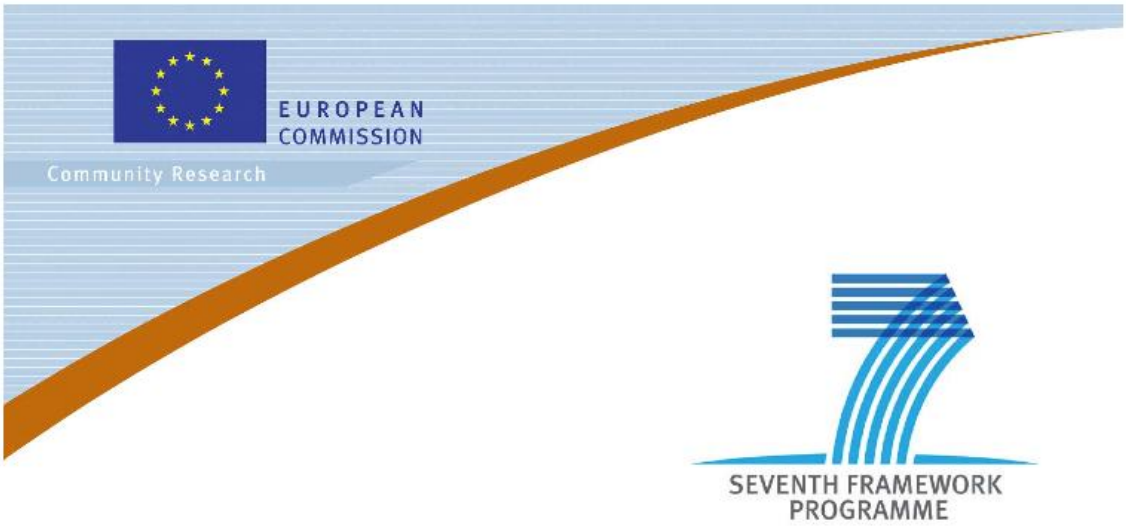

Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

D.13.7.3: Technical Roadmap (Data/Context chapter)

Project acronym: FI-Core
Project full title: Future Internet Core
Contract No.: 632893
Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform
Project Document Number: ICT-2013-FI-632893-13-D.13.7.3
Project Document Date: 2016-09-29
Deliverable Type and Security: PU
Author: Santiago Martínez (TID)
Contributors: Fermín Galán (TID), Francisco Romero (TID), Uri Shani (IBM), Jo Barratt(OKFN).

1 Introduction

1.1 Executive summary

This document describes the Roadmap of the functionalities that FIWARE Data Chapter has delivered deliver as part of its major release 5, giving a detailed account of the schedule for its minor releases and keeping previous releases for accountability reasons.

A clear table shows the releases & sprints, mapping them to calendar dates in the whole history of the platform until the end of Release 5 (September 2016). This technical chapter provides its internal roadmap in relation with this table.

Releases are composed of features associated with overall epics. An epic captures a large body of work. It is essentially a large user story that can be broken down into a number of smaller stories. A feature may have a few related stories. This document supplies the Release 5 roadmap including a description of the features and epics associated with the release.

1.2 About This Document

This document provides an update of the Roadmap of the functionalities that the FIWARE Data chapter has delivered in Release 5 (September 2016).

A clear table shows the releases & sprints mapping them to calendar dates in the whole history of the platform until the end of Release 5. This technical chapter provides its internal roadmap in relation with this table.

Releases (both major and minor) are composed of features associated with global epics. Epics are organization items that can be used to capture a large body of work, typically covered through multiple minor releases. Epics are in turn broken down into features, each of them describing work to be done during a particular minor release. These features are then subsequently broken down into smaller user stories and work items that form each of the sprint's backlog belonging to a particular minor release. A minor release is composed of 3 monthly sprints, each of them with a dedicated backlog. For the purposes of this document, the roadmap for the Data chapter will be shown up to a feature detail level.

The present document is a periodic issue that contains a snapshot of the state of the roadmap as of September 2016. The agile methodology implies a constant evolution of the roadmap and FIWARE strives to keep it up to date, accurately showing the results that will deliver in coming releases.

1.3 Intended Audience

The document targets those interested in the intended direction of FIWARE's Data Chapter.

1.4 Structure of this Document

The document is generated out of resources publicly provided in the FIWARE wiki. For the current version of the documents please visit the public wiki at http://wiki.fi-ware.org/ particularly the page http://wiki.fi-ware.org/FI-WARE_Technical_Roadmap

The following pages were used to generate this document in a MediaWiki collaborative environment:

General:

- http://wiki.fiware.org/FIWARE_Technical_Roadmap
- http://wiki.fiware.org/Releases_and_Sprints_numbering,_with_mapping_to_calendar_dates
- http://wiki.fiware.org/Roadmap_of_Data/Context_Management

Publish/Subscribe Context Broker (Orion):

- FIWARE.Feature.Data.OrionContextBroker.AttributeManagement
- FIWARE.Epic.Data.OrionContextBroker.NGSIv2API
- FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.UpdateContext
- FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.SubscribeContext
- FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.ListingContextBrokerInfo
- FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.QueryContext
- FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.SingleEntityOperations
- FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.EntitiesDeletion
- FIWARE.Feature.Data.OrionContextBroker.Websockets
- FIWARE.Feature.Data.OrionContextBroker.DataManagementOptimizations
- FIWARE.Feature.Data.OrionContextBroker.ErrorManagementOptimizations
- FIWARE.Feature.Data.OrionContextBroker.CORS
- FIWARE.Feature.Data.OrionContextBroker.AlarmManager
- FIWARE.Feature.Data.OrionContextBroker.Retries
- FIWARE.Feature.Data.OrionContextBroker.LocationBasedSubscriptions
- FIWARE.Feature.Data.OrionContextBroker.ServiceSubserviceManagement
- FIWARE.Feature.Data.OrionContextBroker.OptionalParameters
- FIWARE.Feature.Data.OrionContextBroker.NGSI9OneTimeSubscription
- FIWARE.Feature.Data.OrionContextBroker.ImproveNGSIOpsFormat
- FIWARE.Feature.Data.OrionContextBroker.RegisterWithRegExp

Complex Event Processing (CEP):

- FIWARE.Epic.Data.CEP.EventPatternDetection
- FIWARE.Feature.Data.CEP.EventPatternDetection.HistoricalEvents
- FIWARE.Epic.Data.CEP.PatternUsability
- FIWARE.Feature.Data.CEP.PatternUsability.UseParticipantEvents

- FIWARE.Feature.Data.CEP.PatternUsability.TrendRatio
- FIWARE.Epic.Data.CEP.IntegrationWithDataGEs
- FIWARE.Feature.Data.CEP.IntegrationWithDataGEs.SupportOrionJsonFormat
- FIWARE.Feature.Data.CEP.IntegrationWithDataGEs.SupportOrionHeadersInOutputEvents
- FIWARE.Epic.Data.CEP.NewPattern2
- FIWARE.Feature.Data.CEP.NewPattern2.RelativeN
- FIWARE.Feature.Data.CEP.NewPattern2.RelativeNUI

Big Data Analysis (Cosmos):

- FIWARE.Epic.Data.BigData-Analysis.HadoopAsAService
- FIWARE.Feature.Data.BigData-Analysis.HadoopAsAService.Sahara
- FIWARE.Epic.Data.BigData-Analysis.NGSIIntegration
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionHDFSSinkBinary
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionSinkBatchProcessing
- FIWARE.Feature.Data.BigData-Analysis.CustomHiveAuthProvider
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.BatchSupport
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.GeneralDataModelParameter
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.MultipleHDFSFormats
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionHDFSSink
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.StringAggregations
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionKafkaSink
- FIWARE.Epic.Data.BigData-Analysis.GUI
- FIWARE.Feature.Data.BigData-Analysis.GUI.Calendar
- FIWARE.Feature.Data.BigData-Analysis.GUI.FileBrowser
- FIWARE.Feature.Data.BigData-Analysis.GUI.AdminPage
- FIWARE.Feature.Data.BigData-Analysis.GUI.ExecutionPage
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionCartoDBSink
- FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionDynamoDBSink
- FIWARE.Epic.Data.BigData-Analysis.SinfonierAsStormStudio
- FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.DeploySinfonierSharedInstance
- FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.NGSIStormSpoutsCatalogue
- FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.NGSIStormBoltsCatalogue

Stream-oriented (Kurento):

- FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.AdvancedSdpNegotiation
- FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.DataChannelFilter
- FIWARE.Feature.Data.Stream-oriented.RtcProtocols.SDES
- FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.FilterMetadata
- FIWARE.Feature.Data.Stream-oriented.RtcProtocols.RtpPortRange
- FIWARE.Feature.Data.Stream-oriented.Cloud.KmsProvider

- FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.MultiDomainMedia
- FIWARE.Feature.Data.Stream-oriented.RtcProtocols.IceUpdate
- FIWARE.Epic.Data.Stream-oriented.AdvancedMedia
- FIWARE.Epic.Data.Stream-oriented.RtcProtocols

Short Term Historic Open Data Repository (CKAN):

- FIWARE.Epic.Data.CKAN.GenericEnabler.Catalogue
- FIWARE.Feature.Data.CKAN.Release24
- FIWARE.Feature.Data.CKAN.DCAT.Harvest
- FIWARE.Feature.Data.CKAN.MultilingualSupport.DCAT
- FIWARE.Feature.Data.CKAN.DOI.ManualRegistration
- FIWARE.Feature.Data.CKAN.DOI.AutomaticRegistration
- FIWARE.Feature.Data.CKAN.MultilingualSupport.TranslationInterface
- FIWARE.Feature.Data.CKAN.MultilingualSupport.MultilingualMetadata
- FIWARE.Epic.Data.CKAN.MultilingualSupport
- FIWARE.Epic.Data.CKAN.DOI
- FIWARE.Epic.Data.CKAN.GenericEnabler
- FIWARE.Feature.Data.CKAN.EasyInstallAndConfiguration.ConfigurableSettings
- FIWARE.Feature.Data.CKAN.Release.25
- FIWARE.Feature.Data.CKAN.DCAT.DCATAP11
- FIWARE.Feature.Data.CKAN.Harvesting.Sevilla
- FIWARE.Feature.Data.CKAN.EasyExtensionInstallAndConfiguration.ConfigurableSettings
- FIWARE.Epic.Data.CKAN.Harvesting
- FIWARE.Epic.Data.CKAN.EasyExtensionInstallAndConfiguration
- FIWARE.Epic.Data.CKAN.EasyInstallAndConfiguration
- FIWARE.Feature.Data.CKAN.PylonsMigration.Audit
- FIWARE.Feature.Data.CKAN.PylonsMigration.FlaskMiddleware
- FIWARE.Feature.Data.CKAN.PylonsMigration.FlaskMigrate
- FIWARE.Feature.Data.CKAN.DCAT.OASCSupport
- FIWARE.Feature.Data.CKAN.PylonsMigration.ReferenceControllers
- FIWARE.Epic.Data.CKAN.Release
- FIWARE.Epic.Data.CKAN.PylonsMigration

1.5 Keyword list

FIWARE, FI-Core, Acceleration Programme, Accelerators, PPP, Architecture Board, Steering Board, Roadmap, Reference Architecture, Generic Enabler, Open Specifications, I2ND, Cloud, IoT, Data/Media and Context Management, Applications/Services and Data Delivery, Delivery Framework, Security, Advanced Middleware, Interfaces to Networks and Robotics, Communities, Tools, Sustainability Support Tools, ICT, es.Internet, Apiary, Github, Latin American Platform.

