelism based on the number of available CPU cores. This parallel limit is calculated as twice the number of cores. This oversubscription allows taking advantage of CPU time that becomes available due to blocked IO processes. Each major subprocess below is executed serially in the sequence Acquire -> Transform -> Model -> Cube -> Deploy.

##### Acquire

Acquire retrieves and transforms external data that is not already present in the FEWS Archive. It is intended to be composed of supplemental dimensional metadata that would enhance data analysis, filtering, slicing, aggregation, display, and grouping. It is not intended to acquire large quantities for forecast-observed fact data. That task should be left to the FEWS system to provide via its archive. Examples of data currently extracted for the purpose of enhancing dimensions include Location, Parameter, and Qualifier attributes acquired from the FEWS OpenAPI interface. Each of these transformed data sources are stored with a fews prefix, indicating the source from which they were derived. E.g. fews.Locations, fews.Parameters, and fews.Qualifiers.

##### Transform

The transform processing component produces the BI Connector’s table definitions. Each study has its own set of definitions, with one table being defined per definition file produced. The files are defined in YAML (YAML Ain't Markup Language) and conform to the DRDL schema (Document Relational Definition Language). The table and column names, data types, and data sources are defined in yaml, while the MongoDB ‘pipeline’ query is defined as a JSON (JavaScript Object Notation) array.

Transformation drdl.yaml files are based on templates stored in the Verification database (template.DrdlYaml) and are editable through the Verification UI interface. There is one template per dimension and fact table component, varying by the Verification Study metadata definitions. Since these definitions do not join data themselves into forecast-observed pairs, the fact table components consist of individual forecast, observed, and normal queries to the MongoDB database.

The data table definitions produced by this step are written to the filesystem in a folder defined by the BI Connector’s configuration file. It is important to note that these drdl.yaml definitions will not be used until the BI Connector service is restarted.

##### Model

The model processing component produces the Power Query (M language) expressions that define SSAS Tabular Model cube table partitions. These expressions are also responsible for joining forecast and observed data into symmetric pairs, deduplication, sorting, and unioning data queries from the SQL ODBC BI Connector data source – the definitions of which were produced earlier in the transform processing component.

The Power Query expressions are based on templates defined in the metadata of the Verification database (template.PowerQuery) and are editable through the Verification UI interface. There is one template per dimension and fact table, varying by the tables being queried in the BI Connector.

These model definitions are written back to the Verification database (output.PowerQuery) for later inclusion when constructing the final SSAS Tabular Model.

##### Cube

The cube processing component produces the final SSAS Tabular Model (BI Semantic Model or ‘bim’) file. This JSON file uses the Power Query expressions produced earlier in the Model step and applies them to a cube template stored in the Verification database (templace.Cube). Additionally, the cube security roles and data source definition are applied to the template as defined in the Verification settings collection. This template varies by the study for which it is defined. Therefore, each study defined in the Verification database produces one SSAS Tabular Model cube. The cube template is also responsible for defining a common set of statistical measures that apply to each study.

The JSON produced in this step is stored in the Verification database (output.Cube) for subsequent copying to a shared folder for cube deployment and processing.

##### Deploy

The deployment processing component copies the cube definition JSON (bim files) to the filesystem where they can be read and deployed to an SSAS (SQL Server Analysis Services) server by an SSIS (SQL Server Integration Services) Agent job. Additionally, this step updates the BI Connector’s configuration to point to a destination MongoDB server as defined in the Verification DB’s configuration.Settings collection. This allows for production data to be queried by development processing, for instance.

This step is also responsible for restarting the BI Connector service, making the newly deployed drdl.yaml and configuration files active.

#### Data Access Api

##### GraphQL

Graph QL provides a simple interface for queries and mutations of data, facilitating the interactivity of the Web User Interface with the underlying MongoDB database.

#### Web User Interface

##### Vue.js

The web user interface allows the user to directly edit any collection in the MongoDB Verification database. In its current form, it is a minimalistic interface that ensures full control over the underlying data with minimal validation and no integrity assurances. It is up to the user to understand their data and make working associations with the Study collection objects. It is designed for a professional engineering audience with intimate knowledge of their data and this system.

### MongoDB

#### Verification

The verification database provides a series of collections that provide configuration, verification metadata, and some external dimensional supplemental data for defining Forecast Verification Studies. The database is divided into distinct sections of functionality – Study definitions, templates, output, external sources, and configuration.

##### Study Definitions

Study definition collections are those without a prefix. They each contribute to the Study collection, which is obliged to reference them as its Verification Study Definition.

