Functional Data Platform Requirements v0.1

Overview

Functional data is a general term that describes data that characterises the function of the engine. This comprises of gas and materials properties that are generated by physical engines/components or virtual models as well as configuration files use to execute models and analyse results.

There are many different functional disciplines that either align to a component like compressors/turbines, a sub-component like a blade or cover a whole engine view like performance, fluids.

These functional areas execute a variety of tools that read and write data to perform their various business processes. Across all functions there are 1000s of these tools.

The data used by these tools is persisted on a range of stores that include mainframe databases, NAS file shares, local drives and SQL databases. It is this complexity that makes it very difficult for users to find data.

All new data systems must implement a security model that controls data access according to the user and the data's classification. This has proved a barrier to introducing any new data systems because the security model rules are complex.

FDM has been developed over recently years to address these problems. The introduction of the FDM API intends to isolate all systems and tools from any data storage implementations. To do this FDM has introduced a data model that aims to accommodate the requirements of the many functional areas.

With the introduction of the Functional Data Platform, and the intention to use MindSphere, this document attempts to summarise the requirements elicited during the development of FDM so they can be influence MindSphere.

Data Model

A Data Item is a record in FDP that represents a unit of data. Each Data Item will always have meta-data and most Data Items will have content.

The following sections outline the various features of the data model.

Data Classification

Each Data Item will define its data classification and this will be used to evaluate whether specific users can access the content using an Attribute Based Access Control Model (ABAC). It is assumed initially that the meta-data will not contain any sensitive data but it is likely meta-data will contain sensitive information and access will need controlled in the same way.

The following describes the meta-information that each Data Item will need to capture.

Name	Type	Comment
businessSensitivity	Enum	NON_CONFIDENTIAL,PRIVATE,STRICTLY_PRIVATE

countryOfOrigin	String	Country code the data originates from
ownership	Enum	ROLLS_ROYCE_CONTENT_ONLY, ROLLS_ROYCE_CONTENT_WITH_OTHER_RESTRICTIONS, OTHER_PARTIES_DATA
exportControlled	String	NOT_SUBJECT_TO_EXPORT_CONTROL, TECHNOLOGY_FOR_EXPORT_CONTROL
govSecClass	String	NO_CLASSIFICATION
authorisations	String[]	Ids of ADAs that have been applied to this data
exportClassification	String[]	List of export ratings, required only if export controlled.
otherOwnership	String[]	List of companies, required if not only RR owned

Identity

All Data Items in FDP will have a unique id e.g. guid that never changes. However all data in FDP will need to be uniquely identifiable with a human readable identifier. There should not be two data items with the same identifier in the same context. There should be no confusion on

Example

Engine test data is generated during an engine test and whilst the first scan will have the same identifier of "1" each will be unique in the context of a Real Time Test.

General Meta Data

The following meta-data attributes that have been deemed common enough to be defined on all Data Items.

Name	**Type**	**Comment**
description	string	
keywords	String[]	
typeId	String	Id of the data type
date	Date	The primary date (UTC) associated with the data, meaning would vary for types.
utcOffset	String	This captures the timezone when the data

Meta Data Rules

All data items in FDP should be related to a data type definition that categorises. A Data type is in effect a formal file extension that carries additional information about the type. Without some structure a data store that accepts any data will get out of control.

Data Type Meta-data

Name	**Type**	**Comment**
name	String	Display name of type
description	String	Describes the type
owner	String	The owner of the type
documentationUrl	String	Link to any documentation
iconId	String	Id to data item that is a suitable icon
schemaId	String	Id to data item that is a suitable

		schema

No data can be stored without a defined data type.

A Data type will enforce various rules about all instances of that data.

- Define both mandatory and optional attributes (Extensible)
- Define rules about how the data is named (identifier)
    - All Test and Measurement (T&M) types need to be named in a certain way.
- Define rules about how the data's context should be captured
    - All T&M types must have various contexts defined (See example)
- Define rules on primary context that ensure unique identifiers
    - Steady State data is named 1,2,3 but must be unique in the context of an RTS Test.
- Define rules about the content and the format
    - Some types must have 0 or 1 content items and at least one must be use hdf5 format.