1.6 Change History

Release	Major changes description	Date	Editor
v1	Version ready for submission	2016-09-29	Santiago Martinez, TID

1.7 Table of contents

2 FIWARE Technical Roadmap

FIWARE's technical roadmap brings information about what the different major releases of FIWARE will bring and when they will be available on FIWARE Lab. For each of the FIWARE chapters and for every generic enabler, the list of features available for the particular release is linked within the following pages. You can explore which release number the particular features are currently scheduled for and contact the affiliated responsible persons in charge if you need more details.

The first version of the FIWARE platform was made available on the FIWARE Testbed during 3Q2012 for experiments by the use case projects within the FI-PPP program. The overall goal for this first Release was to provide a sufficiently complete set of FIWARE GEs which Use Case Projects selected in the first phase of the FI-PPP program could test and use in their proof of concepts. The second release of FIWARE is focused on integration (both inside and across chapters) as well as evolution of FIWARE GEs based on feedback from target Users (both Use Case Projects selected in the first phase of the FI-PPP or other stakeholders like current/target customers of FIWARE partners). The third release is focused on consolidation of the platform and packaging.

The fourth and fifth releases were delivered by a new set of partners (many of them already present in releases 1,2 and 3) and will bring a reorganization of the map of GEs. These GEs will continue building the core platform ensuring that the results are open and royalty free.

Once the development of a given minor release of FIWARE is finished, it can be made available on FIWARE Lab, typically by the end of the following month. Updates of all FIWARE GEs on FIWARE Lab will be planned after each major release completion. Nevertheless, updates of FIWARE Lab may be decided on a more frequent basis at FIWARE GE level, i.e., the following month after completion of some Sprint. Please check the Releases and Sprints numbering, with mapping to calendar dates for further information.

FIWARE's Technical Roadmap is broken into 7 partial roadmaps, one per FIWARE Technical Chapter:

- Roadmap of Cloud Hosting
- Roadmap of Data/Context Management
- Roadmap of Internet of Things (IoT) Services
- Roadmap of Applications/Services Ecosystem and Delivery Framework
- Roadmap of Security
- Roadmap of Advanced middleware, Interface to Networks and Robotics
- Roadmap of Advanced Web-based UI

The Developers Community and Tools chapter has changed their focus and they do not produce GEs as such anymore. We keep their past roadmap as it was at the end of 2014 for future reference:

- Roadmap of Developers Community and Tools

Important Note: it is important to mention that most (if not all) of the Generic Enabler implementations are based on existing baseline assets, actively developed and promoted by the corresponding companies. Each company has done internal analysis of requirements and priorities for each of the technologies, based on their business needs and the needs of their customers. The goal of FIWARE is to develop these assets even further, to integrate them together to form a coherent platform, and to offer them to the FI-PPP ecosystem, for the benefit of the community. This roadmap reflects this approach.

3 Releases and Sprints numbering, with mapping to calendar dates

The list of Releases and Sprints together with the time frame of each one of them is depicted in the following tables.

Versions	Nov 2011	Dec 2011	Jan 2012	Feb 2012	Mar 2012	Apr 2012	May 2012	Jun 2012	Jul 2012	Aug 2012	Sep 2012	Oct 2012	Nov 2012	Dec 2012	Jan 2013	Feb 2013	Mar 2013	Apr 2013	May 2013	Jun 2013	Jul 2013	Aug 2013	Sep 2013	Oct 2013	Nov 2013	Dec 2013	Jan 2014	Feb 2014	Mar 2014	Apr 2014	May 2014	Jun 2014	Jul 2014	Ago 2014
FIWARE-I Month Number	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22	M23	M24	M25	M26	M27	M28	M29	M30	M31	M32	M33	M34	M35	M36	M37	M38	M39	M40
FIWARE-II Month Number																																		
First Major Release	1.1.1	1.1.2	1.1.3	1.2.1	1.2.2	1.2.3	1.3.1	1.3.2	1.3.3	1.4.1	1.4.2	1.4.3																						
Second Major Release										2.0.1	2.0.2	2.1.1	2.1.2	2.1.3	2.2.1	2.2.2	2.2.3	2.3.1	2.3.2	2.3.3														
Third Major Release																					3.1.1	3.1.2	3.1.3	3.2.1	3.2.2	3.2.3	3.3.1	3.3.2	3.3.3	3.4.1	3.4.2	3.4.3	3.5.1	3.5.2

Versions	Sep 2014	Oct 2014	Nov 2014	Dec 2014	Jan 2015	Feb 2015	Mar 2015	Apr 2015	May 2015	Jun 2015	Jul 2015	Aug 2015	Sep 2015	Oct 2015	Nov 2015	Dec 2015	Jan 2016	Feb 2016	Mar 2016	Apr 2016	May 2016	Jun 2016	Jul 2016	Aug 2016	Sep 2016
FIWARE-I Month Number	M41	M42	M43	M44																					
FIWARE-II Month Number	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22	M23	M24	M25
First Major Release																									
Second Major Release																									
Third Major Release	3.5.3																								
Fourth Major Release		4.1.1	4.1.2	4.1.3	4.2.1	4.2.2	4.2.3	4.3.1	4.3.2	4.3.3	4.4.1	4.4.2	4.4.3												
Fifth Major Release														5.1.1	5.1.2	5.1.3	5.2.1	5.2.2	5.2.3	5.3.1	5.3.2	5.3.3	5.4.1	5.4.2	5.4.3

PLEASE NOTE that software associated to Minor Releases may be made available on the FI-WARE Testbed and FIWARE Lab after completing that Minor Release, typically by the end of the following month. A revised version of the documentation accompanying software delivered after closing a Minor Release is also typically delivered by end of the following month.

As explained in FIWARE Agile Development Methodology, the Releases and Sprints are referred to as

- FIWARE.Release.x.y
- FIWARE.Sprint.x.y.z

4 Roadmap of Data/Context Chapter

4.1 Introduction

The Data/Context chapter in FI-WARE provides key assets enabling high performance Data, Context and Events acquisition, processing, analysis and exploitation.

You can learn more about the Data/Context Chapter by reading the FIWARE Architecture and Open Specifications. Following is a description of the Technical Roadmap planned for the chapter, which will be developed through subsequent releases of the FI-WARE Platform. Please also check the Releases and Sprints numbering, with mapping to calendar dates.

4.2 Fifth Major Release

The planned date for the fifth major release of the project is end of September 2016. The main features and epics from the different GEs in the Data chapter, that are planned to be available in this release can be found below, as part of this section. This list of features and epics is based on the latest roadmap analysis that has been performed for each of the GEs in the chapter. This roadmap analysis effort has also been reflected in our development activities tracking system (Jira), and might be subject to minor changes if new requirements are identified until the planned date for this release.

The list of the main planned features and epics for each of the GEs that are part of the Data/Context chapter can be found below:

Publish/Subscribe Broker (Orion)

- Extend the set of FIWARE NGSIv2 APIs and their flexibility, in particular:
  - Accept empty attribute values in UpdateContext operation: this will allow to update the attribute metadata, without having to update the attribute value itself.
  - Enable the subscription to multiple entities at a time, based on filtering and/or the FIWARE service/subservice they belong to.
  - Allow subscribing to an entity leaving the conditions empty, therefore being notified on any change to the entity's parameters, including those that were not present at the time of subscription.
  - Enable the creation of subscriptions without expiration date.

  - List available registrations, entities, entity attributes, subscriptions, subscription filters, filtered entities and supported operations in the Context Broker.
  - Enable all single entity related NGSIv2 operations, based on their id and type as unique identifiers.
  - Add the FIWARE service/subservice concept in the entity query parameters and subscription notifications.
  - Enable listing the available FIWARE service/subservice tree.
  - Enable changing the FIWARE service/subservice in entities and registrations.
  - Enable massive entities deletion.

- Support websockets in the Context Broker.

- Allow retries in the subscription notifications.

- Support CORS in the Context Broker APIs.

- Support location based subscriptions.

- Improve robustness (error management) and efficiency.

- Add optional parameters for: avoiding initial notification at subscription creation time, returning entity/attribute creation and update timestamp, validating empty subscriptions, deactivation for not listening notification endpoints.

Complex Event Processing (Proton)

The goal of this release is to increase the usability of the CEP. This includes:

- Enhance existing pattern usability to ease the implementation of advanced scenarios. Those additions are based on user requests. In particular:
  - Enhance the TREND pattern, to detect a trend pattern only if a given ratio over the detected trend was preserved.
  - Allow the user to refer to the attributes of the pattern's participant events, even in patterns that usually do not keep the participant events in memory (such as trend and aggregation).

- Support running the CEP on historical events, by injecting those historical events to the engine and detecting patterns as if the events occurred in present time, while reducing the processing time in comparison to the actual historical event time duration. This can also be useful for simulation purposes.

- Implement a revised adapter for the message broker ORION to cope with its deprecated XML API and its move to JSON format. The old adapter will be kept for backward compatibility. This addition is not critical since the back versions of the message broker support the XML format.

- Improve stability and performance with better threads and memory management to prevent memory leakage caused by fault in that management.
- Enhance packaging options such as Puppet and Docker.
- Re-enable text format of messages in RESTful calls (that have been disabled in the past).

