Deliverable D200.7

Operating environment components

WP 200

Project Acronym & Number:	FIspace – 604 123
Project Title:	FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics
Funding Scheme:	Collaborative Project - Large-scale Integrated Project (IP)
Latest version of Annex 1:	2013-10-03
Start date of the project:	01.04.2013
Duration:	24
Status:	Draft
Editor:	Said Rahma (ATOS)
Contributors (to the "R" part of the deliverable[1]; ordered by project partner)	**ATOS**: Said Rahma Rodriguez, Roberto Castillo Jimenez **IBM**: Gidon Gershinsky **ASTON**: Monika Solanki
Document Identifier:	FIspace-D200.7-FIspace_Integrated_Release_V3-CSB-v0.4.docx
Date:	27.02.2015
Revision:	004

Project website address: http://www.FIspace.eu

1 Contributors to FIspace code ("P") include ATB, UDE, IBM, ATOS, KOC, TOG, AST, NKUA, UPM and LimeTri; contributing persons are listed at https://bitbucket.org/fispace/profile/members

The FIspace Project

Leveraging on outcomes of two complementary Phase 1 use case projects (FInest & SmartAgriFood), aim of FIspace is to pioneer towards fundamental changes on how collaborative business networks will work in future. FIspace will develop a multi-domain Business Collaboration Space (short: FIspace) that employs FI technologies for enabling seamless collaboration in open, cross-organizational business networks, establish eight working Experimentation Sites in Europe where Pilot Applications are tested in Early Trials for Agri-Food, Transport & Logistics and prepare for industrial uptake by engaging with players & associations from relevant industry sectors and IT industry.

Project Summary

As a use case project in Phase 2 of the FI PPP, FIspace aims at developing and validating novel Future-Internet-enabled solutions to address the pressing challenges arising in collaborative business networks, focussing on use cases from the Agri-Food, Transport and Logistics industries. FIspace will focus on exploiting, incorporating and validating the Generic Enablers provided by the FI PPP Core Platform with the aim of realising an extensible collaboration service for business networks together with a set of innovative test applications that allow for radical improvements in how networked businesses can work in the future. Those solutions will be demonstrated and tested through early trials on experimentation sites across Europe. The project results will be open to the FI PPP program and the general public, and the pro-active engagement of larger user communities and external solution providers will foster innovation and industrial uptake planned for Phase 3 of the FI PPP.

Project Consortium

- DLO; Netherlands
- ATB Bremen; Germany
- IBM; Israel
- KocSistem; Turkey
- Aston University; United Kingdom
- ENoLL; Belgium
- KTBL; Germany
- NKUA; Greece
- Wageningen University; Netherlands
- PlusFresc; Spain
- FloriCode; Netherlands
- Kverneland; Netherlands
- North Sea Container Line; Norway
- LimeTri; Netherlands
- BO-MO; Slovenia
- MOBICS; Greece
- Fraunhofer IML; Germany
- Q-ray; Netherlands
- FINCONS; Italy
- Kühne + Nagel; Switzerland
- University Duisburg Essen; Germany
- ATOS; Spain
- The Open Group; United Kingdom
- CentMa; Germany
- iMinds; Belgium
- Marintek; Norway
- University Politecnica Madrid; Spain
- Arcelik; Turkey
- EuroPoolSystem; Germany
- GS1 Germany; Germany
- Mieloo & Alexander; Netherlands
- OPEKEPE; Greece
- Innovators; Greece
- CIT; Spain
- SDZ; Germany
- Snoopmedia; Germany
- EECC; Germany
- CBT; Spain

More Information

Harald Sundmaeker (coordinator)	e-mail:	sundmaeker@atb-bremen.de
Bert Vermeer (deputy coordinator)	e-mail:	bert.vermeer@wur.nl
Project Website	Web link:	http://www.fispace.eu/

Dissemination Level

PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	**X**
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Change History

Version	**Notes**	**Date**
001	Creation of the document CSB component	09.12.2014
002	Update of the contents, added new contents and chapters related to the monitoring service development (architecture overview, monitoring services, installation, deployment and configuration) Minor changes related to rephrasing and typo error Update of the abbreviation table, update of the references table	27.01.2015
003	Internal review process, checking URL links to the FIspace Web online documentation Update of the abbreviation table, update of the references table Final version ready for submission of approved document	06.02.2015
004	Update of the coordinator information in the section "*More Information*" Added FIspace development repository and documentation references, formatting improvement Final version ready for submission to EC	27.02.2015
005		
006		

Abbreviations

AAA	Authentication, Authorisation, and Accounting
ACSI	Artifact-Centric Service Interoperation
AdvB	Advisory Board
AJAX	Asynchronous JavaScript + XML
API	Application Programming Interface
App	Software Application
B2B	Business-to-business
B2C	Business-to-Consumer
BCM	Business Collaboration Module in FIspace
BCO	Business Collaboration Objects in FIspace
BE	Business Entities
BPPC	Business Process Participant Configuration
BSS	Business Support Systems
CDR	Charging Detailed Records
CEP	Complex Event Processing
CSB	Cloud Service Bus
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
D	Deliverable
DAO	Data Access Object
DB	Database
DoW	Description of Work
EC	European Commission
EDI	Electronic Data Interchange
EE	Experimentation Environment
e.g.	Exempli gratia = for example
EPA	Event Processing Agent
EPM	Event Processing Module in FIspace
ESB	Enterprise Service Bus
EU	European Union
FIA	Future Internet Assembly
FI-PPP	Future Internet Public Private Partnership
FP7	Framework Programme 7
GA	Grant Agreement
GE	Generic Enabler
GUI	Graphical User Interface
HTML	HyperText Markup Language
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IDM	Identity Management
i.e.	id est = that is to say
IE	Integration Environment
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
I/O	Input / Output
IoT	Internet of Things
IP	Intellectual Property
IP (protocol)	Internet Protocol
IPR	Intellectual Property Rights
IPsec	Internet Protocol Security
IT	Information Technology
ITU	International Telecommunication Union
ISO	International Standardization Organisation
J2SE	Java 2 Platform, Standard Edition
JDK	Java Development Kit
JDT	Related to Eclipse Java Development Tools
JMX	Java Management Extensions
JRE	Java Runtime Environment
JS	JavaScript
JSON	JavaScript Object Notation
JSP	Java Server Page
JVM	Java Virtual Machine
KPI	Key Performance Indicator
LPA	Logistics Planning Application
M	Month
MTBF	Mean Time Between Failures
MVC	Model–View–Controller
OASIS	Organization for the Advancement of Structured Information Standards
OAuth	Open standard Authentication protocol
OMG	Object Management Group
OSS	Operational Support Systems
P2P	Peer-to-peer
PaaS	Platform as a Service
PDE	Related to Eclipse Java Development Tools
PE	Production Environment
PIA	Product Information App

PIE	Preliminary Integration Environment
PKI	Public Key Infrastructure
PM	Person Month
POM	Project Object Model (used by maven tools)
Proton	IBM Proactive Technology Online
QoS	Quality of Service
RBAC	Role-Based Access Control
RCP	Rich Client Platform
REST	Representational State Transfer
RFC	Request for Comments
RSS	Revenue Sharing System
RTD	Research and Technological Development
SaaS	Software as a Service
SDI	System and Data Integration layer in FIspace
SDK	Software Development Kit
SME	Small and Medium Sized Enterprise
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOA-RM	(OASIS) Reference Model for Service Oriented Architecture
SPT	Security, Privacy and Trust Framework
SSH	Secure Shell
SSL	Secure Sockets Layer
SSO	Single Sign On
ST	Sub-Task
SWT	Standard Widget Toolkit
T	Task
TCP	Transmission Control Protocol
TIC	Tailored Information for Consumers
TLS	Transport Layer Security
TPM	Transport Planning Module
UAA	User Management, Authentication and Authorisation
UI	User Interface
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
USDL	Unified Service Description Language
VM	Virtual Machine
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WLAN	Wireless Local Area Network
WP	Work Package
WS	Web Service
WSDL	Web Services Description Language
XLS/XLSX	Microsoft Excel file Format
XML	eXtensible Markup Language
XSD	XML Schema Definition

Table of Contents

List of Figures

List of Tables

1 Introduction

This document aims at describing the third release (V3) of the FIspace, encompassing the implementations along with usage guidance and technical documentation of each FIspace component.

It reports on the description concerning the **Cloud Service Bus core component** that is part of the operative environment, the description of the development and implementation of the CSB core components that is part of the FIspace platform.

The Operating Environment ensures the technical interoperability and communication of (possibly distributed) FIspace components and FIspace Apps and the consistent behaviour of the FIspace, including:

- **A Cloud Service Bus (CSB)** to support the interaction of FIspace components and Apps, which is based on peer-to-peer overlay technology, supporting (1) eventual consistency, (2) events bus, (3) management logic, (4) Pub/Sub abstraction for information dissemination, (5) a bulletin board abstraction for filtering and orchestration, (6) queues supporting various QoS for delivery and execution (e.g., once only or multiple readers).
- **Replication and consistency service** to ensure fault-tolerance and transaction support, which is partition tolerant and guarantees strong consistency (when needed).
- Facilitation of the **management** of the “composed service (application)” lifecycle, based on IaaS Cloud related OSS and BSS (planned to be provided by FI-WARE);
- **Operational registry** for maintaining runtime attributes and supporting real-time operations.
- **Multi-tenancy support**, with the least effort from the developers (both FIspace developers and App developers).
- **Monitoring** of KPIs and health, automate the operation, enforce the SLA, facilitate the problem determination, continuous optimizing the runtime.

The Operating Environment provides automation supporting the application lifecycle and support a “scale out” design model that is decentralized with redundancy for failure tolerance and auto recovery. It supports eventual consistency, as well as strong consistency asynchronous models.

Online documentation for Operating Environment: http://dev.fispace.eu/doc/wiki/csb

1.1 Scope

The aim of this document is mainly to describe and detail the **FIspace CSB core component integrating the monitoring capabilities** at development and implementation level, giving detailed and technical information related to the design and the implementation as well as information about the related technologies and standard taken as a .reference to build each component.

Along this development activities and tasks, there is a set of resources, online documentation, tutorial and other external resource that refer to the Generic Enablers that can provide more technical information and user guides for the community and people who want to use the FIspace platform for Business collaboration or developers who want to create and develop business application (Apps developer) for a specific domain of application.

Table 1 shows the links to other online resources related to FIspace project and FI-WARE.