Extensible Meta-data

To support all data types a data item must support additional attributes. These are key-value pairs with the following value types supported:

- String
- String []
- Double
- Integer
- Date
- Boolean

Life Cycle

A typical day of a functional engineer partially involves editing files that define their model, how it is run and how to process the results. At this stage the data is considered as Work In Progress (WIP). Once the model/files are suitable to be shared more widely they are Released and effectively frozen so they can't be changed.

All data in FDP should support the lifecycle of the data.

- WIP – Users can edit
- PRE-RELEASE – No one can edit but it can be reverted to WIP
- RELEASE – No one can edit.

Data items can't be released when the reference data items that aren't

Relationships

Data Items need to relate to other Data Items in various ways. These are:

1. Audit

2. References
3. Context

Audit

Data Items are either created from scratch or generated as output from a process. For the latter it is important that the system records what process was run and holds references to all the inputs used to run it. This directed graph effectively captures the data's lineage and provides answers to questions like:

- Where was this model used?
- How long did this design iteration take?

The system shall support a reference to another data item that represents an audit of how it was generated.

References

When engineers define models, they often break aspects into re-usable blocks or components that mean similar models can share the same component definition. Historically legacy systems, with memory constraints, meant the blocks were quite small and a performance model could easily have 2-3000 blocks. Even now performance models can reference 3-400 other data items.

A data item shall support multiple references with other data items along with meta-data about the relationship.

Context

All data must be associated with multiple contexts. See Context for more details.

Revision

There are two approaches to working with data.

1. Engineers extract files locally, work on them locally and then store them.
2. Engineers work with the data directly in the system and always store all changes.

As engineers edit files and run models it is important that all required changes are preserved so that any stored results capture a valid audit.

The system shall preserve a copy of the meta-data and content for all required changes until it is released.

Context

All data items will be associated with multiple contexts. For example, all T&M data items must have the following contexts:

- Programme
- Project
- Application Project
- Facility
- Company Site
- Vehicle

- Vehicle Build
- Part Test
- Experiment Item
- Operating Item
- Test Plan
- Test Execution Instruction
- Test Execution Instruction Procedure
- Test Execution Instruction Activity
- Test Activity
- **RTS Test** – Primary context for uniqueness

Contexts are then the primary means to search for and filtering data. Data can exist in multiple contexts and no formal hierarchy is implied.

Context is not just an attribute or a simple pick list since it will also have meta-data and possibly content.

In some cases, context is also used to control access i.e. engineers are added to the XWB project so they can see the XWB data.

Data Retention

These attributes define the retention category of the data.

- CATEGORY_A – Store for the life of the engine + 10 years
- CATEGORY_B

Along with the following additional attributes:

Name	Type	Comment
retentionSchedule	Enum	See above
reviewPeriod	String	The period before the retention should be reviewed
reviewDate	Date	The next review date for continued retention

There is no requirement to implement any data retention logic at this stage.

Content

Content is the file or files that the meta-data describes although some data items will not have content. At its core content is just bytes and the byte stored will be the same bytes returned.

Each content will have additional meta-data.

See Content Services

Search

As stated earlier the ability to find data is a critical in a large data system. To facilitate effective search all Data Items must have comprehensive meta-data. When search results are not guaranteed to be 100% up to data the system should provide an additional query mechanism that is always up to date.

Example

A user adds a new data item to the system and the UI would run a search to display the new record immediately. If that record is not shown immediately when using search it would have to use an alternate query mechanism that is.

Basic Search

It should be possible to find data using free text that matches any meta-data. Results should be scored according to best match.

Facets

A search results should also return a count of most common attributes values (or facets) so that large results sets can be refined further using filters derived from this attribute values. The search logic should provide the ability to filter against all meta-data.

Content Search

The ability to search within the content but any content shown should not expose any data that doesn't conform to the security model.