###### Class

The class collection is a per-forecast or per-forecast-location defined set of breakpoints that divide the forecast and observed measurement data into discrete ranges. The ranges are defined by a single point and a comparison operator (eq, gt, gtw, lt, lte), and are read from top to bottom order. They must be contiguous (non-overlapping, and no gaps) and cover all possible cases.

The purpose of classes in the system is to discretize data into groups that can be used in a true-false contingency matrix. For example, we forecasted high rain, did we observe high rain? The contingency matrix allows for a large variety of statistical formula generation and techniques that go well beyond those available for continuous data analysis alone.

###### Forecast

The forecast collection defines a single forecast that may be composed of multiple filters, typically specified by a moduleInstanceId, parameterId, qualifierId, and encodedTimeStepId. Sometimes it is also necessary to include ensembleId and locationId to arrive at a single forecast. There are two important things to remember when defining a forecast:

1. Multiple filters cannot have overlapping data. Use care to ensure each filter only produces mutually exclusive data sets.
2. Filters must be defined on indexes. Those suggested above are generally sufficient. Only special cases *might* require additional filtering and cause a query to perform a full, unindexed collection scan. If your data is typically large, you will notice a very slow or intractable processing. Be sure any filter you use has representative indexes.

###### LocationAttributes

The LocationAttributes collection defines what attributes from FEWS locations are relevant candidates to be added to the location dimension. This may vary by location and forecast type, so any number of combinations can be made according to your data’s needs and the completeness of location attribute definitions in FEWS.

###### Normal

The normal collection defines a pseudo-persistence forecast baseline for calculating skill scores. It is based on observed data, usually corresponding to that which is defined in the corresponding study’s observed collection.

The filters defining the observed base may be composed of multiple filters, typically specified by a moduleInstanceId, parameterId, qualifierId, and encodedTimeStepId. Sometimes it is also necessary to include locationId to arrive at a single observation. There are two important things to remember when defining an observation:

1. Multiple filters cannot have overlapping data. Use care to ensure each filter only produces mutually exclusive data sets.
2. Filters must be defined on indexes. Those suggested above are generally sufficient. Only special cases *might* require additional filtering and cause a query to perform a full, unindexed collection scan. If your data is typically large, you will notice a very slow or intractable processing. Be sure any filter you use has representative indexes.

###### Observed

The observed collection defines the observations that correspond to a study’s forecast(s).

The filters defining observations may be composed of multiple filters, typically specified by a moduleInstanceId, parameterId, qualifierId, and encodedTimeStepId. Sometimes it is also necessary to include locationId to arrive at a single observation. There are two important things to remember when defining an observation:

1. Multiple filters cannot have overlapping data. Use care to ensure each filter only produces mutually exclusive data sets.
2. Filters must be defined on indexes. Those suggested above are generally sufficient. Only special cases *might* require additional filtering and cause a query to perform a full, unindexed collection scan. If your data is typically large, you will notice a very slow or intractable processing. Be sure any filter you use has representative indexes.

###### Seasonality

The seasonality collection divides a year into groups corresponding to annual climate changes, operational periods, or other annually recurring segments of time that would be of interest in forecast verification. The ranges are defined by a single point and a comparison operator (eq, gt, gtw, lt, lte), and are read from top to bottom order. They must be contiguous (non-overlapping, and no gaps) and cover all possible cases. Examples of existing seasonality include meteorological and pool-level seasons. The results of these definitions are stored in the observed and forecast date dimensions.

###### Study

The study collection defines which forecasts and observed values are to be correlated for analysis. Additional metadata defined here includes class definitions, normal forecast generation, seasonality, location attributes to include, and the forecast data range of data to retrieve. Additionally, this collection defines whether to use local or UTC time and local or SI units. The cube template to be used is also defined here, although there is currently only one definition (Default).

##### Configuration

###### Description

The description collection contains the definitions of all entries in the Settings collection.

###### Settings

The settings collection contains all the settings the system uses *except*:

1. OAuth configuration, which is stored in machine environment variables.
2. Base connection string to the Verification database containing configuration.Settings – also stored in machine environment variables.

##### External sources

###### fews.Locations

The location attribute metadata extracted from FEWS OpenAPI web service.

###### fews.Parameters

The parameter attribute metadata extracted from FEWS OpenAPI web service.

###### fews.Qualifiers

The qualifier attribute metadata extracted from FEWS OpenAPI web service.

##### Output

###### Cube

This collection stores the final cube bim (BI Semantic Model) JSON data for deployment to SSAS Tabular models. It will eventually be copied to a share folder accessible by SSIS for processing.

###### PowerQuery

This collection provides intermediate storage for the Power Query (M) expressions that define the table partitions in the cube bim (BI Semantic Model) template.