Description	Link
FIspace Business collaboration web site	http://www.fispace.eu/
FIspace Developer Documentation web site	http://dev.fispace.eu/doc/wiki/Home
FIspace Deliverables web site	http://www.fispace.eu/deliverable.html
FIspace Tutorial web site	http://www.fispace.eu/tutorials.html
FIWARE web site	http://www.fi-ppp.eu/projects/fi-ware/
FIWARE Catalogue of the Generic Enablers (GEs)	http://catalogue.fi-ware.org/
FIWARE community web site	http://www.fi-ware.org/community/

Table 1: Other FIspace and FIWARE resources

Table 2 shows the links to the Wirecloud online documentation.

Description	Link
FIWARE - Catalogue - Application Mashup - Wirecloud	http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud
FIWARE - Catalogue - Application Mashup - Wirecloud Documentation	http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud/documentation
FIWARE - Application Mashup - Wirecloud - User and Programmer Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Application_Mashup_-_Wirecloud_-_User_and_Programmer_Guide
Dashboard - Wirecloud home page	http://conwet.fi.upm.es/wirecloud/
Dashboard - The WireCloud Mashup Platform	http://conwet.fi.upm.es/docs/display/wirecloud/The+WireCloud+Mashup+Platform
Dashboard - Welcome to CoNWeT-Wirecloud Confluence	http://conwet.fi.upm.es/docs/dashboard.action
Dashboard - User Guide	http://conwet.fi.upm.es/docs/display/wirecloud/WireCloud+User%27s+Guide
Dashboard - WireCloud Installation	http://conwet.fi.upm.es/docs/display/wirecloud/

and Administration Guide	Wire-Cloud+Installation+and+Administration+Guide

Table 2: Wirecloud online documentation

Table 3 shows the links to the WStore online documentation.

Description	Link
FIWARE - Catalogue - Store - WStore	http://catalogue.fi-ware.org/enablers/store-wstore
FIWARE - Catalogue - Store - WStore Documentation	http://catalogue.fi-ware.org/enablers/store-wstore/documentation
FIWARE - Store - W-Store - User and Programmer Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Store_-_W-Store_-_User_and_Programmer_Guide
FIWARE - Store - W-Store - Store - W-Store - Installation and Administration Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Store_-_W-Store_-_Installation_and_Administration_Guide

Table 3: Store online documentation

Table 4 shows the external development tools references.

Description	Link
Java Environment, JVM, JRE, JDK (Oracle)	http://www.oracle.com/technetwork/java/javase/downloads/index.html
Eclipse IDE (Integrated Development Environment)	https://www.eclipse.org/ , https://www.eclipse.org/downloads/
Maven	http://maven.apache.org/ , http://maven.apache.org/download.cgi

Table 4: External development tools references

Table 5 shows the FIspace development repository and documentation references based on the bitbucket tools for collaborative development.

Bitbucket is a hosting site for the distributed version control systems (DVCS) Git (http://git-scm.com/) and Mercurial (http://mercurial.selenic.com/). The service offering includes an issue tracker and wiki, as well as integration with a number of popular services such as Basecamp, Flowdock, and Twitter.

Description	Link
Bitbucket FIspace repository home page	https://bitbucket.org/fispace
Bitbucket FIspace core component home page	https://bitbucket.org/fispace/core/wiki/Home
Bitbucket FIspace Roadmap page	https://bitbucket.org/fispace/core/wiki/roadmap

Table 5: Bitbucket collaborative environment for FIspace development

1.2 Intended audience

The main interest groups of this deliverable are the participating teams and the responsible partners of FIspace project involved in the development activities, setup and preparation of the development phase. This document is relevant to the software engineer, programmers and developers who are the persons directly involved in the development, participating effectively on the design and implementation of the FIspace platform and the underlying components and sub-systems who want to know more about some technical information intrinsic to the FIspace platform.

At the technical level this document is relevant to: system architects; information systems designers; system developers and application developers; software engineers; other audiences who provide design services and applications using relevant standards and the recommendations of standards bodies like IETF, ITU, ISO, W3C, etc.

Partners involved in the integration tasks include: system integrators; people to test, validate and evaluate the FIspace platform and associated systems; can be also interested.

1.3 General remark

This document follows the ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards w.r.t. the usage of the word “shall”. The word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this specification.

This document describes the corresponding core components involved in the FIspace core platform. It presents the development currently done and the corresponding implementation, the main features developed, as well as the related technologies and environment requirements.

In most of the following sections the structure is organized as:

- **Overview**: provides an overall introduction to the component, a description, of the internal architecture and features among other.
- **Interfaces or Application programming interface (API)**: describes the API accessible for the users or entities of the component (typically applications, but a component may also be used by other components).
- **Information model**: describes or specifies the component from an information perspective describing information objects of the component domain.
- **Interaction model**: describes or specifies main usage component “scenarios” associated with the component/GEs, sequence diagrams.

- **High level composite architecture**: describes or shows the main components constituting the set of components (this perspective is optional, since some component consists of only one main component).

Notice that some components only need to describe some of the item above described.

2 Cloud Service Bus (CSB)

2.1 Overview

The Cloud Service Bus (CSB) layer of FIspace Operating Environment is the middle-ware that enables integration of different FIspace service components and applications. CSB provides a set of integration interfaces and qualities of service, required to support a wide spectrum of information exchange scenarios, ranging from best effort notifications, to guaranteed delivery of transactional data.

The Cloud Service Bus has virtually unlimited scalability, both in the number of support-ed end-points, and in the number of communication channels allowed by the Bus transport fabric.

2.2 Interfaces / API

The CSB provides the following main services and corresponding API:

- Publish/subscribe
- Queuing
- Synchronous Request/Response
- Asynchronous Request/Response
- Time-limited asynchronous Request/Response
- Network time service

The description of the API operation is detailed in the next subsections.

2.2.1 Publish/subscribe

Topic-based publish/subscribe is an efficient tool for scalable message distribution to multiple clients. A topic is a virtual address, where any client can send (‘publish’) a message, and any client can listen (‘subscribe’) to the published messages. Pub-lish/subscribe is optimal for fast and scalable distribution of real-time data. However, it provides either best effort or partially reliable qualities of service. The mode of data delivery is synchronous - receivers need to be online when a message is published, and have to keep up with the transmission rate.

2.2.2 Queuing

Unlike Topics, Queues are physical objects, with an associated persistent storage on a hard disk. Any client can send a message to a queue, where it will be written to a disk and replicated for high availability. Any client can consume the queued messages. Queuing is the most reliable and consistent data delivery method, supporting persis-tence, identical message ordering and high availability qualities of service. Also, queues allow for fully asynchronous data exchange – much like in email, receivers don’t have to be online when messages are sent; they can join a queue any time later and retrieve the stored messages. In terms of throughput, one queue can support around a thou-sand messages per second (just a rough estimate). Topics are a few times faster, but less reliable.

When using messaging, choose either pub/sub or queuing for each channel of communication, according to your application quality of service requirements.

2.2.3 Synchronous Request/Response

This mechanism allows clients synchronously query one another for a value of specific attributes or for any other information. Both query (request) and response are POJO objects, in a format that is mutually agreed upon between requesting and responding applications or components. This flexibility allows any kind of input data to be provided in the request, and any kind of output data delivered back as a response. The requester sends a request and waits for a response, up to a specified timeout. The responder receives the request, produces a response and sends it back as soon as possible.

Synchronous request/response is optimal for applications where the response information can be retrieved quickly by the responder, so the requester won't have to wait a long time on a blocking query.

2.2.4 Asynchronous Request/Response

Sometimes, a response cannot be produced immediately – it might involve heavy on-spot data processing, or an interaction with a third party, or other factors with unpredictable timing. Moreover, some applications might need multiple responses to a single request – for example, future notifications upon attribute value change. Asynchronous request/response mechanism supports this functionality, by allowing requester to set a response listener callback, and by enabling the responder to send response messages at any time, without a limit of number of responses for each request.

2.2.5 Time-limited asynchronous Request/Response

This mode addresses the following scenario – the time required to produce a response is more or less known (predictable), but is long, so the requester is not willing to be blocked on a synchronous request. In this case, the requester should send an asynchronous query, and set a CSB response timer. If no response to this specific request arrives with the specified time, the requester will be called with WAIT_TIMEOUT notification.

2.2.6 Network time service

CSB provides a simple implementation of the NTP protocol, which finds the time difference between clocks of different machines. This service allows all machines, running CSB, to use unified time stamping that refers to the same clock (that of the coordinator machine).

2.3 Code samples

Here go the sender and receiver code samples that demonstrate the API usage:

Sender:

```
import com.ibm.csb.client.CSBException;

import com.ibm.csb.client.CSBFactory;

import com.ibm.csb.client.Message;
import com.ibm.csb.client.Queue;
import com.ibm.csb.client.QueueSender;
import com.ibm.csb.client.Session;
import com.ibm.csbx.client.ObjectMessage;

public class QObjSendSample
{
```

Here we define a detailed event listener that looks for a number of defined events and prints them differently.

```
        static final class MyEventListener implements QueueEventListener{
                String _queueName;

                public MyEventListener (String queue)
                {
                        _queueName = queue;
                }

                public void onEvent(QueueEvent event)
                {
                        // Message Ack - means all messages up to this
                        // number are persisted in the queue file storage
                        if (event.getType() == EventType.SND_MESSAGE_ACK) {
                                System.out.println ("QueueSender on " + _queueName +
                                        " GOT ack for messages up to "+ event.getMessageNumber());
                        }
                        else if (event.getType() == EventType.SESSION_SUSPENDED) {
                                System.out.println ("Session suspended. CSB Node is disconnected
                        from
                                        Coordinator");
                                System.out.println ("Don't send messages, wait for session re-
sumption");
```

The Node disconnected from Coordinator – most likely due to a networking problem. The CSB session is suspended; 'message send' and other API calls will throw an exception, since it's impossible to deliver data or commands across the network. You can set a (volatile) boolean flag here, and check it before sending messages, creating queues, etc.

```
                        }
                        else if (event.getType() == EventType.SESSION_RESUMED) {
                                System.out.println ("Session resumed. Feel free to send messages
again");
```

The network is working again, the Node is automatically reconnected to Coordinator. You can reset the boolean flag and continue using the CSB APIs.

```
                        }
                        else {
                                System.out.println(_queueName + " queue event = " + event);
                        }
                }
        }

        public static void main(String[] args)
        {
                CSBFactory factory;
                try {
                        factory = CSBFactory.getInstance();
                } catch (CSBException e) {
                        e.printStackTrace();
                        return;
                }

                Session client_session;
                try {
                        // Using default Node address/port: 127.0.0.1/9199
                        client_session = factory.createSession();
                } catch (CSBException e) {
                        e.printStackTrace();
                        return;
                }

                System.out.println ("Created a client " + client_session.getClientId());

                // Creates queue
                Queue queue;
                try {
                        queue = client_session.createQueue("queue1",null);
```