Big Data Analysis (Cosmos)

- Hadoop clusters as a service (continuing the work from 4th release)
  - Deploy a global instance of Sahara in FIWARE Lab
- Spark clusters as a service (continuing the work from 4th release)
  - Deploy a shared cluster in FIWARE Lab
  - Deploy the Spark plugin in FIWARE Lab's Sahara
- NGSI-like repositories as source of input data for Hadoop (continuing the work from 4th release)
  - Extend the Hadoop API with a new NGSI input format
- Swift as source of input data for Hadoop (continuing the work from 4th release)
  - Extend the Hadoop API with a new Swift input format
- Sinfonier integration (continuing the work from 4th release)
  - Deploy a global instance of Sinfonier in FIWARE Lab as a Storm studio
  - Create a catalogue of NGSI-like Storm bolts
- NGSI integration (continuing the work from 4th release)
  - Binary version of HDFS sink
  - Batch processing on the sink side
  - New sink for DynamoDB
- Cosmos GUI
  - Calendar-like visualization tool showing the available analysis slots in terms of infrastructure and time
  - Administration page
  - File system browser
  - MapReduce uploading and execution page

Stream Oriented (Kurento)

- Extend the enabler to support latest trends on media technologies
  - Media renegotiation
  - Track grouping for synchronization and other purposes
  - DataChannels consumption on filters
  - Filter communication of synchronized meta data
  - Multi-domain filters acting as sensors publishing events through data channels.
- Improve the interoperability and the transport protocols of the enabler
  - Support for legacy SDES key negotiation
  - Support for configurable port ranges
  - Support for latest trends on Interactive Connectivity Establishment
- Improve the efficiency of the enabler
  - Profile the enabler under different load conditions
  - Find performance bottlenecks
  - Optimize code on the appropriate bottlenecks.

Short Term Historic Open Data Repository (CKAN)

- Service Integrations
  - Improve harvesting and federation of data sources to better support various data source that could be harvested.
  - Improve the CKAN API interface
  - Support more data types for indexing and previewing
  - Refactor current revisions handling in CKAN
  - Enhance CKAN visualizations
- Platform updates
  - Migrate away from the Pylons web framework to something more modern
  - Make it easy to install, add and configure CKAN extensions from within CKAN
- Releases
  - 2.5
  - 2.6

4.2.1 Summary list of supported Features and Epics per GE in the chapter

For a more detailed description of the Epics and Features to be provided for each of the GEs in the Data/Context chapter, you can refer to the following table below which helps to identify which Backlog Epics and Features have already been specified and planned for the fifth Major Release of FI-WARE:

FI-WARE GE	Supported Features and Epics
Publish Subscribe Broker- Orion	Release 5.1: FIWARE.Feature.Data.OrionContextBroker.AttributeManagement FIWARE.Epic.Data.OrionContextBroker.NGSIv2API FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.UpdateContext FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.SubscribeContext FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.ListingContextBrokerInfo FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.QueryContext FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.SingleEntityOperations FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.EntitiesDeletion Release 5.2: FIWARE.Feature.Data.OrionContextBroker.Websockets FIWARE.Feature.Data.OrionContextBroker.DataManagementOptimizations FIWARE.Feature.Data.OrionContextBroker.ErrorManagementOptimizations FIWARE.Feature.Data.OrionContextBroker.CORS FIWARE.Feature.Data.OrionContextBroker.AlarmManager Release 5.3: FIWARE.Feature.Data.OrionContextBroker.Retries FIWARE.Feature.Data.OrionContextBroker.LocationBasedSubscriptions FIWARE.Feature.Data.OrionContextBroker.ServiceSubserviceManagement FIWARE.Feature.Data.OrionContextBroker.OptionalParameters Release 5.4: FIWARE.Feature.Data.OrionContextBroker.NGSI9OneTimeSubscription

	FIWARE.Feature.Data.OrionContextBroker.ImproveNGSIOpsFormat FIWARE.Feature.Data.OrionContextBroker.RegisterWithRegExp
Complex Event Processing - Proton	Release 5.3: FIWARE.Epic.Data.CEP.EventPatternDetection FIWARE.Feature.Data.CEP.EventPatternDetection.HistoricalEvents FIWARE.Epic.Data.CEP.PatternUsability FIWARE.Feature.Data.CEP.PatternUsability.UseParticipantEvents FIWARE.Feature.Data.CEP.PatternUsability.TrendRatio Release 5.4: FIWARE.Epic.Data.CEP.IntegrationWithDataGEs FIWARE.Feature.Data.CEP.IntegrationWithDataGEs.SupportOrionJsonFormat FIWARE.Feature.Data.CEP.IntegrationWithDataGEs.SupportOrionHeadersInOutputEvents FIWARE.Epic.Data.CEP.NewPattern2 FIWARE.Feature.Data.CEP.NewPattern2.RelativeN FIWARE.Feature.Data.CEP.NewPattern2.RelativeNUI
Big Data Analysis-Cosmos	Release 5.1: FIWARE.Epic.Data.BigData-Analysis.HadoopAsAService FIWARE.Feature.Data.BigData-Analysis.HadoopAsAService.Sahara FIWARE.Epic.Data.BigData-Analysis.NGSIIntegration FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionHDFSSinkBinary FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionSinkBatchProcessing Release 5.2: FIWARE.Feature.Data.BigData-Analysis.CustomHiveAuthProvider FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.BatchSupport FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.GeneralDataModelParameter FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.MultipleHDFSFormats FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionHDFSSink FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.StringAggregations FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionKafkaSink Release 5.3:

	FIWARE.Epic.Data.BigData-Analysis.GUI FIWARE.Feature.Data.BigData-Analysis.GUI.Calendar FIWARE.Feature.Data.BigData-Analysis.GUI.FileBrowser FIWARE.Feature.Data.BigData-Analysis.GUI.AdminPage FIWARE.Feature.Data.BigData-Analysis.GUI.ExecutionPage FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionCartoDBSink FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionDynamoDBSink Release 5.4: FIWARE.Epic.Data.BigData-Analysis.SinfonierAsStormStudio FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.DeploySinfonierSharedInstance FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.NGSIStormSpoutsCatalogue FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.NGSIStormBoltsCatalogue
Stream Oriented-Kurento	Release 5.1: FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.AdvancedSdpNegotiation FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.DataChannelFilter Release 5.2: FIWARE.Feature.Data.Stream-oriented.RtcProtocols.SDES FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.FilterMetadata Release 5.3: FIWARE.Feature.Data.Stream-oriented.RtcProtocols.RtpPortRange FIWARE.Feature.Data.Stream-oriented.Cloud.KmsProvider Release 5.4: FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.MultiDomainMedia FIWARE.Feature.Data.Stream-oriented.RtcProtocols.IceUpdate FIWARE.Epic.Data.Stream-oriented.AdvancedMedia FIWARE.Epic.Data.Stream-oriented.RtcProtocols

CKAN	Release 5.1: FIWARE.Epic.Data.CKAN.GenericEnabler.Catalogue FIWARE.Feature.Data.CKAN.Release24 FIWARE.Feature.Data.CKAN.DCAT.Harvest FIWARE.Feature.Data.CKAN.MultilingualSupport.DCAT FIWARE.Feature.Data.CKAN.DOI.ManualRegistration FIWARE.Feature.Data.CKAN.DOI.AutomaticRegistration FIWARE.Feature.Data.CKAN.MultilingualSupport.TranslationInterface FIWARE.Feature.Data.CKAN.MultilingualSupport.MultilingualMetadata FIWARE.Epic.Data.CKAN.MultilingualSupport FIWARE.Epic.Data.CKAN.DOI FIWARE.Epic.Data.CKAN.GenericEnabler Release 5.2: FIWARE.Feature.Data.CKAN.EasyInstallAndConfiguration.ConfigurableSettings FIWARE.Feature.Data.CKAN.Release.25 FIWARE.Feature.Data.CKAN.DCAT.DCATAP11 FIWARE.Feature.Data.CKAN.Harvesting.Sevilla FIWARE.Feature.Data.CKAN.EasyExtensionInstallAndConfiguration.ConfigurableSettings FIWARE.Epic.Data.CKAN.Harvesting FIWARE.Epic.Data.CKAN.EasyExtensionInstallAndConfiguration FIWARE.Epic.Data.CKAN.EasyInstallAndConfiguration Release 5.3: Release 5.4: FIWARE.Feature.Data.CKAN.PylonsMigration.Audit FIWARE.Feature.Data.CKAN.PylonsMigration.FlaskMiddleware FIWARE.Feature.Data.CKAN.PylonsMigration.FlaskMigrate FIWARE.Feature.Data.CKAN.DCAT.OASCSupport FIWARE.Feature.Data.CKAN.PylonsMigration.ReferenceControllers FIWARE.Epic.Data.CKAN.Release FIWARE.Epic.Data.CKAN.PylonsMigration

5 Backlog items in the Roadmap of Data/Context chapter

This section has the detailed information of each of the backlog items as publicly available in the wiki.

5.1 Backlog items of Publish/Subscribe Broker -Orion

5.1.1 Epics

FIWARE.Epic.Data.OrionContextBroker.NGSIv2API

Name	NGSIv2API	Chapter	Data
Goal	To extend the set of available APIs for NGSIv2 and improve their flexibility for users.		
Description	NGSIv2 APIs are intended to manage the whole lifecycle of the context information including updates, queries, registrations and subscriptions. They simplify and extend the possibilities of the previous NGSI version operations. The development for NGSIv2 APIs started on July 2015 (Orion 0.23.0) and both NGSI versions will coexist for some time, aiming at deprecating NGSIv1 at some point in time.		
Rationale	Extend the existing capabilities provided by the NGSI APIs while simplifying them and increasing the set of available NGSI operations		

5.1.2 Features

FIWARE.Feature.Data.OrionContextBroker.AttributeManagement

Name	Attribute Management	Chapter	Data
Goal	Enhance flexibility related to attributes used as part of NGSI operations.		
Description	This feature includes several attribute related stories. In particular: 1) Enhance the Query Context operation flexibility: add new attribute filters (equal, less, more, partial match, diff etc. 2) Implement numeric semantics in attributes, by means of using the "type" attribute field. This field may be set to "float" or "integer" (for float or integer values, respectively) and,		

	in that case, CB will implement numeric semantics for them. In particular, creating/updating attributes with invalid format will be rejected and the numeric value may be used in aritmetic filters such as "greater than". 3) Support the inclusion of custom metadata in the attributes within a registerContext NGSI9 operation, to be retrieved with discoverContextAvalability operation.
Rationale	This functionality allows to support several cases where enhanced flexibility for managing attributes related to the context data is required. This will help developers work for interacting with the Context Broker.

FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.UpdateContext

Name	UpdateContext (NGSIv2)	Chapter	Data
Goal	Implement all the changes related to NGSIv2 Update Context operation		
Description	UpdateContext operation will be enhanced by simplifying it and increasing its flexibility in NGSIv2. This will include changes like accepting empty attribute values, to allow cases where the intention is just to update the attribute metadata, and not the attribute value itself.		
Rationale	Simplify the usage of the NGSI operations by the users and enrich them to cover the identified user's requirements		

FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.SubscribeContext

Name	SubscribeContext (NGSIv2)	Chapter	Data
Goal	Implement all the changes related to NGSIv2 Subscribe Context operation		
Description	SubscribeContext operation will be enhanced by simplifying it and increasing its flexibility in NGSIv2. This will include changes like enabling the subscription to multiple entities at a time, allowing the subscription to an entity while leaving the conditions value empty, which will result in being notified about all changes to attributes, or enabling subscriptions without a specific expiration date (permanent)		

Rationale	Simplify the usage of the NGSI operations by the users and enrich them to cover the identified user's requirements

FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.ListingContextBrokerInfo

Name	ListContextBrokerInfo (NGSIv2)	Chapter	Data
Goal	Implement all operations related to listing information available in the Context Broker.		
Description	NGSIv2 will include operations related to the listing of information available in the context Broker. These include the possibility to list all available registrations, entities, entity attributes, subscriptions, subscriptions filters, filters entities and supported operations in the Context Broker.		
Rationale	Enrich and enhance the usability and flexibility of the NGSI operations, by covering new functionality not provided in previous NGSI version		

FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.QueryContext

Name	QueryContext (NGSIv2)	Chapter	Data
Goal	Implement all the changes related to the Query Context NGSIv2 operation.		
Description	QueryContext operation will be simplified and its flexibility will be increased as part of NGSIv2. This will include for example adding the service and subservice concept as part of this operation.		
Rationale	Usability for the Query Context operations will be increased by giving new possibilities of interacting with the Context Broker through this operation.		

FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.SingleEntityOperations

Name	Single entity related operations (NGSIv2)	Chapter	Data

Goal	Implement all the changes related to the management of single entities using the NGSIv2 operations
Description	NGSIv2 will allow to easily manage single entities by using their id and type to create a unique identifier which will be used in the NGSIv2 operations to refer to them.
Rationale	The management of single entities through the Context Broker will be made simple and powerful through the set of NGSIv2 operations implement for this purpose.

FIWARE.Feature.Data.OrionContextBroker.NGSIv2API.EntitiesDeletion

Name	Massive entities deletion (NGSIv2)	Chapter	Data
Goal	Implement a new NGSIv2 operation for allowing the massive deletion of entities in the Context Broker.		
Description	A new NGSIv2 operation will be available for deleting all available entities in the Context Broker. This will avoid the need to interact with the internal MongoDB database for doing so.		
Rationale	There was no way of deleting entities previously created in the Context Broker other than accessing the MongoDB database directly. This new NGSI operation will facilitate this.		

FIWARE.Feature.Data.OrionContextBroker.Websockets

Name	Support websockets	Chapter	Data
Goal	Support websockets as part of the NGSI notification mechanism in the Context Broker.		
Description	Orion should support websocket in NGSI notifications. This would allow to receive notifications behind a NAT. In addition, it will relieve client applications to implement a full-fledged REST server to receive notifications.		

Rationale	This functionality allows to support those cases where notifications are sent to clients working behind a NAT setup and will help the developer's work for interacting with Orion Context Broker.

FIWARE.Feature.Data.OrionContextBroker.DataManagementOptimizations

Name	Data Management Optimizations	Chapter	Data
Goal	Optimize the way Orion Context Broker manages the context data internally, to increase its efficiency.		
Description	A number of improvements will be implemented in order to increase the Context Broker efficiency, including the regular expression evaluation mechanism for the subscriptions, the connections pool in the APIs and the notifications and improving the logs system.		
Rationale	Efficiency in the Context Broker can be increased by implementing a number of optimizations that have been previously identified.		

FIWARE.Feature.Data.OrionContextBroker.ErrorManagementOptimizations

Name	Error Management Optimizations	Chapter	Data
Goal	Optimize the way Orion Context Broker manages the errors, to improve its usability		
Description	A number of improvements will be implemented in order to enhance the management of errors, including the lock of local notifications to avoid infinite loops, allow the retries in the subscription notifications, manage errors for duplicated subscriptions etc.		
Rationale	Enhance the management of different types of possible errors and improve the log management while interacting with the Context Broker.		

FIWARE.Feature.Data.OrionContextBroker.CORS

Name	CORS	Chapter	Data

Goal	Implement CORS schema for GET, POST and DELETE type of NGSI operations
Description	Release 0.22 of Orion Context Broker (0.22) includes CORS support for GET requests. This means that one can query context data from a Web App without having to develop a server-side component to act as a proxy for Orion. Support for performing POST and DELETE operations following the same schema will be added as part of this feature.
Rationale	Interacting with Orion Context Broker from a web app is simplified by implementing CORS schema for all types of operations, eliminating the need to develop a server-side component to act as a proxy for Orion.

FIWARE.Feature.Data.OrionContextBroker.AlarmManager

Name	Alarm Manager	Chapter	Data
Goal	Implement an alarm manager considering different types of alarms as part of the Context Broker implementation.		
Description	Alarm conditions and types to be implemented as part of the alarm manager can be checked at https://github.com/telefonicaid/fiware-orion/blob/develop/doc/manuals/admin/logs.md#alarm-conditions. Alarms to take into account are rows 2, 4 and 5 there.		
Rationale	Currently CB works in a stateless way regarding alarms and release conditions, which has some problems for the system consuming the logs and setting alarms. Thus, we implement a better system based on transient-state in CB.		

FIWARE.Feature.Data.OrionContextBroker.Retries

Name	Retries	Chapter	Data
Goal	Implement a retry mechanism for the failed Subscribe Context notifications		
Description	Currently if the notification from the Context Broker does not reach the Context Consumer for whatever reason it is lost (unless using some additional component		

	like Rush). The purpose of this feature is to implement a retry mechanism as part of the Context Broker itself.
Rationale	For ensuring that the notifications sent by the Context Broker reach successfully the Context Consumer, an additional retry mechanism is needed in the context broker, that can cover the failure cases.

FIWARE.Feature.Data.OrionContextBroker.LocationBasedSubscriptions

Name	Location Based Subscriptions	Chapter	Data
Goal	Allow the inclusion of location based information for discriminating the subscriptions.		
Description	The purpose of this feature is to allow context consumers to limit their subscription to certain entities filtered by their location information.		
Rationale	Currently there is no mechanism for including location based information as part of the Subscribe Context requests. This feature will allow to do so.		

FIWARE.Feature.Data.OrionContextBroker.ServiceSubserviceManagement

Name	ServiceSubserviceManagement	Chapter	Data
Goal	Enable the change of service and subservice information for the registrations and entities		
Description	The purpose of this feature is to allow updating the service and subservice related information for entities and registrations.		
Rationale	There are cases where it may be interesting to update the service and subservice related information to which a particular entity or registration belongs to. This feature will allow to do so.		

FIWARE.Feature.Data.OrionContextBroker.OptionalParameters

Name	Optional Parameters	Chapter	Data
Goal	To add several optional parameters that can be used for indicating a special behaviour to follow in the Context Broker		
Description	The purpose of this feature is to implement several optional parameters that can be included as part of the NGSI operations, such as returning entity/attribute creation and update timestamp, avoiding the initial notification at the time of creating the subscription, or validating empty subscriptions.		
Rationale	There are cases where users may wish the Context Broker to behave in a specific way, other than the usual one. These optional parameters will be included in order to cater for this.		

FIWARE.Feature.Data.OrionContextBroker.NGSI9OneTimeSubscription

Name	NGSI9OneTimeSubscription	Chapter	Data
Goal	Support the NGSI9 one time subscription operation by the Orion Context Broker		
Description	The purpose of this feature is to allow creating a one time subscription in the Context Broker, so once the notified event takes place, the subscription is no longer valid.		
Rationale	There can be scenarios where a one time subscription for receiving the corresponding notifications once the update occurs may be interesting. This feature will allow creating this type of subscriptions.		

FIWARE.Feature.Data.OrionContextBroker.ImproveNGSIOpsFormat

Name	Improve NGSI ops format	Chapter	Data
Goal	To implement several improvements in the format used for several NGSI operations to increase efficiency.		

Description	The purpose of this feature is to implement several general purpose improvements such as returning arrays even for single entity operations, including notification IDs, creating attribute objects for all JSON vectors etc.
Rationale	Increase the efficiency and provide a uniform way of managing NGSI operations by the Context Broker.

FIWARE.Feature.Data.OrionContextBroker.RegisterWithRegExp

Name	Register With RegExp	Chapter	Data
Goal	To allow registering by using regular expression patterns in the Context Broker		
Description	The purpose of this feature is to allow using regular expression patterns for the registration with the Context Broker for increased flexibility		
Rationale	Increase flexibility at the time of registration by allowing the usage of regular expression patterns.		

5.2 Backlog items of Complex Event Processing (Proton)

5.2.1 Epics

FIWARE.Epic.Data.CEP.EventPatternDetection

Name	EventPatternDetection	Chapter	Data
Goal	Detect CEP patterns based on incoming events according to defined semantics		
Description	Implement a CEP engine that detects patterns based on incoming events. The patterns are given in the CEP language (in JSON format). The list of supported patterns divided into the following groups according to their types:		

	Filter operator - the situation is detected if the incoming event passes a threshold condition. The filter is a stateless operator, that does not correlate between its participant events. Join operators All – the situation is detected if all its listed participant events arrive in any order. Sequence – the situation is detected if all its listed participant events arrive in exactly the order of the operands. Absence operator - none of the listed events have arrived during the context. Aggregation operator – an Aggregate EPA is a transformation EPA that takes as input a collection of events and computes values by applying functions over the input events. Those computed values can used for the EPA condition and for its derived events.
Rationale	Allow to define CEP event patterns according to defined semantics (Filter, All, Sequence, Absence, Aggregation), to support the basic use of the CEP technology.

FIWARE.Epic.Data.CEP.PatternUsability

Name	PatternUsability	Chapter	Data
Goal	Enhance existing patterns to increase their usability and make them applicable to more advanced scenarios.		
Description	Evaluate requests from users, and passed use cases, design, and implement additions to existing patterns that improve the expressability of the CEP. Includes support in the CEP Authoring tool, the CEP Engine and required Documentation (mainly user guide)		
Rationale	The usability of the CEP depends on the ease to implement CEP applications. New useful additions to existing patterns can contribute to the usability of the CEP		

FIWARE.Epic.Data.CEP.IntegrationWithDataGEs

Name	IntegrationWithDataGEs	Chapter	Data
Goal	Enhance the integration of the CEP with Data chapter GEs		
Description	Enhance the integration of the CEP with other chapter GEs, such as the Context Broker. Currently known issues: support X-Auth when sending context update events to the context broker, support getting several context updates in the same message from the Context Broker.		
Rationale	Improve over all FIWARE usability by enhancing the integration of the CEP GE with the Context Broker GE and optionally with other Data chapter GEs		

FIWARE.Epic.Data.CEP.NewPattern2

Name	AddNewPatterns-continue	Chapter	Data
Goal	Extend TREND pattern with reference to a relative strength expressed with a length of that trend - a pattern that identifies a trend over some expression above incoming events during a temporal processing context, is now also required to reflect some strength according to that indication. A UI for that indication is required too.		
Description	Evaluate requests from users, and if valuable design that extension, and implement it to improve the expressibility of the CEP, with its appropriate UI Includes support in the CEP Authoring tool, the CEP Engine and required Documentation (mainly user guide).		
Rationale	The usability of the CEP depends on the ease to implement CEP applications. New useful patterns can contribute to the usability of the CEP This is an extension for epic FIWARE.Epic.Data.CEP.NewPattern: DATA-1487.		