##### Template

###### Cube

This collection contains a pseudo-templatized empty bim JSON defined model that is read as a document object and filled according to the metadata definitions contained within each active Study collection entry.

###### DrdlYaml

This collection contains a text line array templatized representation of dimensional and fact-component (forecast, observed, and normal) table definitions for the BI Connector. It is filled according to the filters defined in the Study collection. See the Transform processing discussion for more information.

###### PowerQuery

This collection contains a text line array templatized representation of dimensional and fact table definitions used to define table partitions within an SSAS Tabular Model cube. It is populated according to the filters defined in the Study collection with queries referencing those in the BI Connector. See the Model processing discussion for more information.

### External Sources

External Sources is a placeholder indicating that other valuable metadata from reference data management (RDM) or other sources of interest might be included in the dimensions of a study for improving data analysis, filtering, slicing, aggregation, display, and grouping.

### Open API

This REST Web Service provided by FEWS allows access to location, parameter, and qualifier metadata, as well as other internal FEWS operations.

### Machine Environment Variables

These environment variables provide the initial seed configuration for OAuth and MongoDB configuration.Settings connections.

#### FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_AUTHORIZATION\_URI

e.g. https://login.microsoftonline.com/00000000-0000-0000-0000-000000000000/oauth2/v2.0/authorize

#### FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_CLIENT\_ID

e.g. 00000000-0000-0000-0000-000000000000

#### FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_CLIENT\_SECRET

e.g. iV\_000000000000000000000000000000000000

#### FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_ISSUER\_URI

e.g. https://login.microsoftonline.com/00000000-0000-0000-0000-000000000000/v2.0

#### FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_KEYS\_URI

e.g. https://login.microsoftonline.com/00000000-0000-0000-0000-000000000000/discovery/v2.0/keys

#### FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_REDIRECT\_URI

e.g. {baseUrl}/login/oauth2/code/{registrationId}

#### FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_SCOPE

e.g. openid

#### FEWS\_VERIFICATION\_AUTHENTICATION\_OAUTH2\_TOKEN\_URI

e.g. https://login.microsoftonline.com/00000000-0000-0000-0000-000000000/oauth2/v2.0/token

#### FEWS\_VERIFICATION\_DB\_CONNECTION

e.g. mongodb://localhost:27017/Verification?tls=true&authSource=admin

#### FEWS\_VERIFICATION\_DB\_AES\_PASSWORD

e.g. 00000000000000000000

#### FEWS\_VERIFICATION\_DB\_USERNAME

e.g. fews\_admin

#### FEWS\_VERIFICATION\_SETTINGS\_TYPE

Optional setting (usually omitted) that overrides the machine name lookup in configuration.Settings. e.g. Prod

### OAuth2

OAuth2 is used for authentication and authorization for Vue.js and the underlying GraphQL service.

## MongoDB

### FEWS Archive

This is the archive data store for FEWS and provided the forecast and observed data for Verification. For each forecast (and its timestep) there must be a corresponding observation timeseries on the same timestep for comparison. Additionally, non-equidistant timeseries are not supported. Use FEWS data processing options to snap any of there timeseries to an aggregate, fixed timestep for Forecast Verification analysis.

### Verification

The verification database provides a series of collections that provide configuration, verification metadata, and some external dimensional supplemental data for defining Forecast Verification Studies. The database is divided into distinct sections of functionality – Study definitions, templates, output, external sources, and configuration.

See the earlier discussion of the verification database for more detail.

### BI Connector

The BI Connector provides a SQL ODBC (MySQL-like) table-based abstraction of the MongoDB document (JSON) database. This is accomplished through a BI Connector MongoDB Enterprise service.

#### Drdl.yaml

The drdl.yaml file provides a definition to flatten data into a row-column format that common analytics tools can use natively. YAML (YAML Ain't Markup Language) conforms to MongoDB’s DRDL schema (Document Relational Definition Language). The table and column names, data types, and data sources are defined in yaml, while the MongoDB ‘pipeline’ query is defined as a native JSON (JavaScript Object Notation) query array.

#### Drdl.yaml.config

This file defines the target MongoDB server, the drdl.yaml folder of definition files, and various other security and logging configuration concerns that the system is not interested in. The purpose of manipulating this file is to change the destination MongoDB server so that PROD data can be extracted for testing in DEV, for instance.

## Diagnostic Model

The diagnostic model provides users with real-time what-if data exploration across a wide range of forecasts and metrics in real-time. It enables users to explore data for interesting anomalies and their causes as well as gain a more complete understanding of data to assist in feature selection for AI/ML tasks.

