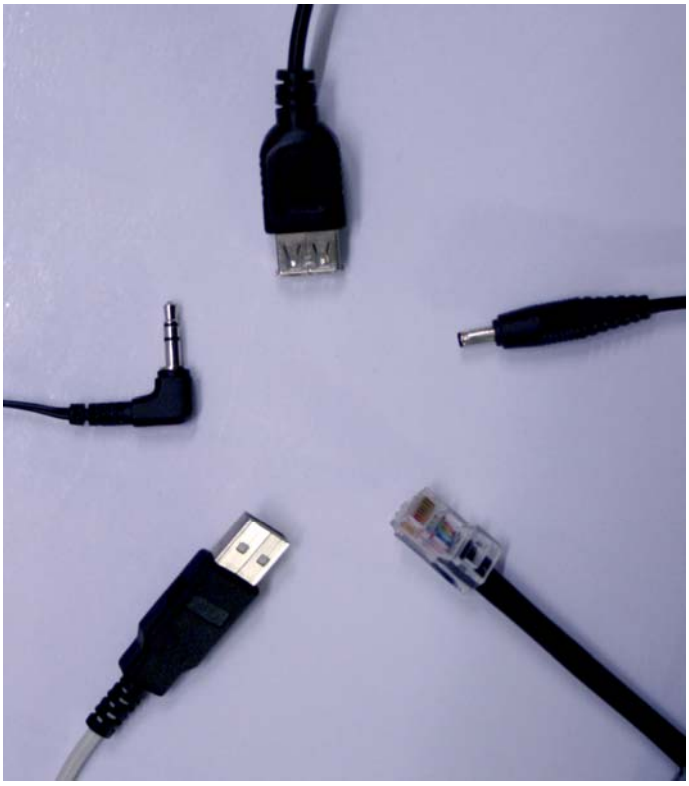
UrbanFlood



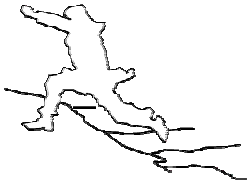
Specification of Plug-in Technology for EWS Components

Work Package 6 – D6.3

version 0.3, 2010-11-22

October 2010

URBAN FLOOD



A project funded under the EU
Seventh Framework Programme
Theme ICT-2009.6.4a
ICT for Environmental Services and
Climate Change Adaption

Grant agreement no. 248767
Project start: December 1, 2009
Project finish: November 30, 2012

Coordinator

Urban Flood Project Office at TNO-ICT
Prof dr Robert J. Meijer

Eemsgolaan 3
PO Box 1416
9701 BK Groningen
The Netherlands

E : robert.meijer@tno.nl
T: +31 50-5857759
W: www.urbanflood.eu

DOCUMENT INFORMATION

Title	Plug-in Technology for EWS Components
Lead Author	Artem Ozhigin
Contributors	
Distribution	Public
Document Reference	UFD6.3.3SIE

DOCUMENT HISTORY

Date	Revision	Prepared by	Organisation	Approved by	Notes
1 Oct. 2010	0.1	Artem Ozhigin	Siemens		First draft version
18 Oct 2010	0.2	Jan Sipke van der Veen	TNO		Added some comments
22 Nov 2010	0.3	Artem Ozhigin	Siemens	Rob Meijer	Review feedback from the first technical review round accounted

ACKNOWLEDGEMENT

The work described in this publication was supported by the European Community's Seventh Framework Programme through the grant to the budget of the Project **UrbanFlood**, Grant Agreement no. 248767.

DISCLAIMER

This document reflects only the authors' views and not those of the European Community. This work may rely on data from sources external to the UrbanFlood project Consortium. Members of the Consortium do not accept liability for loss or damage suffered by any third party as a result of errors or inaccuracies in such data. The information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and neither the European Community nor any member of the UrbanFlood Consortium is liable for any use that may be made of the information.

© URBANFLOOD CONSORTIUM

TABLE OF CONTENT

SUMMARY

This document specifies the design of UrbanFlood EWS hosting platform and plug-in technology for UrbanFlood EWS components. Plug-in technology defines all of the necessary properties for component to be able to integrate into a EWS running on the UrbanFlood platform.