Notice – unlike with topics, only one client actually creates the physical queue. If another client creates the same queue later, it will get a reference to the existing queue.

```
                } catch (CSBException e) {
                        e.printStackTrace();
                        return;
                }
```

```
System.out.println ("Created Q queue1");

QueueSender qs;
try {
       qs = client_session.createQueueSender(queue,
                      new MyEventListener(queue.getQueueName()),
                      null);
} catch (CSBException e) {
       e.printStackTrace();
       return;
}

SampleMessage sm = new SampleMessage();

String str = "Queue message from "+client_id+". Message #";

for (int i=0; i<10; i++) {

       sm.setText(str+i);
       sm.setInt(i);

       Message message;
       try {
              message = ObjectMessage.createObjectMessage(sm);
       } catch (CSBException e) {
              e.printStackTrace();
              return;
       }

       try {
              qs.send(message,600000); // msg expiration in queue: 10 minutes
```

Each queue message has a finite lifetime, otherwise a queue will fill up. This sample sends the messages with 10-minute lifespan (expiration is specified in milliseconds). You can use any message lifetime, short or long, depending on your application requirements. Notice: pub/sub messages have expiration time 0, they are not stored anywhere.

```
       } catch (CSBException e) {
              e.printStackTrace();
              return;
```

```
                        }

                        System.out.println ("sent msg "+i);

                        Thread.sleep(100);
                }
                System.out.println ("Done...");
        }
}
```

Receiver:

```
import com.ibm.csb.client.CSBException;
import com.ibm.csb.client.CSBFactory;
import com.ibm.csb.client.Message;
import com.ibm.csb.client.MessageListener;
import com.ibm.csb.client.Queue;
import com.ibm.csb.client.QueueReceiver;
import com.ibm.csb.client.Session;
import com.ibm.csb.client.QueuePosition.Reference;
import com.ibm.csbx.client.ObjectMessage;

public class QObjReceiveSample
{
        static final class MyMessageListener implements MessageListener{

                int nmsg;
                long size;
                String queueName;

                MyMessageListener(String q_name) {
                        nmsg = 0;
                        size=0;
                        queueName = q_name;
                }

                public void onMessage(Message msg) {
                        nmsg++;
                        size+=msg.getLength();

                        //boolean res = myQueueR.deleteMessage(msg);
                        //if   (!res)   System.out.println   ("Failed   to   delete   message
"+msg.getMessageNumber());
```

A message can be explicitly deleted from a Queue. This helps to clean a queue and prevent it from filling up, so do use it if you know the message is not required anymore by you or other queue receivers.

```
                        System.out.println("Got message from Queue "+queueName);

                        SampleMessage sm;
                        try {
                                sm = (SampleMessage) ObjectMessage.getObject(msg);
                        } catch (CSBException e) {
                                e.printStackTrace();
                                return;
                        }

                        System.out.println("Message textField: "+sm.getTextField());
                        System.out.println("Message intField: "+sm.getIntField());

                        System.out.println(nmsg+" MESSAGES Received, total size "+size+" B");
                }
        }

        public static void main(String[] args) throws CSBException, InterruptedException
        {

                CSBFactory factory = CSBFactory.getInstance();

                //Unique client ID. Use your own.
                String client_id = "Client"+System.currentTimeMillis();

                // Using default Node address/port: 127.0.0.1/9199
                client_session = factory.createSession();

                System.out.println ("Created a client " + client_session.getClientId());

                // Create the queue
                Queue queue = client_session.createQueue("queue1",null);

                System.out.println ("Found Q queue1");
```

```
            MyMessageListener m_list = new MyMessageListener("queue1");

            // Create q receiver
            client_session.createQueueReceiver(queue, m_list,
                                  QueueReceiver.createQueuePosition(Reference.OLDEST,0),
```

When joining a queue, you need to specify initial position – do you want to get all old messages, already stored in the queue, or only the most recently queued message. In both cases, you will start receiving all future messages (See the Javadoc).

```
                                  null,null);
        }
}
```

2.4 Information model

CSB messages have simple structure – basically, a payload (byte array), a sequence number and the name of the sender.

However, CSB provides tools for conversion of complex messages (Serializable Object) into a CSB message, and back:

```
SampleMessage sm = new SampleMessage();
message = ObjectMessage.createObjectMessage(sm);
SampleMessage sm = (SampleMessage) ObjectMessage.getObject(msg);
```

Here goes another example, of complex Request/Response objects that can be transmitted by CSB query interfaces:

RequestSample is a simple emulation of FarmerApp, that sends a request for advice (say, how much water to use for today's sprinkling of wheat, grown in Toulouse, per acre), to an advisor app (ResponseSample), and gets the advice as a response. The farmer and advisor have agreed on the following format of request and response objects:

```
public class SampleRequestObject implements java.io.Serializable{
        public static final long serialVersionUID = 1L;

        private String _queryAttribute;
        private String _cropName;
        private String _location;
        private int _soilHumidityPcnt;

        public SampleRequestObject(String queryAttribute, String cropName,
                int soilHumidityPcnt, String location) {
```

```
            _queryAttribute = queryAttribute;
            _cropName = cropName;
            _location = location;
            _soilHumidityPcnt = soilHumidityPcnt;
    }

    public String getQueryAttribute() {
            return _queryAttribute;
    }

    public String getCropName() {
            return _cropName;
    }

    // soil humidity is percents
    public int getSoilHumidityPcnt() {
            return _soilHumidityPcnt;
    }

    // where the crops are grown
    public String getLocation() {
            return _location;
    }
}
public class SampleResponseObject implements java.io.Serializable{
    public static final long serialVersionUID = 1L;

    public static final byte OK = 0;
    public static final byte BAD_REQUEST = 1;
    public static final byte FAILED_TO_CREATE_ADVICE = 2;

    private int _value;

    public SampleResponseObject(int value) {
            _value = value;
    }

    public int getValue() {
            return _value;
    }
}
```

2.5 Interaction model

CSB is a core component, without dependencies on other FIspace components or GEs. However, it uses Apache Mina libraries for communication, and Log4j2 libraries for the logging. CSB defines three entities – Coordinator, Node and client. The coordinator and nodes need to be deployed before the clients can run.

The following FIspace components use CSB for communication and integration: **SDI**, **B2B**, **Front-End** and **Store**.

2.6 High level composite architecture

The scalability challenge, posed by the large number of data producers, service providers and consumers in this cloud-based project, is addressed by Peer-to-Peer overlay technology. The CSB core is comprised of Bus Nodes, connected by a peer2peer structured overlay fabric. The CSB clients are the FIspace service components and applications, using the Bus for integration and connectivity. This two-tier architecture allows us to build an enterprise-strength service bus, with extensive functionality set and assured delivery of transactional data. The CBS supports multi-tenancy, enabling creation of separate Bus domains, each using a different P2P overlay structure.

Figure 1 shows the CSB architecture. The blue boxes are the CSB Nodes, connected by a peer-to-peer overlay ring (red dash line). Green blobs are the CSB clients – applications, services or mediations, using CSB for connectivity.

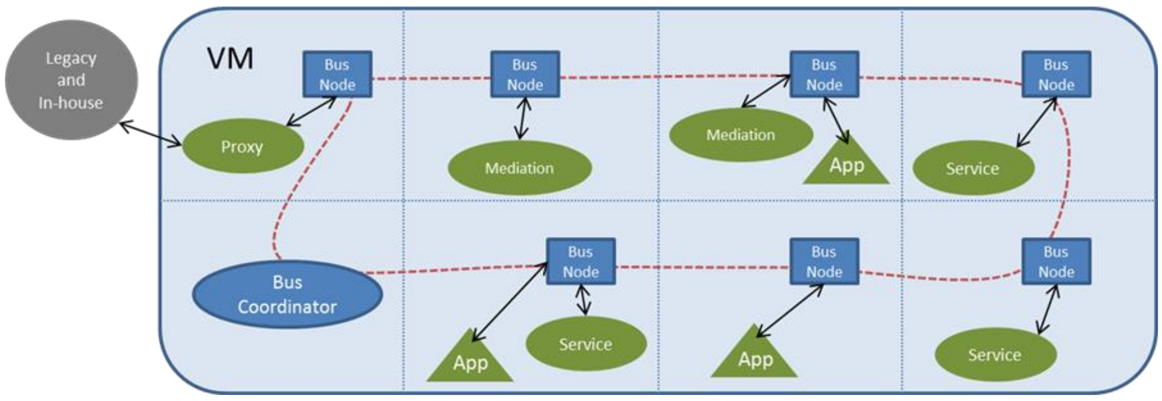


Figure 1: CSB architecture

2.7 Logging

CSB uses SLF4J for logging, with Log2J 2 as the default binding.

2.7.1 Client logging

The logging is configured in the log4j2.xml file, included with the samples.

```
<?xml version="1.0" encoding="UTF-8"?>

<Configuration status="WARN" monitorInterval=10>

  <Appenders>

    <File   name="File1"   fileName="csb_client.log"   bufferedIO="true"   ap-
pend="false">
```

```
      <PatternLayout pattern="%d{dd/MM HH:mm:ss.SSS}  [%t] %-5level %log-
ger{36} - %msg%n"/>
    </File>
  </Appenders>
  <Loggers>
    <logger name="com.ibm.csb.logMessageIO" level="info"/>
    <Root level="info">
      <AppenderRef ref="File1"/>
    </Root>
  </Loggers>
</Configuration>
```

If you want to log every message sent or received in your CSB client, modify

```
<logger name="com.ibm.csb.logMessageIO" level="trace"/>
```

2.7.2 Node logging

The logging is configured in the log4j2.xml file, included with the distribution.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN" monitorInterval=10>
  <Appenders>
    <File  name="File1"  fileName="csb_node.log"  bufferedIO="true"  ap-
pend="false">
      <PatternLayout pattern="%d{dd/MM HH:mm:ss.SSS}  [%t] %-5level %log-
ger{36} - %msg%n"/>
    </File>
  </Appenders>
  <Loggers>
    <Root level="info">
      <AppenderRef ref="File1"/>
    </Root>
  </Loggers>
</Configuration>
```

2.7.3 Coordinator logging

The logging is configured in the log4j2.xml file, included with the distribution.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN" monitorInterval=10>
  <Appenders>
    <File name="File1" fileName="csb_coordinator.log" bufferedIO="true" ap-
pend="false">
```

```
      <PatternLayout  pattern="%d{dd/MM  HH:mm:ss.SSS}   [%t]  %-5level  %log-
ger{36} - %msg%n"/>

    </File>

  </Appenders>

  <Loggers>

    <Root level="info">

      <AppenderRef ref="File1"/>

    </Root>

  </Loggers>

</Configuration>
```

3 Monitoring Service Development

3.1 Overview

The monitoring service of FIspace Operating Environment is the service that enables the monitoring of the different FIspace services and components. The monitoring service provides information about different metrics for each component to allow the feasibility of the platform. The monitoring service has virtually unlimited scalability, and uses the available capabilities offered by the Cloud Service Bus.