API

FDP should provide a common interface to all data that is implementation neutral. This interface should be robust and long lived since it will be linked into many systems that will be long lived.

At its core access to data should be provided by a suitable interface that has SDKs for all key languages used in RR.

The API should support the following data access interfaces:

- Query all data items
- Get a single data item by its id
- Get a single data item by its identifier
- Get many data items by their ids.
- Get a specific content item.
- Update Data items
- Created Data Items
- Change lifecycle

All interactions should also be exposed via an API i.e. Data Type handling etc but these are the important ones for data access.

Content Services

Whilst access to the content is a key feature of FDP additional services should provide more fine-grained access to the content so that access to the right information is as simple as possible. Whilst

this content services are not an immediate requirement they have been included to provide visibility of future scope.

There will be lots of these services and in simple terms we want to minimise what users need to download and the amount of effort they need to go through to get to the information they need.

Data Table

This is one of the most common types of data typically viewed as a table or spreadsheet. Columns represent parameters and rows represent an individual point in time or a specific condition.

All parameters carry extensible meta-data and as well as value support uncertainty and quality data. One of the key attributes of a parameter is the definition of its units.

Whilst there are many different types of file format that support tabular data most don't support the full set of meta-data. As a result, the main file format to represent this data is HDF5.

Engineers need easy access to this tabular data without having to always handle HDF5 files directly so this service would support access to this information in the following way:

- Access to individual parameters, rows and cells
- Access to the data translated into a specific units system
- Access to the information transformed into a number of other formats csv,tsv,html etc

Data Model

A data type is a high-level definition of how a particular data item fits in FDP, this is effectively the coarse-grained data model. However, in many situations there is a need to align a particular data item's content to a fine-grained data model. An example of this would be an xml or json with a corresponding schema (xsd/json). This fine-grained data model would ensure that the content of each data item instance would conform to the applicable schema.

This fine grained data model would be used to represent model data that is organised as a hierarchy or tree of parameters. These trees would be used by many systems to share information between systems.

Engineers need easy access to these parameter hierarchies so the service would need to support access in the following way:

- Direct access to any node in the tree
- Access to the data translated into a specific units system
- Ability to validate a data model against the corresponding schema

Data Structures

A data structure is used to construct a file/folder hierarchy from various data items in FDP. The data item content contains a json representation of the structure allowing users to check out all the files to replicate the structure on the local file system.

Additional endpoints:

- Access to any node of the structure
- Easy access to all the referenced dataitems

With additional cli tools this service extends FDM to provide git like capability like pull/push etc.

Data Archives

A data archive is effectively a compressed file i.e. zip. The service ensures the content has the right format and exposes additional endpoints.

Additional endpoints:

- Access to the structure or contents of the zip file
- Access to individual files from within the zip.