5.2.2 Features

FIWARE.Feature.Data.CEP.EventPatternDetection.HistoricalEvents

Name	Historical Events	Chapter	Data
Goal	Allow to run the CEP on historical events while preserving the correctness of the CEP patterns detection		
Description	The CEP gets as input a file of historical event data, and a CEP application definition. The CEP processes those historical input events, taking into account their historical occurrence time, and detects the CEP patterns as if the events occur in present time. This is done while shortened the CEP processing time in comparison to the actual time interval in which those event occurred historically.		
Rationale	We had several scenarios in which the user had records of historical events and they wanted to identified CEP patterns over those events. This can be done for simulation purposes as well.		

FIWARE.Feature.Data.CEP.PatternUsability.UseParticipantEvents

Name	Use Participant Events	Chapter	Data
Goal	Enhance the usability of some CEP patterns by allowing them to refer to the participant events of the pattern.		
Description	Till this feature, some CEP patterns such as TREND and AGGREGATION didn't save the participant events, and the user could not refer to their participant events in the derivation of an output event. By adding this feature, the user can refer to the participant events even in those patterns. The CEP engine analyses the pattern definition and decides which participant events need to be saved to allow the user to refer to their attributes.		
Rationale	Several use cases required this feature, since just the detection of the pattern was not good enough and the users wanted to output some attributes of the events that caused the pattern detection. Doing it without this addition required work-arounds that complicated the CEP application definition. The CEP usability depends on the ease to implement CEP applications. Enhanced pattern can contribute to the usability of the CEP.		

FIWARE.Feature.Data.CEP.PatternUsability.TrendRatio

Name	TrendRatio	Chapter	Data
Goal	Enhance the Trend pattern to support more advance scenarios and fine grain control on a trend pattern detection.		
Description	A TREND pattern identifies where there is a trend over some expression above incoming events during a temporal processing context. This addition allows the user to define expected ratio between the events that participate in the trend pattern. Only if the events hold the given ratio the pattern will be detected.		
Rationale	The TREND pattern was identified as very useful pattern in several use cases in several domains, and more advanced control over the trend pattern was identified as useful. The CEP usability depends on the ease to implement CEP applications. Enhanced pattern can contribute to the usability of the CEP.		

FIWARE.Feature.Data.CEP.IntegrationWithDataGEs.SupportOrionJSONformat

Name	Add JSON support for Orion	Chapter	Data
Goal	Keep CEP operational with the new revision of ORION which dropped the support for XML format messages and can now support only JSON messages.		
Description	While the old versions of ORION support XML and JSON format messages, the XML format has been deprecated and dropped from that GE. To keep up with the supported version of ORION, CEP should work with JSON messages. This means that an additional adapter to ORION for CEP needs to be developed. CEP work natively with JSON events, but its present adapter for ORION exchanges XML format messages. The additional adapter will be used to work with the new versions of ORION, exchanging JSON messages. The XML option will work with older versions, which will allow CEP to be backward compatible with ORION. As this happens, the XML adapter will not longer be supported.		
Rationale	CEP usability with ORION needs to be kept up to date. Yet, it is not a critical addition since the older version of ORION can still be used. Hence, this feature is categorized as minor.		

FIWARE.Feature.Data.CEP.IntegrationWithDataGEs.SupportOrionHeadersInOutputEvents

Name	Support Orion Headers In Output Events	Chapter	Data
Goal	Allow the user to add additional headers to the output events sent to external REST services		
Description	Allow to add additional headers to the CEP output events in a producer definition. Original feature (FIWARE.Feature.Data.CEP.AdditionalHeadersInOutputEvents) needs to be generalized, in particular to resolve issues on the integration with Orion where FIWARE-Service and FIWARE-ServicePath items are required in the message header		
Rationale	When the CEP sends output events to the Context Broker there needs to be a way to provide header entries for FIWARE-Service and FIWARE-ServicePath, so that the CEP engine will include it when sending output events to the CB REST service		

FIWARE.Feature.Data.CEP.NewPattern2.RelativeN

Name	Add New Relative N Pattern	Chapter	Data
Goal	Add a new Relative N pattern - a pattern that identifies the N events with the biggest or lowest values in a given temporal window		
Description	A Relative N pattern, (also called Top K) identifies the N events with the biggest or lowest values of some attribute in a given temporal window. The user defined the number N, and the attribute of the events that need to be monitored. The pattern is detected when at least N events arrived, and identified the N events that caused its detection. The user can refer to those N events in the output event derivation. As part of this feature implementation: design the Relative N pattern's metadata and implementation, implement the pattern in the engine, and add required documentation		
Rationale	The Relative N pattern was identified as a useful pattern in several use cases in several domains. The CEP usability depends on the ease to implement CEP applications. New useful pattern can contribute to the usability of the CEP.		

FIWARE.Feature.Data.CEP.NewPattern2.RelativeNUI

Name	Add New Relative N Pattern UI	Chapter	Data
Goal	Add a new Relative N pattern - a pattern that identifies the N events with the biggest or lowest values in a given temporal window		
Description	A Relative N pattern, (also called Top K) identifies the N events with the biggest or lowest values of some attribute in a given temporal window. The user defined the number N, and the attribute of the events that need to be monitored. The pattern is detected when at least N events arrived, and identified the N events that caused its detection. The user can refer to those N events in the output event derivation. As part of this feature implementation: design the relative N pattern's UI add it to the CEP authoring tool (import, export, UI, metadata).		
Rationale	The Relative N pattern was identified as a useful pattern in several use cases in several domains. The CEP usability depends on the ease to implement CEP applications. New useful pattern can contribute to the usability of the CEP.		

5.3 Backlog items of Big Data Analysis (Cosmos)

5.3.1 Epics

FIWARE.Epic.Data.BigData-Analysis.SinfonierAsStormStudio

Name	SinfonierAsStormStudio	Chapter	Data
Goal	To integrate Sinfonier as a Storm studio for the Big Data enabler.		
Description	This will comprise the deployment of Sinfonier in FIWARE Lab for real-time processing. This will require a shared Storm cluster and an initial library of NGSI-like Storm spouts and bolts (then, it may be extended by the community).		
Rationale	Telefónica's Sinfonier asset, an already developed Storm studio, fits perfectly with the real-time processing side of the Big Data enabler.		

FIWARE.Epic.Data.BigData-Analysis.HadoopAsAService

Name	HadoopAsAService	Chapter	Data
Goal	Offering Hadoop clusters (storage and computing) as a service		
Description	A platform allowing for using Hadoop as a service. Primitives such as requesting a private computing cluster, destroying it, uploading analysis jobs, etc. will be needed. A shared but secure storage cluster is needed for input and output data storage. Sahara from Openstack seems to be a good alternative to our previous implementation.		
Rationale	Private computing clusters are expensive, thus a platform allowing for reusing the underlying infrastructure is mandatory in a FIWARE node.		

FIWARE.Epic.Data.BigData-Analysis.NGSIIntegration

Name	NGSI Integration	Chapter	Data
Goal	To allow the integration of Orion Context Broker and Cosmos/any other HDFS-based system such as Hadoop.		
Description	The integration of Orion Context Broker and Cosmos or any other HDFS-based system (e.g. Hadoop) is about persisting context data managed by Orion into HDFS, in a historical basis. This means Orion will handle the last value regarding a certain entity's attribute, but HDFS will handle the complete history regarding the values that attribute has had all along the time. This kind of integration will be done by adding new capabilities to the Cygnus too.		
Rationale	Orion handles the last value regarding a certain entity's attribute, but HDFS may handle the complete history regarding the values that attribute has had all along the time.		

FIWARE.Epic.Data.BigData-Analysis.GUI

Name	Cosmos GUI	Chapter	Data

Goal	To develop a Graphical User Interface (GUI) for Cosmos management in a first stage, both storage and computing; and Map and Reduce jobs execution and HDFS data management in a second stage.
Description	Storage and computing management through this GUI will be related to provision of Cosmos accounts, which means to have a HDFS user space in the shared storage cluster, and temporal access to the computing cluster for Map and Reduce jobs execution. Later, the GUI will allow for running those jobs in a remote way, either by using predefined jobs or custom ones; HDFS user space could be managed as well, allowing for a full fylesystem management (creating and deleting files and folders, moving data, uploading and downloading data, etc).
Rationale	The infrastructure supporting the storage and computing resources is shared among many users, thus a tool for managing and orchestrating them is necessary.

5.3.2 Features

FIWARE.Feature.Data.BigData-Analysis.HadoopAsAService.Sahara

Name	Sahara	Chapter	Data
Goal	To deploy a Hadoop as a service platform based on Sahara.		
Description	Sahara from Openstack will be used for the basis of a Hadoop as a service platform. On the one hand, this platform will maintain the previously created HDFS-based shared storage cluster in order the data is not lost. On the other hand, Sahara will use the infrastructure from the shared computing Hadoop cluster in order to support the dynamic creation of private Hadoop clusters.		
Rationale	The real aim of the Big Data enabler is to provide a Hadoop as a service platform, not a shared Hadoop cluster.		

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionHDFSSinkBinary

Name	OrionHDFSSinkBinary	Chapter	Data

Goal	To develop an alternative backend implementation for Cygnus's custom HDFS sink, based on the binary API of Hadoop.
Description	HDFS final storage can be accessed from a Cygnus custom HDFS sink through two different means: RESTful based API, as done until now, or Hadoop API based, which can be named as "binary" and it is supposed to be faster than the so-called "rest" API. This way of accessing the HDFS file system will be used by an alternative implementation of OrionHDFSink, in order to improve the performance.
Rationale	RESTful based API is supposed to be slow, or at least slower than a binary based API. Performance is critical in Cygnus, thus its sinks must use the best and faster implementations.

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionKafkaSink

Name	OrionKafkaSink	Chapter	Data
Goal	To create a sink for Apache Kafka, the distributed streaming platform, where to persist Orion context data.		
Description	This sink will persist the notified context data in queues which name will depend on the data model being configured and the notified FIWARE service, FIWARE service path, entity ID, entity type and entity's attributes. Within the queues, the data will be persisted a Json object having two main fields: the Flume headers internally handled at Cygnus and the notification itself in Json format as well		
Rationale	Cygnus is able to persist Orion context data in several final storages. Apache Kafk must be one of them.		