On the market there are many solutions to measure and to monitor the performance of a system.

Some of the relevant criteria or characteristics related to a monitoring solution/system in the FIspace context are:

- The cost of monitoring tools: It is suitable to use an open source or low cost solution.
- The solution should be easy to deploy, to configure and to setup.
- The solution should be scalable, reliable, adaptable and extendable.
- The monitoring system should be based on non-intrusive tools on the host to monitoring.
- The monitoring system should allow sending alerts and provide tools for events management.
- The monitoring system should provide reporting tools (graphics, statistics) and should allow to presents the information, data monitor and indicator performances based on web interfaces.

The envisaged tools for monitoring activities in the context of the FIspace platform is based currently on the Zabbix tools [32] this tool seems to fulfil the requirement and it is also used by the FIspace cloud infrastructure provider, for this reason it is the monitoring software tool selected.

Zabbix [32] is an enterprise-class open source distributed monitoring solution for networks and applications. Zabbix is an “All In one“ Solution that offer True Open Source software, Performance monitoring, agents for all platforms, agent-less monitoring, availability and SLA reporting, collection of any data and great graphs and network maps.

Zabbix software is released under the GNU General Public License (GPL) version 2. The formal terms of the GPL can be found at http://www.fsf.org/licenses/ . If Zabbix is used in a commercial context such as profit by its use, it is requested to further the development of Zabbix by purchasing some level of support.

3.2 Monitoring architecture overview

The monitoring software tool selected has been Zabbix. This tool implements client-server architecture to collect and shows the monitoring information from every machine or VM (Virtual Machine).

Figure 2 shows the architecture overview for the monitoring implementation approach and consists to the following elements:

- The agents (the client side) are located in each machine (VM) where it´s necessary to collect the information about its status.
- The server (the server side) is the software where the information is sent from every machine or VM to this server machine (VM).

Afterwards, it is possible to visualize this information in the reports and the graphs offered by the monitoring software tool.

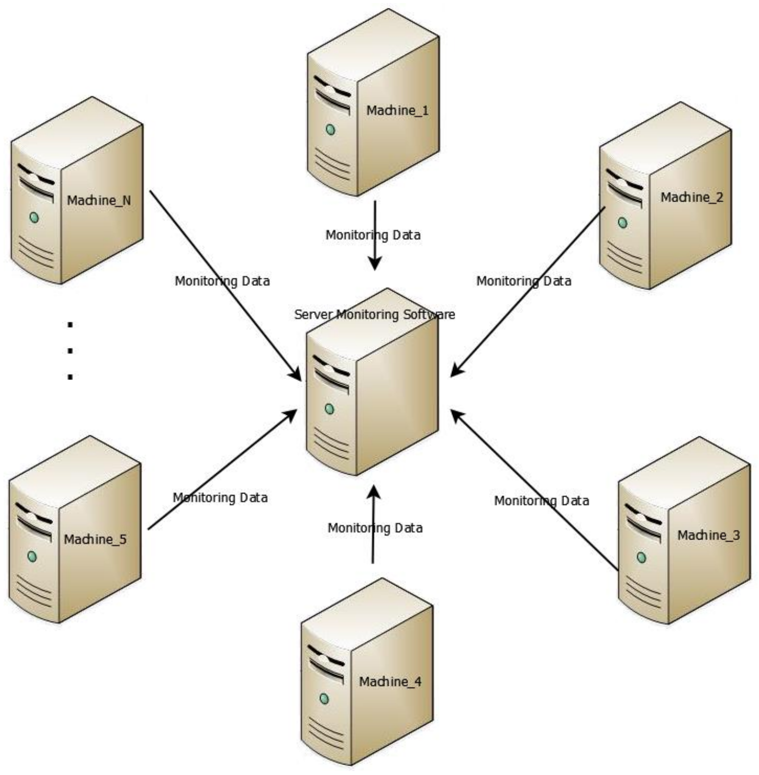


Figure 2: Monitoring architecture overview

Each VM that host the corresponding components are configured to include a monitoring agent. This agent is installed and configured by default to collect a set of data as a list of parameters or key indicators that are scheduled to send these data to the monitoring server based on the JMX and Zabbix technology. The Zabbix server integrates a set of capabilities to report these data graphically and provide a configurable dashboard that can be managed by an administrator. Java Management Extensions (JMX) is a Java technology that supplies tools for managing and monitoring applications, system objects, devices and service-oriented networks.

3.3 Key Performance Indicators (KPIs) and metrics definition

The main goals of monitoring and control system include the following:

- Observe the “health” of IT services.
- Take remedial actions that minimize the impact of service incidents and system events.
- Understand the infrastructure components responsible for the delivery of services.
- Provide data on component or service trends that can be used to optimize the performance of IT services.

Any system performance evaluation relies on a set of key performance indicators, the corresponding data to be acquired, and measured and/or calculated using some specific software tool or ad-hoc middleware. Both, the key performance indicators and the suggested tools, monitoring services developed and implemented are introduced in the next sections.

Currently, there are a set of common metrics that will be able to be monitored in every node and corresponding VM. Table 6 describes the common and generic metrics at client side currently collected by the monitoring tool.

More information concerning the default parameters or KPI managed by the Zabbix tools and product can be found in the corresponding documentation of Zabbix (http://www.zabbix.com/documentation.php).

3.3.1 Metrics at client side

Table 6 describes the current metrics collected at the client side by the standard Zabbix agent deployment.

These metrics are categorized or classified as the following: CPU, File systems, General, Memory, Network Interfaces, OS (Operating System), Performance, Processes, Security and the proper Zabbix agent.

Specific metrics can be necessary to define and to develop as custom metrics related to each FIspace component. This definition process for the custom metrics in each component is being analyzed and currently done for some components. In the section 3.6.1 the first custom metrics are defined and is related to monitories the CSB (Cloud Service Bus) component.

Metric by type of information (Client side)
CPU
Context switches per second
CPU idle time
CPU interrupt time
CPU iowait time
CPU nice time
CPU softirq time
CPU steal time
CPU system time
CPU user time
Interrupts per second
Processor load (1 min average per core)
Processor load (5 min average per core)

Processor load (15 min average per core)
File systems
Free disk space on /
Free disk space on / (percentage)
Free inodes on / (percentage)
Total disk space on /
Used disk space on /
General
Host boot time
Host local time
Host name
System information
System uptime
Memory
Available memory
Free swap space
Free swap space in %
Total memory
Total swap space
Network Interfaces
Incoming network traffic on eth0
Outgoing network traffic on eth0
OS (Operating System)
Host boot time
Host local time
Host name
Maximum number of opened files
Maximum number of processes
Number of logged in users

System information
System uptime
Performance
Context switches per second
CPU idle time
CPU interrupt time
CPU I/O wait time
CPU nice time
CPU software IRQ time
CPU steal time
CPU system time
CPU user time
Interrupts per second
Processor load (1 min average per core)
Processor load (5 min average per core)
Processor load (15 min average per core)
Processes
Number of processes
Number of running processes
Security
Checksum of /etc/passwd
Number of logged in users
Zabbix agent
Agent ping
Host name of zabbix_agentd running
Version of zabbix_agent(d) running

Table 6: Metric at client side

3.3.2 Metrics at server side

Table 7 describes the current metrics collected and monitored at the server side and which correspond essentially to monitories the proper Zabbix server, i.e. the server part where the Zabbix server monitoring tool is deployed.

Metric by type of information (Server side)
Zabbix server
Values processed by Zabbix server per second
Zabbix busy alerter processes, in %
Zabbix busy configuration syncer processes, in %
Zabbix busy db watchdog processes, in %
Zabbix busy discoverer processes, in %
Zabbix busy escalator processes, in %
Zabbix busy history syncer processes, in %
Zabbix busy housekeeper processes, in %
Zabbix busy http poller processes, in %
Zabbix busy icmp pinger processes, in %
Zabbix busy poller processes, in %
Zabbix busy proxy poller processes, in %
Zabbix busy self-monitoring processes, in %
Zabbix busy timer processes, in %
Zabbix busy trapper processes, in %
Zabbix busy unreachable poller processes, in %
Zabbix configuration cache, % free
Zabbix history write cache, % free
Zabbix queue
Zabbix queue over 10m
Zabbix text write cache, % free
Zabbix trend write cache, % free
Zabbix value cache, % free
Zabbix value cache hits

Zabbix value cache misses

Table 7: Metric at server side

3.4 Background on Runtime Monitoring

The implementation of software is governed by requirements which represent an abstraction of the functional expectations from the system. However, the analysis and specification of precise requirements is often error prone. As the size of software grows, it becomes more difficult to test or verify the correctness of a system. Thus, it can be difficult to provide assurances about the state of a software's execution. Continuous monitoring of a running system for specific behaviours, is a complementary approach to increase the assurance of correct execution.

In order to enable monitoring and management, a common approach is to instrument the code. Programmers implement instrumentation in the form of code instructions that monitor specific components in a system (for example, instructions may output logging information to appear on screen). When an application contains instrumentation code, it can be managed using a management tool such as the **Java Management Extensions** (**JMX**). The JMX technology provides developers with a flexible means to instrument Java technology-based applications (Java applications), create smart agents, implement distributed management middleware and managers, and smoothly integrate these solutions into existing management and monitoring systems. Instrumentation is necessary to review the performance of the application.

Using the JMX technology, a given resource is instrumented by one or more Java objects known as *Managed Beans*, or *MBeans*. These MBeans are registered in a core-managed object server, known as an *MBean server*. The MBean server acts as a management agent and can run on most devices that have been enabled for the Java programming language. A JMX agent consists of an MBean server, in which MBeans are registered, and a set of services for handling the MBeans. In this way, JMX agents directly control resources and make them available to remote management applications such as Zabbix. The way in which resources are instrumented is completely independent from the management infrastructure. Resources can therefore be rendered manageable regardless of how their management applications are implemented.

3.5 Monitoring development

This section describes the approach for the monitoring development and the two technology that has been implemented and tested to monitor the underlying component of the FIspace platform.

Two different ways have been considered to develop a monitoring service for the FIspace platform.