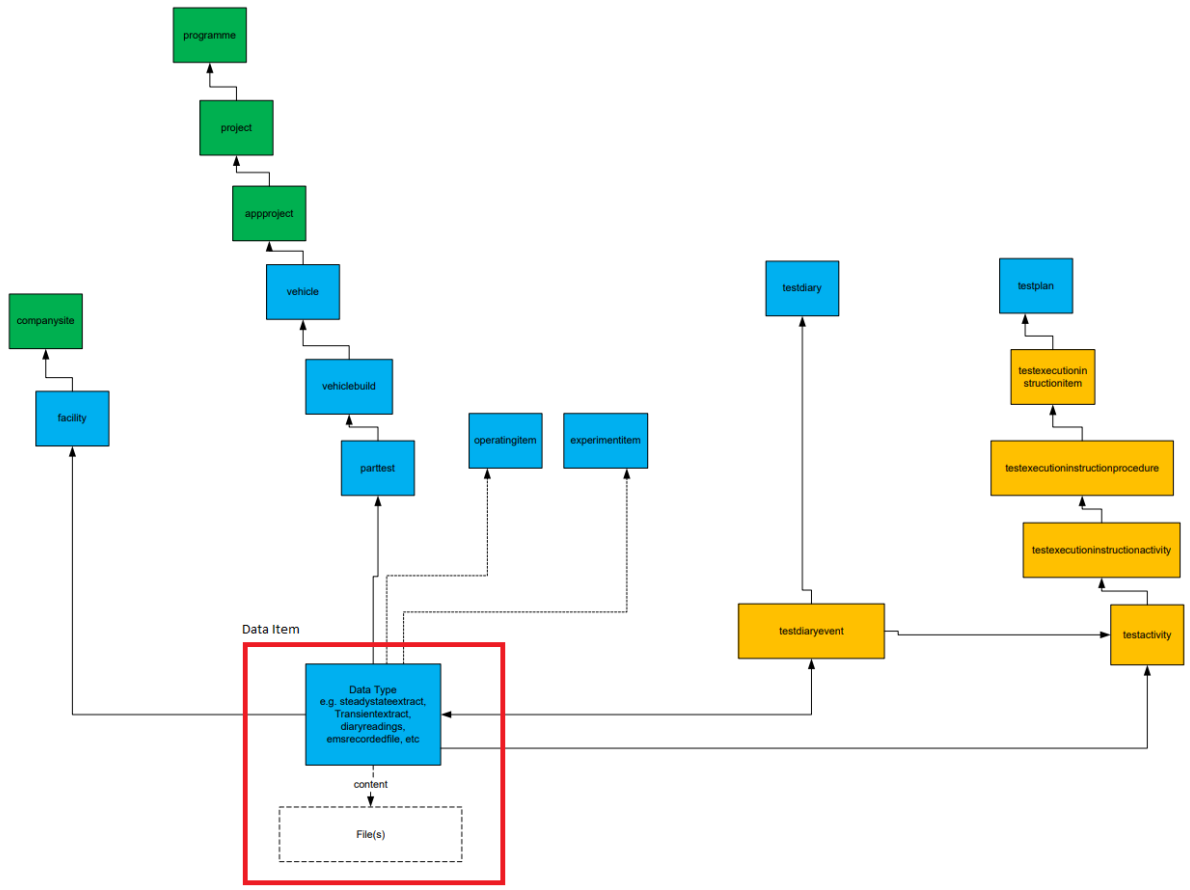
Examples

To provide context here are couple of use case examples.

1. Test & Measurement are using this via FDM in production
2. Blade File is still a concept

Test & Measurement

Test and measure have defined their data model that provides an example of how this is used.



This is the high level view for T&M and how all T&M data items (types) are stored against the various contexts. (Ignore colours for now)

Then picking one of the 70+ T&M data types, Steady Stated Extract, the following table shows the original summary of that type. (Needs rework)