### Power Query

Power Query enables the real-time streaming transformation of data from MongoDB (and its BI Connector) to SSAS Tabular Model cube ingestion. Power Query (M language) expressions define SSAS Tabular Model cube table partitions. These expressions are also responsible for joining forecast and observed data into symmetric pairs, deduplication, sorting, and unioning data queries from the SQL ODBC BI Connector data source.

#### Joining

Joining data occurs only during the ‘fact’ table generation of forecast-observed pairs. The join occurs between the final forecasts (including *normal*) and observed data tables on the columns event time and location. It is assumed that the user defined each forecast and the corresponding observed filters to be mutually exclusive with respect to location. i.e. no location can be represented twice, in multiple filters that define a single forecast or observed data definition entry.

#### Deduplication

Deduplication is performed by first sorting data (see sorting below) such that a sequential scan of the data will present the ‘preferred’ row of a potential duplicate first and therefore have that first row retained in the final dataset, rejecting all subsequent occurrences. Deduplication occurs over the set of columns as follows: forecastName, location, ensemble, ensembleMember, forecastTime and eventTime.

#### Sorting

Pre-sorting data is for the purpose of arbitrary presentation convenience and bears no direct functional impact on the system’s operations – ***except*** in the case of forecast deduplication. Simulated forecasts may have duplicate data with respect to dispatch time. Therefore, we include a descending sort on dispatch time to ensure that it is presented first to the deduplication processing and, therefore, retained. Sorting occurs over the set of columns as follows: forecastName, location, ensemble, ensembleMember, forecastTime, eventTime, filterForecastTime (Descending), dispatchTime (Descending). [filterForecastTime] is a transient column containing the original forecast time of the forecast timeseries. Since forecast times do not always occur on a cardinal timestep, they must be ‘snapped’ to the nearest one. This ‘snapping’ is effectively an aggregation, where two forecasts could be collocated to the same cardinal timestep due to having a similar proximity to it. The original forecast entry’s forecast time is, therefore, used to resolve the ‘preferred’ instance to retain.

#### Unioning

Unioning data is used in combining multiple forecasts and is equivalent to SQL’s UNION ALL operation. It is simply used to produce a single dataset of multiple forecasts, differentiated by their forecastName column.

### Tabular Model

The SSAS Tabular Model is the heart of the real-time analytics provided by the Forecast Verification System. It incorporates the *exact* same technology as is used by Power BI and its corresponding cloud-based XMLA premium components. It is included as part of Microsoft’s SQL Server product at no additional cost. The usefulness of this toolset is speed, reusable measures, reusable data schemas, and reusable hierarchies and metadata. The number and variety of existing toolset that can connect to SSAS cubes makes it a convenient choice for diagnostic analytics.

### SSIS

SSIS provides decoupled cube deployment and processing of SSAS Tabular Models that operates with minimal references to the parent Verification system. In fact, it only relies on only two dependencies to run:

1. A JSON serialized copy of configuration.Settings.
2. A series of cube bim files that represent updated models to deploy and process.

#### Bim

The bim folder contains a series of completed SSAS Tabular Model cube definitions from the output.Cube collection of the Verification Database. The folder must be accessible by both the SSIS Agent process and the Deploy process of the Verification WAR site. This location is currently on the \*.mc1 share.

#### Config

The config file represents a JSON serialized copy of configuration.Settings produced during the Deploy process of the Verification WAR site. It is placed in the same folder as the bim files. The *execute* Boolean key value provides the only synchronization between the FEWS Verification System and SSIS/SSAS implementation. The value is true when a new config file is written and subsequently set to false when the SSIS job is completed. A new config file will not be generated until it sees this false appear in the config file – thus synchronizing the operations between the two decoupled processes.

#### Timing

This file (timing.json) is written by SSIS to capture each SSAS Tabular Model’s processing time. This is done to ensure that the longest running, bottleneck processes are executed in descending order, giving them the greatest chance of completing in N(max) time rather than an otherwise cumulative processing result. It uses the same folder as the bim files.

## Analytics

Some analytics tools or use cases for Forecast Verification.

### AI/ML

The system can be used to help identify interesting features that have decision making power for inclusion as attributes in AI/ML processes like neural networks. The cube or its underlying data sources can easily be used to provide data to these models – making a perfect first step in AI/ML development.

### Power BI, Excel, SSRS, Tableau

These tools can connect directly to SSAS Tabular models, out-of-the-box. This provides convenience and speed of analysis to answer questions quickly with zero development time.

### Bokeh, Plotly, Seaborn, MatPlotLib

These tools provide precision reporting and advanced graphical models customized to specific use cases. Although not executive friendly, many engineer-programmers are still able to take advantage of these technologies without the need for IT intervention.