The first one of them is to use the Zapcat [33] library (JMX Zabbix Bridge), a third party library to communicate the monitoring information from one machine to a Zabbix server using few lines of code. Zapcat is free software released under GNU General Public License version 2.0 (GPLv2) (http://sourceforge.net/projects/zapcat/)

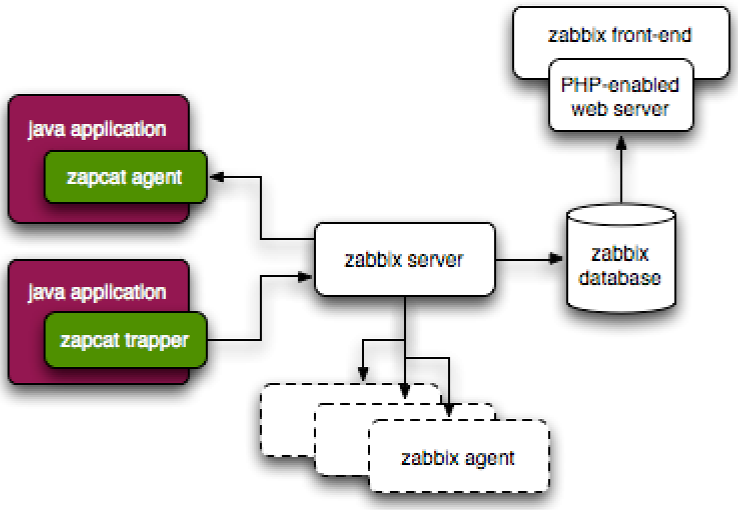


Figure 3: Zapcat architecture overview (Push / Pull model)

The data is pushed from the java application into Zabbix through a Zapcat trapper.

One of the main drawbacks of using this trapper is that it decentralises the configuration of your Zabbix monitoring system. No longer are all items configured in the Zabbix server, they are also configured in the trappers. (For more information see http://www.kjkoster.org/zapcat/Architecture.html). Figure 4 shows the setup that has been tested and that can be used.

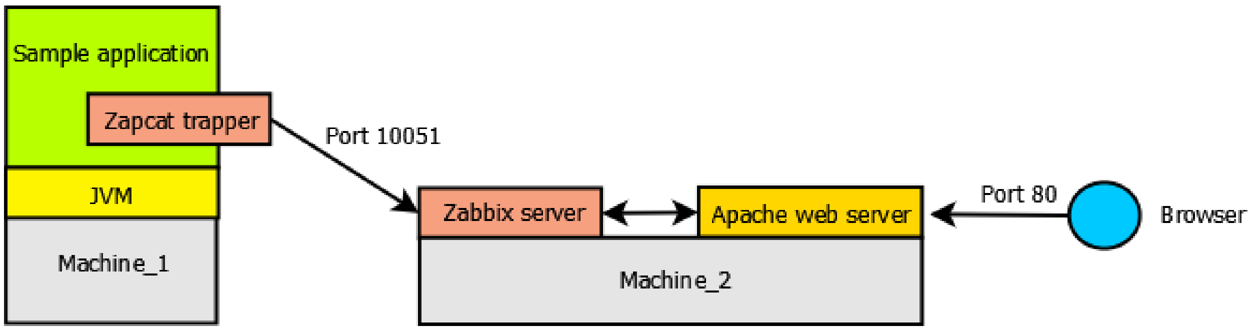


Figure 4: Zapcat trapper overview (Push model)

The second one is JMX, a Java technology that supplies tools for managing and monitoring applications, system objects, devices and service-oriented networks. Those resources are represented by objects called MBeans (for Managed Bean). In the API, classes can be dynamically loaded and instantiated. Managing and monitoring applications can be designed and developed using standard Java code. Figure 5 shows the JMX technology architecture overview.

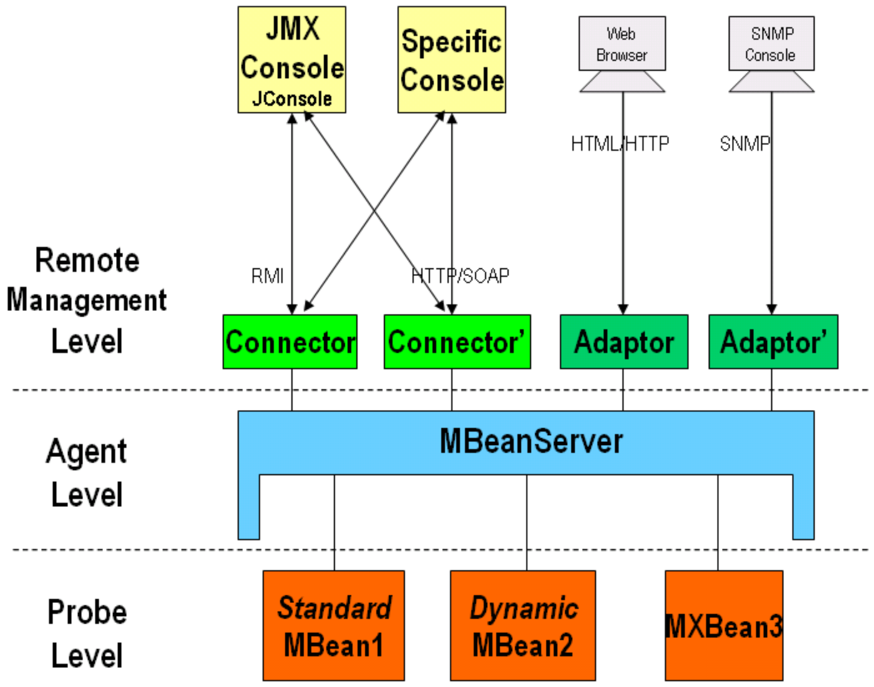


Figure 5: JMX technology overview

Finally for the integration of monitoring capabilities in the CSB component, this second option, based on the implementation of JMX MBean in the CSB has been implemented and tested successfully. Nevertheless Zapcat can be a valid option and lightweight solution depending on the other specific component to be monitored.

3.6 Monitoring Services

This section describes the different services and current component which are involved in the monitoring of the FIspace platform. The main component that allows and can provide fundamental information of monitoring is based on the Cloud Service Bus capabilities and design, as it is a common and centralized part where other component communicates between each other. Nevertheless other specific components monitoring services can be designed and described in the future.

3.6.1 CSB monitoring

CSB has three types of entities: **Coordinator**, **Node** and **Client**. Each CSB entities can be monitored, collecting a set of data utile to know, to identify possible problems or for statistical purposes among others.

3.6.1.1 Coordinator

Coordinator is a stand-alone process, typically run in a separate host (VM). It is responsible for overall centralized management of CSB in general, and the Queues in particular.

Log file monitoring

Coordinator's log file is the primary source of monitoring data related to run-time detection of Coordinator problems.

The file is called csb_coordinator.log, located **in CSB_HOME/coordinator folder**. Its format is governed by a log4j2 configuration.

This file is to be processed by a local Zabbix Log agent, which needs to be configured to look for log lines containing a word "**error**" (case insensitive).

JMX service

In addition, CSB Coordinator will include JMX Server and Beans that will allow to query it for the following information:

- Number of connected CSB Nodes
- List(*) of connected CSB Nodes (name, status, IP address and port)
- List of existing Queues (names and host Nodes)

In csb_coordinator.config, the following variable should be added:

- addActivateJMXMonitor=true

```
com.ibm.csb.coordinator:type=CoordMonitor

public interface CoordMonitorMBean {

      public int getNumberOfNodes();

      public String getNodeList();

      public String getQueueList();
}
```

(*) Here and later, a List is a variable-length string, comprised of a number of items, separated by a pre-defined separation symbol ('\n'). Each item has a pre-defined structure, potentially understood by the Zabbix server. A list of connected CSB Nodes looks like this:

"*Node: NodeA|Address: 37.131.248.119|Port: 20101|Suspended: false\nNode: NodeB|Address: 37.131.248.120|Port: 20101| Suspended: false \nNode: NodeC|Address: 37.131.248.125|Port: 20101| Suspended: false\n*"

This can be displayed as a single string in Zabbix server dashboard, or parsed and displays as 3 lines in a table.

You also need to modify the CSB coordinator launch script to add options to the java command line:

-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=9188

-Dcom.sun.management.jmxremote.authenticate=false

-Dcom.sun.management.jmxremote.ssl=false

-Djava.rmi.server.hostname=”your_host_name"

3.6.1.2 Node

CSB Node is a standalone process, typically run on the same with a component that uses it (B2B, SDI, etc). All Nodes are connected to the Coordinator, and to each other over a peer-to-peer network. A Node can host Queue(s) assigned to it by the coordinator.

Log file monitoring

Node's log file is the primary source of monitoring data related to run-time detection of Node problems.

The file is called csb_node.log. It is located in **CSB_HOME/coordinator folder** in B2B host. SDI component uses a configurable path to this file.

The log files format is governed by a log4j2 configuration.

This file is to be processed by a local Zabbix Log agent, which needs to be configured to look for log lines containing a word "**error**" (case insensitive).

JMX service

In addition, CSB Node will include JMX Server and Beans that will allow to query it for the following information:

- Number of hosted Queues
- List of hosted Queues (name, number of messages, etc)
- Number of connected clients
- List of connected clients (number of sent messages, number of received messages)

In csb_coordinator.config, the following variable should be added:

- addActivateJMXMonitor=true

```
com.ibm.csb.node:type=NodeMonitor

public interface NodeMonitorMBean {

	public int getNumberOfHostedQueues();

	public String getQueueStats();

	public int getNumberOfClients();

	public String getClientStats();

}
```

You also need to modify your CSB Node launch script to add options to the java command line:

-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=9188

-Dcom.sun.management.jmxremote.authenticate=false

-Dcom.sun.management.jmxremote.ssl=false

-Djava.rmi.server.hostname=”your_host_name"

3.6.1.3 Client

Client is a library, used by components (B2B, SDI, etc) to send and receive messages via CSB. Client connects to a local CSB Node.

Since it is not an independent process, its monitoring should be merged with the monitoring of the component itself. However, it could also be monitored separately

Log file monitoring

Log: If not merged with component's logging, it will be written to a file csb_client.log in a local runtime directory, governed by a log4j2 configuration.

This file is to be processed by a local Zabbix Log agent, which needs to be configured to look for log lines containing a word "**error**" (case insensitive).