Meta-data	Description	Source	Primary	Required
id (built in data item field)	GUID of the item	Allocated on first creation	No	Mandatory
identifier -> name (built in data item field)	The DAS recording name for the scan e.g. 1, 001, etc	From scan	No	Mandatory
resourceId (built in data item field)	The resource id of a dataitem can be thought of like the file extension for the item.	"datatables"	No	Mandatory
lifeCycle (built in data item field)	The current life cycle state of a dataitem	Initially Set to "WIP", changed to "PRERELEASED" or "RELEASED" when validated	No	Mandatory
dataVisibility (built in data item field)	The current visibility state of the DataItem	"VISIBLE"	No	Mandatory
retentionCategory (built in data item field)	Retention category of data item	From TEDS download or based on rule for Application Project and Site	No	Mandatory
retentionPeriod (built in data item field)	Retention period for data item (years)	From TEDS download or based on rule for Application Project and Site	No	Only if Category B
reviewDate (built in data item field)	Review date for data item	From TEDS download or based on rule for Application Project and Site	No	Only if Category B
date (built in data item field)	The date and time of the start of the scan	From scan	No	Mandatory
comments (built in data item field)	A comment from the DAS associated with the scan	From scan	No	Optional
country (string attribute)	Country code the scan was recorded in	Based on lookup based of test to companysite -> country	No	Mandatory
scantype (string attribute)	The equivalent of the Manoeuver Extraction Point in ALICE, e.g. P01	From scan	No	Optional
duration (double attribute)	The duration the scan was averaged over (secs)	From scan	No	Optional
[Parameters] (array of strings attribute)	The parameters in the scan	From Scan	No	Mandatory
accessSecurity -> businessSensitivity (built in data item field)	Business sensitivity	From TEDS download or based on rule for Application Project and Site	No	Mandatory
accessSecurity -> businessOwnership (built in data item field)	The legal corporate entity(ies) that own the data	From TEDS download or based on rule for Application Project and Site	No	Mandatory
accessSecurity -> otherOwnership (built in data item field)	If not RR Only must list other parties names	From TEDS download or based on rule for Application Project and Site	No	If Applicable
accessSecurity -> exportControlled (built in data item field)	This attributes identifies whether the data refers to technology that is subject to export control	From TEDS download or based on rule for Application Project and Site	No	Mandatory
accessSecurity -> exportClassification (built in data item field)	If export controlled must define list of applicable ECCNs in the format that encodes the country and control list as defined in Capricorn eg. NULL / UKDL_PL9001.C / USML22.A / UKDL_PL8001.C/ DEDL_2E002/ ... AND MORE	From TEDS download or based on rule for Application Project and Site	No	If Applicable
accessSecurity -> govSecClass (built in data item field)	Not suitable for any government classification but all data must be explicitly marked	From TEDS download or based on rule for Application Project and Site	No	Mandatory

association to programme data item / security group	Programme this is part of. Also determines which RBAC rules apply to this object	GUID of relevant programme data item / security group based on lookup of test id to programme.	No	Mandatory
association to project data item / security group	Project this is part of. Also determines which RBAC rules apply to this object	GUID of relevant project data item / security group based on lookup of test id to project	No	Mandatory
association to appproject data item / security group	Application Project this is part of. Also determines which RBAC rules apply to this object	GUID of relevant application project data item / security group based on lookup of test id to application project	No	Mandatory
association to facility data item	Facility the data was acquired on	GUID of relevant facility data item based on lookup of test id to facility	No	Mandatory
association to companysite data item / security group	Company site the data was acquired on. Also determines which RBAC rules apply to this object	GUID of relevant company site data item/security group based on lookup based on facility	No	Mandatory
association to vehicle data item	Vehicle the data was acquired on	GUID of relevant vehicle data item based on lookup of test id to vehicle	No	Mandatory
association to vehiclebuild data item	Vehicle build the data was acquired on	GUID of relevant vehicle build data item based on lookup of test id to vehicle build	No	Optional
association to parttest data item	The part test the the data was acquired during	GUID of relevant part test data item based on lookup of test id to part test	Yes	Optional
[association to experimentitem data items]	The experiment items the scan is acquired as part of satisfying	GUIDs of relevant experiment item data items based on lookup of test id to experiment items from TEDS download	No	Optional
[association to operatingitem data items]	The operating items the scan is acquired as part of satisfying	GUIDs of relevant operating item data items based on lookup of test id to operating items from TEDS download	No	Optional
association to testplan data item	The test plan this scan was taken as a result of	GUID of relevant test plan data item based on lookup of test activity in the diary to test plan	No	Optional
association to testexecutioninstructionitem tag	The test execution item this scan was taken as a result of	GUID of relevant test execution instruction item tag based on lookup of test activity in the diary to test execution instruction item	No	Optional
association to testexecutioninstructionprocedure tag	The test execution procedure this scan was taken as a result of	GUID of relevant test execution instruction procedure tag based on lookup of test activity in the diary to test execution instruction procedure	No	Optional