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionCartoDBSink

Name	OrionCartoDBSink	Chapter	Data
Goal	To create a sink for CartoDB, the geolocated PostGIS database, where to persist geolocated Orion context data.		

Description	This sink will persist the notified context data in CartoDB tables which names will depend on the configured data model and the notified FIWARE service, FIWARE service path, entity ID, entity type and entity's attributes. Within the tables, for each notified attribute there will be a column except for the geolocation-related attribute, which will be a special column in the table.
Rationale	Cygnus is able to persist Orion context data in several final storages. CartoDB must be one of them.

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionSinkBatchProcessing

Name	OrionSinkBatchProcessing	Chapter	Data
Goal	To modify the OrionSink base class behaviour, which is currently based on processing events one by one, by implementing a batching mechanism based on processing batches of events, which are accumulated for a while.		
Description	Currently, events are processed as they are notified, one by one. This is approach may result slow with certain backend implementations, specially those based in RESTful APIs. In order to improve the performance it is necessary to reduce the amount of writes/inserts in the destination element (file, table, resource, etc). This can only be done if several notified data related to the same final destination element is accumulated for a while and persisted as a single data unit.		
Rationale	One by one event processing is supposed to be slow, or at least slower than a batch write. Performance is critical in Cygnus, thus its sinks must use the best and faster implementations.		

FIWARE.Feature.Data.BigData-Analysis.CustomHiveAuthProvider

Name	CustomHiveAuthProvider	Chapter	Data
Goal	To create a custom Hive authenticator provider based on OAuth2 mechanism.		
Description	Hive allows for creating custom authentication providers that can be plugged and configured in order to bypass the default authentication mechanisms. This custom		

	authentication provider will be based on OAuth2. Regarding the Java interfaces that must be implemented, the user will have to provide a user and a OAuth2 token.
Rationale	OAuth2 mechanism is the standard way of authenticating enabler interfaces. Hive, as natively designed, does not provide any kind of authentication mechanism based on OAuth2.

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.BatchSupport

Name	BatchSupport	Chapter	Data
Goal	To allow Cygnus, as NGSI connector with HDFS, can work with batches of data instead of processing it notification by notification.		
Description	Instead of persisting data notification by notification, it is better to accumulate a batch of data regarding a destination path (a HDFS file, a MySQL/Postgre/DynamoDB table, a CKAN resource, a MongoDB collection, etc) and perform a unique rite/insert/upsert. The performance of Cygnus will be largely increased.		
Rationale	Cygnus performance must be constantly improved since real time data generated by Orion Context Broker may be large in terms of notifications and data per notification.		

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.GeneralDataModelParameter

Name	GeneralDataModelParameter	Chapter	Data
Goal	Generalize the usage of the data_model parameter, allowing for working with several ones.		
Description	To add the data_model parameter as a general parameter in OrionSink (the base for all the sinks). This will allow all the sinks work with the several available data models. i.e. dm-by-service, dm-by-service-path, dm-by-entity and dm-by-attribute. Sinks already using this parameter must be adapted to the new available data model values.		

Rationale	All the Cygnus sinks may work with the same data models some sinks are already working (e.g. OrionMySQLSink or OrionMongoSink).

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.MultipleHDFSFormats

Name	MultipleHDFSFormats	Chapter	Data
Goal	To allow Cygnus, as NGSI connector with HDFS, can handle several output formats such as Json and CSV.		
Description	Modify the current HDFS sink at Cygnus in order, upon configuration, the data can be persisted in Json or CSV format. In addition, apart from the format, the data model (row or column) must be maintained. Thus, the sinks must be able to persist data in all the four combinations (i.e. json-row, json-column, csv-row and csv-column). The code must be open enough to handle new formats in the future.		
Rationale	HDFS is a (distributed) plain file system where the data can be written in several output formats. Among these formats can be found Json and CSV, but many others can be used in the future, such as Parquet.		

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionHDFSSink

Name	OrionHDFSSink	Chapter	Data
Goal	To create a sink for Hadoop Distributed File System (HDFS) where to persist Orion context data.		
Description	This sink for HDFS is different than the existing one in the native Apache Flume distribution (Cygnus is based on Flume). This sink will be able to persist the data in NGSI-like format, i.e. persisting the data in a specific path depending on the notified service, service path and entity data.		
Rationale	Cygnus is able to persist Orion context data in several final storages. Hadoop Distributed File System (HDFS) must be of them.		

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.StringAggregations

Name	String Aggregations	Chapter	Data
Goal	To allow the current existing OrionSTHSink may aggregate string-based data, for instance, based on the number of occurrences of a substring.		
Description	Current existing OrionSTHSink must be ready to aggregate non numerical context data, i.e. string-based data, in terms of the certain string-based aggregation functions such as the number of occurrences of a substring in the original notified string.		
Rationale	The sink for STH must be able to aggregate data not only for numerical notifications but string-based notifications as well.		

FIWARE.Feature.Data.BigData-Analysis.GUI.Calendar

Name	GUI Calendar	Chapter	Data
Goal	To implement a Calendar-like feature in the Cosmos GUI for cluster usage scheduling.		
Description	FIWARE Cosmos global instance is limited in storage and computation resources. In order to correctly schedule its usage, a Calendar-like feature is required, in order the users may check available free slots for computation, and request them for their job runs.		
Rationale	The cluster must be used in an uniform way, i.e. all the hours of the day it must be active, instead of having peaks of load, or directly hours of the day where it is not used at all.		

FIWARE.Feature.Data.BigData-Analysis.GUI.FileBrowser

Name	GUI File Browser	Chapter	Data
Goal	To implement a graphical tool for file browsing a Hadoop Distributed File System.		
Description	This tool must be able to implement all the available Hadoop File System Shell commands, e.g. creating files and directories, renaming, moving or deleting them,		

	changing permissions and ownership, etc. Apache Hue's File Browser may be used as an inspiration, or directly integrated some way.
Rationale	This kind of tool shows two advantages: the first one is the user will have a friendly UI when dealing with his/her HDFS content management; the second one is the cluster is not accessed through ssh anymore (best for administration purposes).

FIWARE.Feature.Data.BigData-Analysis.GUI.AdminPage

Name	GUI Admin Page	Chapter	Data
Goal	To implement a graphical tool for administration purposes (user management, cluster management, etc).		
Description	This tool must allow administrators to manage the users (adding, removing, editing them), the cluster (administrative scripts to be run, cluster details, processes, etc) and any other aspect of a Cosmos deployment. The Cosmos GUI must be able to recognize administration accounts in order to properly show the admin interface.		
Rationale	This kind of tool will allow administrators to manage a Cosmos deployment through a user friendly interface, joining in a single place all the administration operations. This will homogenize all the administration tools in a single one as well.		

FIWARE.Feature.Data.BigData-Analysis.GUI.ExecutionPage

Name	GUI Execution Page	Chapter	Data
Goal	To implement a graphical tool for Map and Reduce jobs execution.		
Description	This tool must allow a user to run a Map and Reduce job by simply providing a set of parameters: jar path, main class within the jar, input and output paths and Java options. the tool must provide means for tracking the execution, showing the map and reduce progress, any error that may happen and the final result, of course.		
Rationale	This kind of tool will allow users to run Map and Reduce jobs without accessing the cluster through ssh, which is an advantage from the user point of view (he/she has		

	not to deal with the Hadoop command details, classpaths, etc), and from the administration point of view as well.

FIWARE.Feature.Data.BigData-Analysis.NGSIIntegration.OrionDynamoDBSink

Name	DynamoDB Sink	Chapter	Data
Goal	To implement a Cygnus custom sink for Amazon Web Services' cloud-based DynamoDB.		
Description	Amazon Web Service's DynamoDB is a NoSQL database hosted on the cloud. The way the data is organized is about creating tables within the AWS user space and adding items into those tables. Items has no fixed number of fields (this is why it is NoSQL). A custom sink for Cygnus will be able to create the above-mentioned tables and insert data items regarding the notified contest data by Orion.		
Rationale	Certain clients may need to persist Orion notified context data in a cloud-based NoSQL database such as AWS DynamoDB.		

FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.DeploySinfonierSharedInstance

Name	DeploySinfonierSharedInstance	Chapter	Data
Goal	To deploy a Sinfonier instance.		
Description	A Sinfonier instance is based on a shared Storm cluster and the Sinfonier designing tool itself.		
Rationale	Real time processing is required by the Big Data enabler. Sinfonier may help from the analysis designing point of view.		

FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.NGSIStormSpoutsCatalogue

Name	NGSIStormSpoutsCatalogue	Chapter	Data

Goal	Create a catalogue of NGSI-related Storm spouts.
Description	Spouts are the source of input data for any Storm topology/algorithm. These spouts will mainly receive real-time streaming data both directly from the sources or by means of Orion notifications.
Rationale	Current version of Sinfonier has no NGSI-related spouts. A minimal set of spouts is required in order users can start working with the tool.

FIWARE.Feature.Data.BigData-Analysis.SinfonierAsStormStudio.NGSIStormBoltsCatalogue

Name	NGSIStormBoltsCatalogue	Chapter	Data
Goal	Create a catalogue of NGSI-related Storm bolts.		
Description	Bolts are the analysis nodes for any Storm topology/algorithm. These bolts will implement primitives such as aggregation, filtering, addition, minimum/maximum calculation, etc.		
Rationale	Current version of Sinfonier has no NGSI-related bolts. A minimal set of bolts is required in order users can start working with the tool.		

5.4 Backlog items of Stream-Oriented (Kurento)

5.4.1 Epics

FIWARE.Epic.Data.Stream-oriented.AdvancedMedia

Name	Advanced Media	Chapter	Data
Goal	To align the Stream-oriented GE with latest trends on media technologies		
Description	Media transport and processing mechanisms are progressing at a very high speed in the last few months. This makes necessary to introduce in the Stream-oriented GE		

	the ability to tackle such progresses through the introduction of advanced mechanisms both at the signaling and at the media plane. These mechanisms may include aspects such as the following: Support for streams composed of multiple video and audio tracks Media renegotiation for modifying, in hot, the characteristics of tracks and the composition of streams Media interoperability techniques for adapting among heterogeneous families of protocols (e.g. H.264 to/from VP8) Advanced mechanisms for media browsing Advanced mechanism for media track grouping and synchronization Mechanisms for combining sensor data with audiovisual data Mechanism for leveraging codec scalability in the provisioning of differentiated quality of service.
Rationale	The ability to maintain an attractive set of features in the Stream-oriented GE depends on complying with latest trends in the market.