JMX service

In addition, CSB client includes JMX Server and Beans that will allow to query it for the following information:

- Number of received messages (all queues)
- Number of sent messages (all queues)

```
com.ibm.csb.client.impl:type=ClientMonitor


public interface ClientMonitorMBean {


	public int getNumberOfReceivedMessages();
	public int getNumberOfSentMessages();
}
```

You also need to modify your component launch script to add options to the java command line:

-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=9187

-Dcom.sun.management.jmxremote.authenticate=false

-Dcom.sun.management.jmxremote.ssl=false

-Djava.rmi.server.hostname=”your_host_name"

The port 9187 is different from Node JMX port of 9188 - if there are two or more processes with JMX in the same host, the second will use 9187, the third 9186 - etc.

3.7 Monitoring tools installation and deployment

This section describes the main steps and configuration information needed to install and deploy the monitoring tools based on the Zabbix agent (client side) and Zabbix Server.

3.7.1 Main Zabbix features

Zabbix is software that monitors numerous parameters of a network and the health and integrity of servers. Zabbix uses a flexible notification mechanism that allows users to configure e-mail based alerts for virtually any event. This allows a fast reaction to server problems. Zabbix offers excellent reporting and data visualisation features based on the stored data. All Zabbix reports and statistics, as well as configuration parameters, are accessed through a web-based frontend. A web-based frontend ensures that the status of the network and the health of servers can be assessed from any location.

3.7.2 Setting up Zabbix Server, Agent and Java gateway

this section aims to describe the setting up, the installation and configuration, in a virtual machine (VM) where a Debian OS “flavor” (Debian squeeze - https://wiki.debian.org/DebianSqueeze/) was installed but all the step should be valid for other Linux distribution based on Debian as for example Ubuntu.

More instructions can be found at https://www.zabbix.com/documentation/2.0/manual/installation/install_from_packages

3.7.2.1 Setting up Zabbix Server

The following steps are necessary to install the Zabbix server and the typical commands are:

```
sudo wget http://repo.zabbix.com/zabbix/2.0/debian/pool/main/z/zabbix-release/zabbix-release_2.0-1squeeze_all.deb

sudo dpkg -i zabbix-release_2.0-1squeeze_all.deb

sudo apt-get update
```

Or for other version of Zabbix and with Ubuntu precise 12.04 LTS

```
sudo wget http://repo.zabbix.com/zabbix/versionXXX/

sudo dpkg -i zabbix-release_2.2-1+precise_all.deb

sudo apt-get update
```

To setup mysql and php front end:

```
sudo apt-get install zabbix-server-mysql zabbix-frontend-php
```

The above step involves setting up the mysql user for zabbix and a wizard will start, when the packages installation finish.

Restart zabbix-server: sudo /etc/init.d/zabbix_server restart (you need to restart once you modify the zabbix_server.conf file for the changes to setup the gateway.)

3.7.2.2 Setting up Zabbix Agent

On the other side, an agent is used to send the common metrics from a machine to a Zabbix server.

The following step is necessary to install the agent in every machine:

sudo apt-get install zabbix-agent

The Zabbix agent configuration file should be located to "*/usr/share/zabbix-agent/zabbix_agentd.conf*" or to "*/etc/zabbix/zabbix_agentd.conf*"

To configure an agent, the parameters 'Server' and 'ServerActive' have to be modified in the agent configuration file to add the Zabbix server´s IP. Afterwards, the agent has to be restarted.

3.7.2.3 Setting up Zabbix Java gateway

Concerning the setting up of the zabbix-java-gateway more information can be found at:

- https://www.zabbix.com/documentation/2.0/manual/concepts/java
- https://www.zabbix.com/documentation/2.0/manual/config/items/itemtypes/jmx_monitoring

The following steps are necessary to install the Zabbix java gateway and the typical commands is:

sudo apt-get install zabbix-java-gateway

3.7.2.4 Setting up environment

The zabbix server needs to be setup to listen to the gateway.

In the corresponding zabbix server configuration file (e.g. "*/etc/zabbix/zabbix_server.conf*"), uncomment and set the IP of the gateway, leave the default port as 10052 and very important, uncomment the StartJavaPollers and set it to 5.

Restart the zabbix server. No need to restart the gateway. Start the JMX agent (MBean server on the desired port, firewalls can be a problem and this has to be taken into consideration during the setup. In our case everything was on the same server, no changes to the java gateway were needed. The zabbix server config file was modified to include the IP of the machine where the java gateway is deployed).

In the front end for zabbix (zabbix server management and dashboard page), setup the host, the JMX interface and the item to be monitored. Useful instructions can be found in the example at: http://www.sysads.co.uk/2014/02/install-configure-jmx-zabbix-java-gateway/

Start the JMX agent as below:

```
java -Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=<port on which the service will be available>
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=<fully qualified hostname>
-Djava.net.preferIPv4Stack=true
-cp <path to the service class>
```

Very useful documentation on setting up composite data objects with MXBean and the difference between MBean and MXBean is available at:

http://www.oracle.com/technetwork/java/javase/tech/best-practices-jsp-136021.html

3.7.2.5 Instructions on Monitoring log files using Zabbix

More documentation can be found at: https://www.zabbix.com/documentation/2.0/manual/config/items/itemtypes/log_items

Setup the "item/log" file to be monitored in the Zabbix agent, while **specifying the name of the file and the regexp to be checked**, make sure that there is no space be-tween the "," and the string to be checked.

The **zabbix_agentd.conf** must be modified to set up the "hostname" as the Hostname set up in the front end.

Don't forget to restart both the agent and the server, and check the agent's log file (generally at "/var/log") to see if the active agent has started.

3.7.3 Using Zabbix

Information concerning everything you need to know about Zabbix (the zabbix con-cepts, installation, configuration, web monitoring,…) can be found at: https://www.zabbix.com/documentation/2.4/

The following screenshots are related to the dashboard web application as a result of our tests in a context of the preliminary integration environment .and in the FIspace Front-End VM.

The home page (e.g. http://myzabbixserver/zabbix) which is the login page:

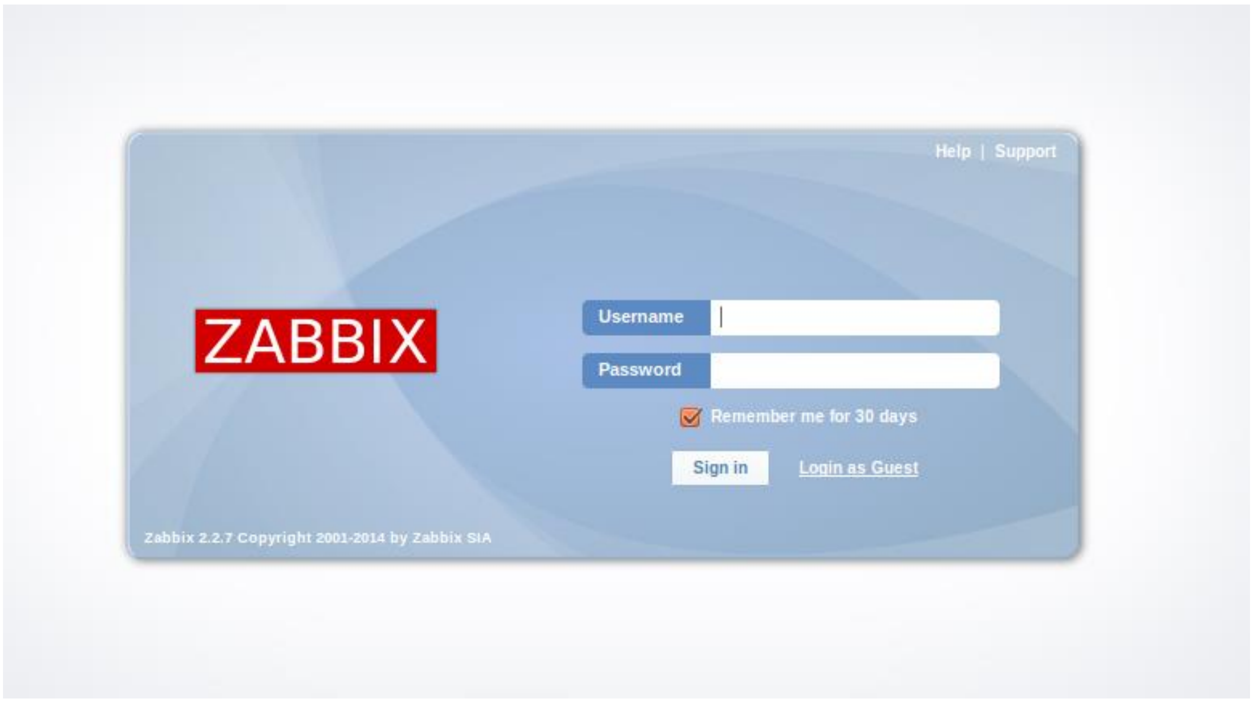
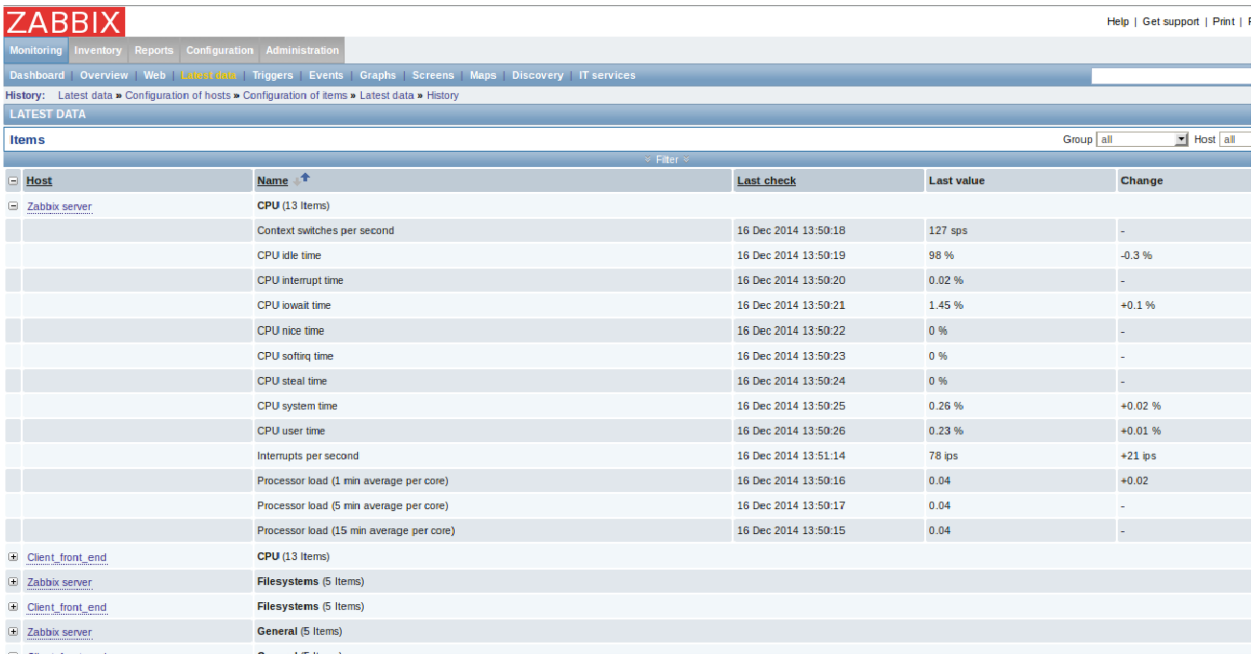