association to testexecutioninstructionactivity tag	The test execution activity this scan was taken as as a result of	GUID of relevant test execution instruction activity tag based on lookup of test activity in the diary to test execution instruction activity	No	Optional
association to testactivity tag	The instance of the test execution activity this item was taken as part of.	GUID of relevant test activity tag based on finding scan in test diary and finding relevant test activity.	No	Optional
[association to qt65graph data items]	The graphs used to create derived values. Named GSN1-N	GUIDs of relevant QT65 graphs based on lookup from info in scan	No	Optional
contentDetails - size	Size of the HDF5 File	From size of HDF5 file	No	Mandatory
contentDetails - mimeType	A MIME type is a string identifier composed of two parts: a type and a subtype. The "type" refers to a logical grouping of many MIME types that are closely related to each other; it's no more than a high level category. "subtypes" are specific to one file type within the "type".	"application/x-hdf5"	No	Mandatory
contentDetails - originalFileName	Filename (less path) of the HDF5 file on TLS	From original scan file name	No	Mandatory
contentDetails - md5CheckSum	MD5 Checksum of the HDF5 File	Calculated	No	Mandatory

Blade File Example

The blade file is an extension to a mainframe database that is used to store information about engine blades, their environment (annulus) and results from various applications. There are 55 types of data held on the mainframe that are used by fans, compressors and turbines.

Type#	Name
1	AERO ANNULUS DEFINITION
2	Q263 AERO INPUT
3	DESIGN AERODYNAMICS
4	MECH ANNULUS/SHROUD DATA
5	CONICAL SECTION (S180)
6	STREAM SECTION - DT6
7	PLANE SECTION - DT7
8	ANALYSIS AERODYNAMICS
9	STREAM-TUBE DATA (Q312)
10	INTERNAL GEOMETRY - DT10
11	INTERNAL FLOW - DT11
12	FINITE ELEMENT MESH
13	PLATFORM STREAMLINE DATA
14	COMPRESSOR PROFILE DEFN
15	GENERAL AERODYNAMICS
16	AUTO ITER INPUT - DT16
17	STREAMTUBE DEFINITION

18	PVD OR TITAN DATA
19	FILM COOLING DATA - DT19
20	G605 INPUT DATA
21	MECHANICAL PROPERTIES
22	UNTWIST DEFINITION
23	STAGE STACKING DEFN
24	3D FLOW OP MASTER RECORD
25	SPAN INPUT DATA - DT25
26	G633 INPUT DATA
27	DISC DESIGN DATA
28	PRELIMINARY DESIGN DATA
29	TURBINE TEST SERIES
30	TURBINE TEST BUILD
31	TURBINE TEST DATA
32	TURBINE OPTIMISE (G414)
33	PASSAGE WINDOWS
34	VERITAS MASTER INDEX
35	BOUNDARY LAYER DATA
36	COMP. STAGE CHIC INPUT
37	CHIC STORAGE
38	QUASI-3D AUTO ITERATION
39	BLADE TO BLADE INPUT
40	STREAM SECTION - DT40
41	STREAM SECT AERO - DT41
42	STREAM SECT FLOWFIELD
43	STREAM SECT B LAYER
44	AEROFOIL NURBS SURFACE
45	BOUNDARY LAYER
46	LOSS MODEL
47	S1BYL OUTPUT DATA
48	JE01 CONTROL DATA
49	JB16 CONTROL DATA - DT49
50	BLADE GEOMETRY DATA
51	SURFACE AERODYNAMIC DATA
52	GRID DEFINITION
53	BOUNDARY LAYER DATA
54	SD04 STRESSING BALANCING
55	EXECUTABLE CONTROL DATA

Blade file data records can have some/all the following meta-data. The rows marked Y imply that the attribute is used to uniquely define the data.

Name	**Description**	**Example**	**Identifier**	**Scope**
Data Type	The data type	1-55	Y	All DTs
EWA	Engine Work Authority	11000	Y	All DTs
Type	Compressor or Turbine	Comp	Y	All DTs
Shaft	LP / IP / HP	LP	Y	All DTs
Version	Version of item 1-999	1-999	Y	All DTs
Stage	The stage of the compressor	1-10	Y	Some DTs

Blade	The type of blade Rotor/Stator	R/S	Y	Some DTs
Section	The blade section typically 1-21	1	Y	Some DTs
Title	40 character string that often includes a userid		N	All DTs

Data Quantities

TBD