FIWARE.Epic.Data.Stream-oriented.RtcProtocols

Name	RtcProtocols	Chapter	Data
Goal	To provide high quality RTC (Real-Time Communications) transport mechanisms for the transmission of multimedia streams for conversational applications.		
Description	Basically transport is done on RTP and its secure version (SRTP) and are complemented by the control protocol RTCP. As part of the different problems usually linked to protocols capabilities, here a few examples: Consider low latency capabilities for interactive applications based on WebRTC or equivalent technologies. NAT traversal, passing the firewall, and maintain connectivity in streaming modes.		

	Quality optimization mechanisms basing on different RTCP messages such as NACK, PLI and REMB.
Rationale	Different services (e.g. full-duplex, simplex), QoS (e.g. low-latency, high-quality, etc.) and network architectures (e.g. NAT traversal) require different transport protocols for the streams. Those protocols need to be provided by the Stream-oriented GE.

5.4.2 Features

FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.AdvancedSdpNegotiation

Name	AdvancedSdpNegotiation	Chapter	Data
Goal	To refactor SDP negotiation mechanism in order to provide support to advanced media features including SDP renegotiation, simulcast, stream grouping, etc.		
Description	SDP negotiations are the mechanism used in most RTC media services for negotiating the establishment of media sessions. Our current SDP negotiation mechanism is based on direct string manipulations and cannot be extended or adapted easily for supporting new capabilities. For this reason, we need to create a more powerful SDP negotiation library where new extensions might be added in a more straightforward way.		
Rationale	A number of advanced WebRTC media features are emerging and they cannot be integrated into the Stream-oriented GE without refactoring the SDP negotiation mechanism.		

FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.DataChannelFilter

Name	DataChannelFilter	Chapter	Data
Goal	To provide support for consuming DataChannel information on filters, so that sensor data can be used for enriching the media.		
Description	DataChannels provide client applications the capability of sending arbitrary data together with the audio-visual streams. This data can be useful for enriching media streams in filters using some kind of augmented reality (e.g. drawing a thermometer		

	for showing a temperature sensor, etc.) In this feature, we shall evolve Kurento module systems for making possible filters to consume such data.
Rationale	The integration of sensor data into media services can be done visually using augmented reality filters

FIWARE.Feature.Data.Stream-oriented.RtcProtocols.SDES

Name	SDES	Chapter	Data
Goal	To provide SDES support to Kurento		
Description	WebRTC security mechanisms are based on DTLS key exchange for DTLS. However, may legacy systems still use the SDES mechanism, as described in RFC 4568. This feature accounts for the creation of the appropriate extensions for supporting such mechanism and guaranteeing interoperation with legacy systems.		
Rationale	SDES is the only mechanism supported for SRTP key exchange on many legacy RTC systems.		

FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.FilterMetadata

Name	FilterMetadata	Chapter	Data
Goal	To provide a mechanism enabling inter-filter meta data exchange synchronized with video.		
Description	Many augmented reality mechanisms are based on two steps: detection and rendering. The detection phase uses computer vision techniques for detecting a specific pattern in images (e.g. marker, or planar). The rendering phase depicts some kind of content on it (e.g. image, 3D model, etc.) Currently, Kurento augmented reality filters need to implement both steps. However, splitting the process into two filters would be more flexible and maintainable, given that the same detector filter would be use later with different renderers. This feature shall create the appropriate mechanisms enabling this by making possible the exchange of synchronized metadata among the filters.		

Rationale	To split detection and rendering into independent filters for enhancing modularity, maintainability and re-usability of the code.

FIWARE.Feature.Data.Stream-oriented.RtcProtocols.RtpPortRange

Name	RtpPortRange	Chapter	Data
Goal	To make possible to specify a port-range to be used by Rtp derived endpoints.		
Description	Currently, all Rtp derived endpoints (including the WebRtcEndpoint) use port on an open range for communications. However, this may cause problem on systems with strong security rules. For this reason, this feature shall be in charge of providing the appropriate mechanisms enabling port-rages to be configured and used on all RtpEndpoint children.		
Rationale	To make possible to integrate Kurento in systems with strong security policies in relation to open ports.		

FIWARE.Feature.Data.Stream-oriented.Cloud.KmsProvider

Name	Kms Provider	Chapter	Data
Goal	To enable KurentoClient to be used in a cloud environment.		
Description	KurentoClient is linked to a specific media server, whose IP is provided in constructions. This is not compatible with self-scalable cloud environments where the number of media server instances can grow or shrink dynamically. This feature shall be in charge of adapting KurentoClient implementation, and eventually the Stream-oriented Open API for enabling the media server to be used in cloud environments seamlessly.		
Rationale	Self-scalable systems are not supported in our current Kurento architecture.		

FIWARE.Feature.Data.Stream-oriented.AdvancedMedia.MultidomainMedia

Name	Multi Domain Media	Chapter	Data
Goal	To enable the seamless integration of arbitrary sensor data with audio-visual data.		
Description	DataChannels and meta-data are two different ways of combining sensor data with audio-visual data in the Stream-oriented GEri. There are planned mechanisms for using both in combination with filters and endpoints. However, there is not a clear mechanism for combining both, which can be interesting in cases where multiple domains of media need to be combined (e.g. using a Stream-oriented GE computer vision filter as a sensor, which publishing its data through a DataChannel). This feature shall create such mechanism making possible the transformation of DataChannel data into synchronized meta-data and vice versa.		
Rationale	To enable multi-domain media filters suitable for acting as sensors publishing events through DataChannels.		

FIWARE.Feature.Data.Stream-oriented.RtcProtocols.IceUpdate

Name	IceUpdate	Chapter	Data
Goal	To update ICE implementations for supporting novel ICE features.		
Description	ICE (Interactive Connectivity Establishment) is a protocol designed for enabling media connectivity with independence on the network topology. Current ICE implementation of the Stream-oriented GEri is becoming obsolete and we need to enhance it with new capabilities such as TCP ICE, HTTP CONNECT, IPv6, etc. This feature is in charge of doing it.		
Rationale	To update the ICE implementation to support latest standards and capabilities.		

5.5 Backlog items of CKAN

5.5.1 Epics

FIWARE.Epic.Data.CKAN.Genericenabler.Catalogue

Name	Catalogue	Chapter	Data
Goal	Register CKAN as a Generic Enabler (for open data) in the FIWARE catalogue		
Description	In order to make CKAN available a generic enabler in the FIWARE catalogue: http://catalogue.fiware.org/ it needs to be registered and approved as such by the FIWARE consortium		
Rationale	Without being registered in the FIWARE catalogue as a generic enabler, no user of FIWARE will be able to find and install CKAN and thus not able to make use of the platform to open their data.		

FIWARE.Epic.Data.CKAN.MultilingualSupport

Name	MultilingualSupport	Chapter	Data
Goal	Improve multilingual support (internationalisation and thus localisation) on more levels		
Description	CKAN already has multilingual support and has been translated into numerous languages. There is however always room for improvements and CKAN could support localization (l10n) on more levels, e.g. by adding internationalization (i18n) to fields in the database or for display of harvested DCAT datasets for example (i.e. metadata in DCAT can be translated).		
Rationale	Since fi-ware stretches across many regions and languages it is very important to have good multilingual support on more levels than just the user facing web interface.		

FIWARE.Epic.Data.CKAN.DOI

Name	DOI	Chapter	Data
Goal	Improve Digital Object Identifier (DOI) handling in CKAN to make digital data citations easier to achieve		
Description	Digital Object Identifier (DOI) is an ISO standard identifier scheme for digital objects (ISO 26324:2012). It is used by for example the European Union to publish official documents (via the EU publication office) and by researchers who want to cite digital objects (e.g. datasets). It is possible to use DOI in CKAN but the handling is not optimal. For example it cannot really be used to its full extent in harvesting of data so this is related to FIWARE.Epic.Data.CKAN.Harvesting.		
Rationale	Global and standardized identifiers for datasets in CKAN avoid duplication of datasets and make processing, citation and publishing easier to achieve.		

FIWARE.Epic.Data.CKAN.Harvesting

Name	Harvesting	Chapter	Data
Goal	Improve harvesting in CKAN (i.e. fetching data from other sources and importing it into a CKAN instance)		
Description	Improve harvesting and federation of data sources to better support various data source that could be harvested. This includes better harvesting of geographic data, different taxonomies and ontologies etc. Through better harvesting integration with other data platforms in fi-ware (and outside of the fi-ware platform) is better supported.		
Rationale	Better integration and interoperability with other FIWARE projects, e.g. harvesting from the metadata store, other CKAN instances etc. means data is able to flow automatically between different systems without any manual labour.		

FIWARE.Epic.Data.CKAN.GenericEnabler

Name	GenericEnabler	Chapter	Data
Goal	Make CKAN the Generic Enabler implementation of an open data platform in FIWARE		
Description	This is closely related to the FIWARE.Epic.Data.CKAN.EasyInstallAndConfiguration which will also include a script to install CKAN. However this involves hooking that up to the fi-ware specific tools, i.e. chef or puppet, providing documentation for fi-ware users and getting CKAN listed in the fi-ware generic enabler catalogue.		
Rationale	To be useful for fi-ware CKAN needs to be a generic enabler, findable in the registry and installable via a script.		

FIWARE.Epic.Data.CKAN.EasyExtensionInstallAndConfiguration

Name	EasyExtensionInstallAndConfiguration	Chapter	Data
Goal	Make it easy to install, add and configure CKAN extensions from within CKAN		
Description	Extensions are now installed via a shell command (complexity relies on extension) and by modifying the .ini configuration file. Instead it should be possible to add extensions and configure them from within CKAN using a web interface. For some it might not be possible, but for others it should be simple. This should strive for a one click install and configuration interface.		
Rationale	Similarly to the rationale for FIWARE.Epic.Data.CKAN.EasyInstallAndConfiguration this is important for maintenance of CKAN in a cloud infrastructure (like FIWARE) where the command line might not be accessible (or problematic for automatic deployments) and the .ini file might also cause problems.		

FIWARE.Epic.Data.CKAN.EasyInstallAndConfiguration

Name	EasyInstallAndConfiguration	Chapter	Data
Goal	Make installation of CKAN easy and configuration possible from within CKAN		
Description	Installing CKAN currently requires performing numerous manual steps in order to install CKAN. These steps can be automated to a large extent which has been shown to work with the Docker installation. That however does rely on Docker and even though Docker has simplified the installation, the configuration still takes place via a .ini file. Configuration needs to be simpler and more accessible, e.g. via a web interface (although out of security reasons, some things might require to be manual configurations). This epic is closely related to the FIWARE.Epic.Data.CKAN.GenericEnabler epic which also requires CKAN to be easy to install.		
Rationale	Simplifying the installation process and making configurations more user friendly on the frontend is essential for automated deployments and management (e.g. one might not have access to the .ini file in a cloud infrastructure like fi-ware).		