Figure 6: Zabbix web application – Login page

A list showing several available metrics in the dashboard web application:


ZABBIX
Help | Get support | Print |
Monitoring | Inventory | Reports | Configuration | Administration
Dashboard | Overview | Web | Latest data | Triggers | Events | Graphs | Screens | Maps | Discovery | IT services

History: Latest data » Configuration of hosts » Configuration of items » Latest data » History

LATEST DATA

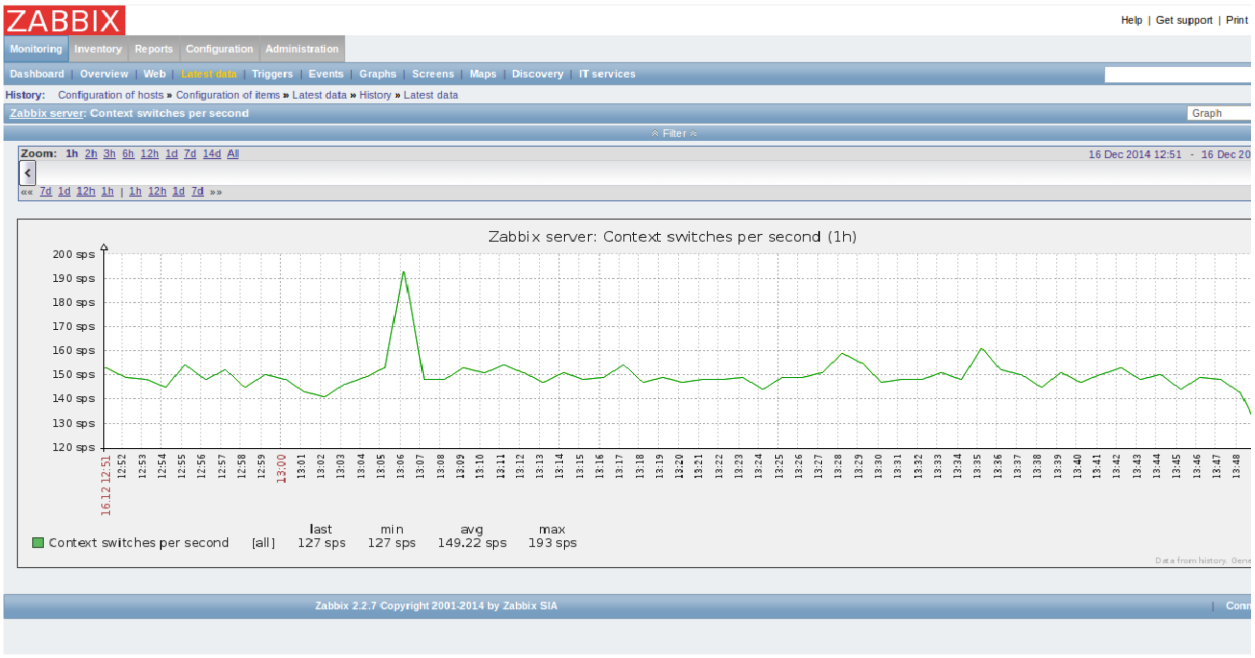
Items — Group all — Host all

Filter

Host	Name	Last check	Last value	Change
Zabbix server	**CPU** (13 Items)			
	Context switches per second	16 Dec 2014 13:50:18	127 sps	-
	CPU idle time	16 Dec 2014 13:50:19	98 %	-0.3 %
	CPU interrupt time	16 Dec 2014 13:50:20	0.02 %	-
	CPU iowait time	16 Dec 2014 13:50:21	1.45 %	+0.1 %
	CPU nice time	16 Dec 2014 13:50:22	0 %	-
	CPU softirq time	16 Dec 2014 13:50:23	0 %	-
	CPU steal time	16 Dec 2014 13:50:24	0 %	-
	CPU system time	16 Dec 2014 13:50:25	0.26 %	+0.02 %
	CPU user time	16 Dec 2014 13:50:26	0.23 %	+0.01 %
	Interrupts per second	16 Dec 2014 13:51:14	78 ips	+21 ips
	Processor load (1 min average per core)	16 Dec 2014 13:50:16	0.04	+0.02
	Processor load (5 min average per core)	16 Dec 2014 13:50:17	0.04	-
	Processor load (15 min average per core)	16 Dec 2014 13:50:15	0.04	-
Client_front_end	**CPU** (13 Items)			
Zabbix server	**Filesystems** (5 Items)			
Client_front_end	**Filesystems** (5 Items)			
Zabbix server	**General** (5 Items)			

Figure 7: Zabbix web application – Dashboard page example

The following graphic corresponds to a specific metric.


ZABBIX
Help | Get support | Print
Monitoring | Inventory | Reports | Configuration | Administration
Dashboard | Overview | Web | Latest data | Triggers | Events | Graphs | Screens | Maps | Discovery | IT services
Graph
Filter
Zoom: 1h 2h 3h 6h 12h 1d 7d 14d All
«« 7d 1d 12h 1h | 1h 12h 1d 7d »»
16 Dec 2014 12:51 - 16 Dec 20
200 sps
190 sps
180 sps
170 sps
160 sps
150 sps
140 sps
130 sps
120 sps
16.12 12:51
13:00
13:48
Data from history. Gene
Conn

History: Configuration of hosts » Configuration of items » Latest data » History » Latest data

Zabbix server: Context switches per second

Zabbix server: Context switches per second (1h)

		last	min	avg	max
🟩 Context switches per second	[all]	127 sps	127 sps	149.22 sps	193 sps

Zabbix 2.2.7 Copyright 2001-2014 by Zabbix SIA

Figure 8: Zabbix web application – Graphic page example

The screenshots above was captured during our test in an instance of the FIspace platform deployed in our preliminary integration environment.

More external resources and example of screenshots can be found at: http://www.zabbix.com/screenshots.php

4 Glossary

The glossary provides the coherent terminological framework used in this document.

4.1 Terms and definitions

This section provides definitions of any terms that may be needed in order for the reader to understand the terminology used in the document. The author should define any definition/acronym or technical term used in the document that may be unfamiliar to the reader, and it is best to err on the side of too many rather than too few definitions. This also allows the author to frame a word within a specific context, which provides the reader with a common understanding of the author's definition.

Access control

Authorisation (or denegation) for performing a certain action (based on privileges management). The access control is carried out once the Identification and Authentication procedures have been performed.

Accounting

Process of gathering information about the usage of resources by subjects.

Acceptance and trust

Acceptability indicates the degree of approval of a technology by the users. It depends on whether the technology can satisfy the needs and expectations of its users and potential stakeholders. Within the framework of introducing new technologies, acceptability relates to social and individual aspects as well.

Application

Use of capabilities, including hardware, software and data, provided by an information system specific to the satisfaction of a set of user requirements in a given application domain.

Application Domain

Integrated set of problems, terms, information and tasks of a specific thematic domain that an application (e.g. an information system or a set of information systems) has to cope with.

Application Schema [ISO/FDIS 19109:2003]

Conceptual schema for data required by one or more applications.

Architecture (of a system) [ISO/IEC 10746-2:1996]

Set of rules to define the structure of a system and the interrelationships between its parts.

Architecture (of a system) [ISO/IEC 10746-2:1996]

Set of rules to define the structure of a system and the interrelationships between its parts.

Authentication

Process of verifying the identity of a certain subject. In other words authentication indicates whether a subject is who/what it seems to be.

Generally speaking, this proof can depend on a secret that can be, e.g. what somebody has (key, smart card, …), what somebody knows (password, …), what somebody is (biometrical data, …)

Authorisation

Process of determining whether a subject is allowed to have the specified types of access to a particular resource. This is done by evaluating applicable access control information contained in a so called authorisation context. Usually, authorisation is carried out after the identification and authentication. Once a subject is identified and authenticated, it may be authorized (or not) to perform different types of access.

Availability

Availability refers to the degree to which a system, subsystem, or equipment is in a specified operable and committable state at the start of a mission, when the mission is called for at an unknown, i.e., a random time. So, availability is the proportion of time that a system is in operating condition.

Capability

Capabilities are a set of functionalities, through a combination of software and hardware, used to provide services and data. They can reside in a system or for example in a terminal itself as embedded capabilities or they can be available through the network services and infrastructure and others communication technologies as external capabilities.

Catalogue [derived from http://www.opengeospatial.org/resources/?page=glossary]

Collection of entries, each of which describes and points to a feature collection. Catalogues include indexed listings of feature collections, their contents, their coverages, and of meta-information. A catalogue registers the existence, location, and description of feature collections held by an Information Community. Catalogues provide the capability to add and delete entries. A minimum Catalogue will include the name for the feature collection and the locational handle that specifies where these data may be found. Each catalogue is unique to its Information Community.

Certificate Authority

A Trusted Third Party, responsible for ensuring the binding between the public keys and the personal data of their respective owners.

Component

Hardware component (device) or Software Component.

Conceptual model [ISO/FDIS 19109:2003(E); ISO 19101]

Model that defines concepts of a universe of discourse.

Conceptual schema [ISO/FDIS 19109:2003(E); ISO 19101]

Formal description of a conceptual model.

Coverage [ISO 19123]

Function from a spatial, temporal or spatiotemporal domain to an attribute range. A coverage associates a position within its domain to a record of values of defined data types. Thus, a coverage is a feature with multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type.

Data acquisition

Methods of data acquisition include methods to collect background data, digitally acquire data from sensors, and subjective data (such as data acquired from questionnaires). In addition, data in the form of manually or automatically transcribed data and reductions of collected data is also considered sensor acquired data (but with a manual sensor – the analyst).

Description Logics

Family of logic based knowledge representation languages that are a decidable subset of first order logic with well-defined semantics and inferencing (problem decision procedures). In Description Logics, a distinction is made between the terminological knowledge and the assertional knowledge. This distinction is useful for knowledge base modelling and engineering: for modelling it is just natural to distinguish between concepts and individuals; for engineering it helps by separating key inference problems.

Digital Certificate

A kind of digital document that contains structured information about the identity of its owner along with her/his public key, signed all together with a Certificate Authority's private key.