FIWARE.Epic.Data.CKAN.Release

Name	Release	Chapter	Data
Goal	Help, support and ensure features and work items are released/updated in an official release		
Description	CKAN is an open source project and as such it is necessary to work closely with the developer community to get new features and work items into a release or ensure that they stay in the release. It is also important to make sure the extensions which fi-ware uses are updated and migrated when a new version is released.		
Rationale	Maintain fi-ware as an official part of CKAN (or keep extensions up to date so that fi-ware can use the most recent version of CKAN instead of being stuck on an older version)		

FIWARE.Epic.Data.CKAN.PylonsMigration

Name	PylonsMigration	Chapter	Data
Goal	Migrate away from the Pylons web framework to the more modern Flask framework		
Description	CKAN runs on a deprecated web framework which even though it has worked well it starting to burst at the seams. CKAN needs to move to a different web framework in order to be able to stay up to date, stay secure and be able to continue improving.		
Rationale	Security and continued improvements are not possible if CKAN sticks to a web framework which relies on old software that cannot be upgraded without breaking the web framework (and thus CKAN)		

5.5.2 Features

FIWARE.Feature.Data.MultilingualSupport.DCAT

Name	Multilingual support DCAT	Chapter	Data
Goal	Ensure that DCAT metadata labels, supported by the CKAN dcat extension can be translated.		
Description	Allow translations of DCAT metadata labels so that when a visitor looks at DCAT published dataset all of the metadata will be in the localised language (based on the CKAN setting or user's choice) if it has been translated.		
Rationale	DCAT and DCAT application profiles have a standardised vocabulary in English of metadata labels that will look out of place in a localised CKAN instance.		

FIWARE.Feature.Data.CKAN.Release24

Name	Release 2.4	Chapter	Data
Goal	Help, support and ensure features and work items are released/updated in an official release		
Description	CKAN is an open source project and as such it is necessary to work closely with the developer community to get new features and work items into a release or ensure		

	that they stay in the release. It is also important to make sure the extensions which fi-ware uses are updated and migrated when a new version is released.
Rationale	Maintain fi-ware as an official part of CKAN (or keep extensions up to date so that fi-ware can use the most recent version of CKAN instead of being stuck on an older version)

FIWARE.Feature.Data.CKAN.DCAT.Harvest

Name	DCAT	Chapter	Data
Goal	Make sure the CKAN extensions harvester and dcat integrate well together		
Description	Make sure that the harvester extension ckanext-harvester and the dcat extension ckanext-dcat work well together so that external dcat endpoints can be harvested automatically at configured intervals, e.g. every week, month etc.		
Rationale	DCAT catalog publishing will most likely be consumed via the CKAN harvester extension and they therefore need to work well together.		

FIWARE.Feature.Data.CKAN.DOI.ManualRegistration

Name	DOI Manual REgistration	Chapter	Data
Goal	Allow publishers of datasets to manually provide DOI numbers when uploading or adding datasets		
Description	Add a field for dataset creation (create or update dataset) and render it as a link to the DOI registered object, e.g. at dx.doi.org. This should all be on a dataset level, not resource level.		
Rationale	Not all DOI publishers allow CKAN to automatically register or dataset publishers might already have a DOI assigned and just need to add it.		

FIWARE.Feature.Data.CKAN.DOI.AutomaticRegistration

Name	DOIAutomaticRegistration	Chapter	Data
Goal	Automatically register for a DOI (Digital Object Identification) at a given provider if requested		
Description	When a dataset is created and user asks for it to have a DOI (need to evaluate whether this should always happen or not) CKAN should communicate with a pre-configured DOI api endpoint and get a DOI number issued/assigned to the dataset.		
Rationale	This makes the life of dataset publishers easier as manually registering for a DOI is not necessary if API is known and can be handled on publication		

FIWARE.Feature.Data.CKAN.MultilingualSupport.TranslationInterface

Name	Translation Interface	Chapter	Data
Goal	Ensure extensions are able to translate their text and be processed by CKAN.		
Description	This Feature provides an Interface to allow CKAN extensions that have been created to be able to translate their text. It also ensures that this text is able to be processed by CKAN		
Rationale	Multi language support for FIWARE is essential to the success of the project as users exist across a wide geographical area. The FIWARE project covers several different languages and it is important these users have access to the same data in their native language.		

FIWARE.Feature.Data.CKAN.MultilingualSupport.MultilingualMetadata

Name	MultilingualMetadata	Chapter	Data
Goal	The aim of this Feature is to ensure Support multilingual metadata is supported in CKAN		

Description	Main changes required for this are in ckanext-fluent, changes are required in core ckan for the templates to use the translated metadata, this is done in https://github.com/ckan/ckan/pull/2729 and is pending review and merging.
Rationale	Multi language support for FIWARE is essential to the success of the project as users exist across a wide geographical area. The FIWARE project covers several different languages and it is important these users have access to the same data in their native language.

FIWARE.Feature.Data.CKAN.EasyInstallAndConfiguration.ConfigurableSettings

Name	Configurable Settings	Chapter	Data
Goal	Allow FIWARE administrators to define what settings can be configured by users		
Description	This allows FIWARE administrators (i.e. those managing the FIWARE cloud) to configure what settings can be configured by instance creators. These settings do not include cloud infrastructure settings like database connections and Solr connections, but things like title, description etc.		
Rationale	CKAN has a lot of different settings that can be configured which will change based on extensions or updates to CKAN. These will have to be configurable to FIWARE cloud maintainers while others will be managed by the maintainers.		

FIWARE.Feature.Data.CKAN.EasyExtensionInstallAndConfiguration.ConfigurableSettings

Name	Extension configuration configurable settings	Chapter	Data
Goal	Allow FIWARE administrators to define what settings can be configured by users		
Description	This allows FIWARE administrators (i.e. those managing the FIWARE cloud) to configure what settings can be configured by instance creators. These settings do not include cloud infrastructure settings like database connections and Solr connections, but things like title, description etc.		

Rationale	CKAN has a lot of different settings that can be configured which will change based on extensions or updates to CKAN. These will have to be configurable to FIWARE cloud maintainers while others will be managed by the maintainers.

FIWARE.Feature.Data.CKAN.Release25

Name	CKAN Release 2.5	Chapter	Data
Goal	Help, support and ensure features and work items are released/updated in an official release		
Description	CKAN is an open source project and as such it is necessary to work closely with the developer community to get new features and work items into a release or ensure that they stay in the release. It is also important to make sure the extensions which fi-ware uses are updated and migrated when a new version is released.		
Rationale	Maintain fi-ware as an official part of CKAN (or keep extensions up to date so that fi-ware can use the most recent version of CKAN instead of being stuck on an older version)		

FIWARE.Feature.Data.CKAN.DCAT.DCATAP11

Name	DCATAP11	Chapter	Data
Goal	The goal is to Make sure that CKAN is Compatible with DCATAP11		
Description	FIWARE NGSI as an API for in-time context information from cities, CKAN as open data platform, and definition of data models for different city verticals, starting with CitySDK models. The fact is that the vision is shifting towards DCAT-AP as the way to describe datasets and CKAN as the open source reference implementation. That means that CKAN should support DCAT-AP (v1.1). If it's not fully compliant, then this would be a priority.		
Rationale	This will ensure better interoperability with other many other fi-ware projects, for example it will allow us to parse a list of datasets into CKAN.		

FIWARE.Feature.Data.CKAN.Harvesting.Sevilla

Name	Harvesting Sevilla	Chapter	Data
Goal	Ensure that the FIWARE instance of CKAN contains all the data which has been published by Sevilla and ensure that it is updated automatically.		
Description	The aim of this ticket is to ensure that the FIWARE instance of CKAN is able to successfully harvest files from Sevilla's own portal. The process will be set up to run automatically so that file will automatically be added to the main portal at regular intervals.		
Rationale	The rationale is to ensure that the FIWARE portal is making the most of the European resources that are available and that the FIWARE portal provides information for people to access open data resources in the European Union.		

FIWARE.Feature.Data.CKAN.PylonsMigration.Audit

Name	PylonsMigrationAudit	Chapter	Data
Goal	To have clearly documented the necessary steps that need to be taken.		
Description	Audit of Pylons use in the CKAN code base, including: code occurrences potential replacements or refactorings needed plugins toolkit objects that need to be migrated plugin interfaces that may be affected by the change		
Rationale	To have documented the necessary steps that need to be taken so all CKAN developers are on the same page regarding the migration roadmap. This will involve proof of concept branches, discussion the development meetings etc.		

FIWARE.Feature.Data.CKAN.PylonsMigration.FlaskMiddleware

Name	PylonsMigration Flask Middleware	Chapter	Data
Goal	The aim here is to create Flask middleware with the Flask specific logic		
Description	As part of the current middleware stack for CKAN, we will create a new Flask based one which will route incoming requests to Flask based controllers. It should support the current authorization transparently.		
Rationale	The rationale here is to Create Flask middleware which that can run side by side with the Pylons one and operate seamlessly.		

FIWARE.Feature.Data.CKAN.PylonsMigration.FlaskMigrate

Name	Flask Migrate	Chapter	Data
Goal	Move selected controllers to use the Flask stack, including routing and template rendering		
Description	This is a major piece of work and cannot be estimated or described accurately at this stage. Therefore, specifics will be added later. They are strongly dependent on previous tasks.		
Rationale	These will sit alongside the current Pylons based ones. Part of this work might involve refactoring the controllers actions to move logic to the model or logic layer.		

FIWARE.Feature.Data.CKAN.PylonsMigration.ReferenceControllers

Name	Reference Controllers	Chapter	Data
Goal	To implement a Flask based reference implementation of several controllers.		
Description	Once the routing and controllers pattern is agreed, migrate a frontend view controller to Flask. That will involve handling the templates and the helpers, so will add some complexity to the API one. The goal is to have a reference implementation that can be later replicated on all the other frontend controllers.		

Rationale	It is important to have a reference implementation of the controllers that we can then used to migrate the rest of the controllers

FIWARE.Feature.Data.CKAN.DCAT.OASCSupport

Name	DCAT OASC Support	Chapter	Data
Goal	The goal here is to make sure that CKAN is able to offer OASC Support		
Description	FIWARE has come to an agreement with the Paneuropean Open Data Portal [2]. One of the elements here is to use NGSI as a way to publish in time context data from cities. But this requires that DCAT-AP is fully able to support OASC principles, which in my view is not the case. We have to work on a DCAT extension to support all the OASC mechanisms (NGSI, Data models), and it would be really important for FIWARE and CKAN to support that extension. We are now launching an Industry Specification Group at ETSI to define that, which might eventually be sent into ISA to be officialised.		
Rationale	This will allow for better interoperability with the other fi-ware projects, for example it will allow us to parse a list of datasets into CKAN.		