Digital Signature

The encrypted form of a message with the private key of the owner, indicating in a secure way the creator of the message, as well as the identity of a signed data.

Encryption

The act of modifying the contents of a message in an algorithmic and secure way, so that it can not be observed or altered in while in transit.

End-User

All users that are involved in an application domain and that use the applications, the services built by the system users according to the system and service Architecture.

Feature [derived from ISO 19101]

Abstraction of a real world phenomenon [ISO 19101] perceived in the context of an Application. In this general sense, a feature corresponds to an "object" in analysis and design models.

Framework [http://www.opengeospatial.org/resources/?page=glossary]

An information architecture that comprises, in terms of software design, a reusable software template, or skeleton, from which key enabling and supporting services can be selected, configured and integrated with application code.

Generic

A service is generic, if it is independent of the application domain. A service infrastructure is generic, if it is independent of the application domain and if it can adapt to different organisational structures at different sites, without programming (ideally).

Identification

The identification process allows relating a person/device with the service environment. The “electronic identity” is something like a credential or a “business card”, suitable to be verified throughout the authentication process.

Implementation [http://www.opengeospatial.org/resources/?page=glossary]

Software package that conforms to a standard or specification. A specific instance of a more generally defined system.

Info-structure Service

Service that is required to operate a system oriented service in the sense that it plays an indispensable role in the operation of an architecture or system oriented service.

Interface [ISO 19119:2005; http://www.opengis.org/docs/02-112.pdf]

Named set of operations that characterize the behaviour of an entity.

The aggregation of operations in an interface, and the definition of interface, shall be for the purpose of software reusability. The specification of an interface shall include a static portion that includes definition of the operations. The specification of an interface shall include a dynamic portion that includes any restrictions on the order of invoking the operations.

Interoperability [ISO 19119:2005 or OGC; http://www.opengeospatial.org/resources/?page=glossary]

Capability to communicate, execute programs, or transfer data among various functional units in a manner that require the user to have little or no knowledge of the unique characteristics of those units [ISO 2382-1]. (http://www.opengeospatial.org/ogc/glossary/i)

Loose coupling [W3C; http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#loosecoupling]

Coupling is the dependency between interacting systems. This dependency can be decomposed into real dependency and artificial dependency: Real dependency is the set of features or services that a system consumes from other systems. The real dependency always exists and cannot be reduced. Artificial dependency is the set of factors that a system has to comply with in order to consume the features or services provided by other systems. Typical artificial dependency factors are language dependency, platform dependency, API dependency, etc. Artificial dependency always exists, but it or its cost can be reduced. Loose coupling describes the configuration in which artificial dependency has been reduced to the minimum.

Middleware [http://www.opengeospatial.org/resources/?page=glossary]

Software in a distributed computing environment that mediates between clients and servers.

Open Architecture [based on (Powell 1991)] [34]

Architecture whose specifications are published and made freely available to interested vendors and users with a view of widespread adoption of the architecture. An open architecture makes use of existing standards where appropriate and possible and otherwise contributes to the evolution of relevant new standards.

Operation [ISO 19119:2005; http://www.opengis.org/docs/02-112.pdf]

Specification of a transformation or query that an object may be called to execute. An operation has a name and a list of parameters.

Performance indicators definition (PI)

PIs are quantitative or qualitative measurements, agreed on beforehand, expressed as a percentage, index, rate or other value, which is monitored at regular or irregular intervals and can be compared with one or more criteria.

Platform (Service)

Set of infrastructural means and rules that describe how to specify service interfaces and related information and how to invoke services in a distributed system.

Reference Model [ISO Archiving Standards; http://ssdoo.gsfc.nasa.gov/nost/isoas/us04/defn.html]

A reference model is a framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts and may be used as a basis for education and explaining standards to a non-specialist.

Reliability

Reliability is the ability of a system or component to perform its required functions in routine circumstances, as well as hostile or unexpected circumstances, under stated conditions for a specified period of time.

Resource

Functions (possibly provided through services) or data objects.

Service [ISO 19119:2005; ISO/IEC TR 14252; http://www.opengis.org/docs/02-112.pdf]

Distinct part of the functionality that is provided by an entity through interfaces.

REST

Representational state transfer (REST) is an abstraction of the architecture of the World Wide Web; more precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

Service [ISO 19119:2005; ISO/IEC TR 14252; http://www.opengis.org/docs/02-112.pdf]

Distinct part of the functionality that is provided by an entity through interfaces.

Session

Temporary association between a subject and a principal as a result of an authentication process initiated by the subject. Information about a session is stored in authentication session information.

SOAP

Simple Object Access protocol is a protocol specification for exchanging structured information in the implementation of web services in computer networks. It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

Software Component [derived from component definition of http://www.opengeospatial.org/resources/?page=glossary]

Software program unit that performs one or more functions and that communicates and interoperates with other components through common interfaces.

Source System

Container of unstructured, semi-structured or structured data and/or a provider of functions in terms of services. The source systems are of very heterogeneous nature and contain information in a variety of types and formats.

Support Service

Service that facilitates the operation of an architecture or system oriented service, e.g. providing an added value by combining the usage of Info-Structure Services.

System [ISO/IEC 10746-2:1996]

Something of interest as a whole or as comprised of parts. Therefore a system may be referred to as an entity. A component of a system may itself be a system, in which case it may be called a sub-system.

Note: For modelling purposes, the concept of system is understood in its general, system theoretic sense. The term "system" can refer to an information processing system but can also be applied more generally.

System User

Provider of services that are used for an application domain as well as IT architects, system developers, integrators and administrators that conceive, develop, deploy and run applications for an application domain.

Terminal

Terminals are a mobile device that is capable of running mobile services and/or mobile applications.

Use case

A common definition of use cases is the one described by Jacobson (Jacobson et al (1995) [35]): "*When a user uses the system, she or he will perform a behaviourally related sequence of transactions in a dialogue with the system. We call such a special sequence a use case*". In Other words, a use case is a textual presentation or a story about the usage of the system told from an end user's perspective.

The use cases provide some tools for people, with different skills (e.g. software developers and non-technology oriented people), to communicate with each other. The use cases are general descriptions of needs or situations that often are related to basic scenarios and that are independent of the technologies and implementations of the underlying system.

User

Human acting in the role of a system user or end user of the service and system.

WADL

The Web Application Description Language is a machine-readable XML description of HTTP-based web applications (typically REST web services) WADL models the resources provided by a service and the relationships between them. WADL is intended to simplify the reuse of web services that are based on the existing HTTP architecture of the Web. It is platform and language independent and aims to promote reuse of applications beyond the basic use in a web browser.

Web Service

Self-contained, self-describing, modular service that can be published, located, and invoked across the Web. A Web service performs functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.

W3C Web Service [W3C, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice]

Software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

5 References

The following references are used as background documents for the preparation of this document. References are categorized standards (i.e. standards and specifications from the consortium working groups or alliances and specifications or drafts standardization bodies) and other documents, publications and technical or scientific books.

[1]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.1 *“FIspace Design and Release Plan”*, 2014.
[2]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.2 *“FIspace Technical Architecture and Specification”*, 2014
[3]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.3 *“FIspace Integrated Release V1”*, 2014
[4]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.4 *“FIspace Development Progress Report and V1 Updates”, 2014*
[5]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.5 *“FIspace Integrated Release V2”*, 2014.
[6]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.6 *“FIspace Development Progress Report and V2 Updates”, 2014*
[7]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.5 Annex *“FIspace Front-End User Guide”*
[8]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.5 Annex *“FIspace SDK User and Developer Guide”*
[9]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.7 *“FIspace Integrated Release V3”*, 2014.
[10]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.7 Annex *“FIspace Front-End User Guide”*
[11]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.7 Annex *“FIspace SDK User and Developer Guide”*

[12]	FIspace Business collaboration web site. http://www.fispace.eu/
[13]	FIspace Developer Documentation web site. http://dev.fispace.eu/doc/wiki/Home
[14]	FIspace Deliverables web site. http://www.fispace.eu/deliverable.html
[15]	FIspace Tutorial web site. http://www.fispace.eu/tutorials.html
[16]	FIspace Front-End Users Information web site. http://dev.fispace.eu/doc/wiki/gui
[17]	FIspace Front-End User Guide web site. http://dev.fispace.eu/doc/wiki/gui/gui-guide
[18]	FIspace App Developer Intro web site. http://dev.fispace.eu/doc/wiki/App%20Developer%20Intro
[19]	FIspace SDK Guide web site. http://dev.fispace.eu/doc/wiki/sdk
[20]	FIspace apps for newbies web site. https://bitbucket.org/fispace/apps/wiki/FIspace%20apps%20for%20newbies
[21]	FIWARE web site. http://www.fi-ppp.eu/projects/fi-ware/
[22]	FIWARE Catalogue of the Generic Enablers (GEs). http://catalogue.fi-ware.org/
[23]	FIWARE community web site. http://www.fi-ware.org/community/
[24]	FIWARE - Catalogue - Application Mashup – Wirecloud web site. http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud
[25]	FIWARE - Catalogue - Store – Wstore web site. http://catalogue.fi-ware.org/enablers/store-wstore
[26]	Eclipse web site. https://www.eclipse.org/ , https://www.eclipse.org/downloads/
[27]	Maven web site. http://maven.apache.org/ , http://maven.apache.org/download.cgi
[28]	RabbitMQ web site. http://www.rabbitmq.com/
[29]	Express web site. http://expressjs.com/
[30]	ACSI EU-project web site. http://www.acsi-project.eu/
[31]	Proton - CEP recorded webinar and tutorial. http://edu.fi-ware.eu/course/view.php?id=58 Proton - CEP user and programmers guide. http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/CEP_-

	_User_and_Programmer_Guide
[32]	Zabbix Open source software. http://www.zabbix.com/ , http://www.zabbix.com/documentation.php
[33]	Zapcat for Java developers and for system administrators who wish to monitor Java applications from Zabbix. http://www.kjkoster.org/zapcat/Zapcat_JMX_Zabbix_Bridge.html
[34]	Powell, D. (Ed.) (1991). Delta-4: A Generic Architecture for Dependable Distributed Computing. Re-search Reports ESPRIT. Project 818/2252 Delta-4 Vol.1. ISBN 3-540-54985-4 Springer-Verlag 1991.
[35]	Jacobson, I., Bylund, S., Jonsson, P., and Ehneboom, S. (1995), “Mod-eling with Use Cases: Using contracts and use cases to build plugable architectures”. Journal of Object Oriented Programming, Vol. 8, No. 2, pp. 18-24.

FIspace

Business Collaboration