Acronyms and Abbreviations

C&D	Communication and Dissemination
CIS	Common Information Space
CYF, CY	Cyfronet AGH, Cracow, Poland – UrbanFlood partner
Defra	Department for Environment, Food and Rural Affairs, UK
DoW	Description of Work, annex to the UrbanFlood Grant Agreement
EC	European Community
EWS	Early Warning System
FLOODsite	EU funded project: Integrated Flood Risk Analysis and Management Methodologies. 6th Framework Programme
FRM	Flood Risk Management
FTP	File Transfer Protocol
GPL	General Public License
GUI	Graphical User Interface
HRW	HR Wallingford
ICT	Information and Communication Technology
IJkdijk	IJkdijk Foundation, dike testing site in Groningen, the Netherlands
IWS	International Online Early Warning Workshop
JMS	Java Message Service
LiveDijk	Test location for sensor technologies at Eemshaven, Groningen, the Netherlands
M&E	Monitoring and Evaluation
NEMO	Science museum in Amsterdam, the Netherlands
NFS	Network File System
OSS	Open Source Software
PU	Public (report)
RE	Restricted (report)
RTD	Research (in DoW)
SIE	Siemens, Germany. OOO Siemens in Russia is an UrbanFlood project partner
SMB	Server Message Block
SME	Small and Medium sized Enterprise

STOWA, STO	Dutch acronym for the Foundation for Applied Water Research, Utrecht, The Netherlands – UrbanFlood Partner
TNO	TNO Dutch organisation for Applied Research, The Netherlands – UrbanFlood lead partner
UK	United Kingdom
US, USA	United States of America
UvA	University of Amsterdam
VM	Virtual Machine
WP	Work Package (there are 7 work packages in UrbanFlood)
WS	Web Service
XML	Extensible Markup Language

1 Introduction

1.1 UrbanFlood

UrbanFlood is a project investigating the use of sensors within flood embankments to support an online early warning system, real time emergency management and routine asset management. It is a project under the EU 7th framework Programme which started in December 2009 and will run for 3 years. Partners of UrbanFlood include TNO Information and Communication Technology, the University of Amsterdam and STOWA (Dutch acronym for the Foundation for Applied Water Research) from the Netherlands; HR Wallingford in the UK, ACC Cyfronet AGH in Poland and OOO Siemens in Russia

1.2 Overview of Service Platform and Plug-in Technology

The goal of WP6 is the development of a distributed service platform for UrbanFlood Early Warning Systems (EWS) which will allow for easy adoption, development, deployment and management of many different kinds of applications and provide their integration with the Common Information Space (CIS, see Deliverable 5.1) through distinctly defined interfaces. The platform shall provide to applications complete independency on physical location and hardware. From the other side the platform itself shall be able to deploy, manage and execute applications disregarding their implementation details (programming languages, to some extents architecture, and even operating system to which applications are targeted). Of course in order to fully support all intrinsic platform features some pre-developed applications might require specific modifications but there shall be a way to keep these as small as possible.

Plug-in technology as a crucial part of the service platform is intended to facilitate the fulfilment of above mentioned requirements. In general plug-in technology defines the following matters:

- wrapping of applications in uniform containers, forming isolated and manageable components;
- interfaces for component interaction with CIS and users within an EWS.

Both of these matters are described in more details further in this document.

1.3 Structure of this Document

Chapter 2 Describes the technology chosen for component wrapping – enclosing application into a separate entity with appropriate environment.

Chapter 3 Dedicated to the description of interfaces through which all component's interaction with the platform is held.

Chapter 4 Describes the stages of component lifecycle

2 Component Wrapping

2.1 Virtualization Technology

It is decided that for the component wrapping virtualisation will be used. This means that every application will be installed within its own virtual machine. Virtualization completely hides the physical characteristics of the underlying computer platform showing an abstract computing platform instead. Such approach will ensure a comfortable environment (including desired operating system) for every kind of application (either newly-developed or pre-developed) and strict separation preventing unintended components influence on each other. Besides that, virtualized components are quite easy to manage (deploy, relocate, load balance etc).

Some exclusion might be made for very hardware-intensive or hardware-dependent components as 4-D visualisations and smart user interfaces (e.g. using multi-touch panels). Those components might be deployed on a dedicated physical device without wrapping into a virtual machine.

Several alternatives were considered as candidates for the UrbanFlood virtualisation platform:

- Microsoft Hyper-V
- KVM
- Sun VirtualBox
- Microsoft Virtual PC
- VMware
- Xen
- Citrix XenServer

Finally Citrix XenServer was chosen due to the following features:

- Free of charge;
- Lightweight highly-efficient hypervisor implying small performance overhead;
- Availability of paravirtualized drivers for a wide range of guest operating systems;
- Convenient API to observe and manage server installation and guest virtual machines;

- Significant experience of consortium members with this particular virtualisation software.

Citrix XenServer is based on open source Xen 64-bit hypervisor and supports paravirtualization and hardware-assisted virtualization. Citrix also offers several additional commercial components for XenServer which are optional and are not considered to be used in UrbanFlood project.

2.2 Architecture of Hosting Platform

The UrbanFlood hosting platform shown in Figure 1 consists of a set of virtualization servers (XenServer hosts) or pools of virtualization servers (XenServer pools) on which components in the form of virtual machines and virtual machine templates are stored and executed. XenServer pool is a set of XenServer hosts tied together into a single entity from the management point of view. Further in this document no distinction will be made between pools of XenServer installation and single XenServer hosts not bound to any pool. Both pools and hosts will be referred to as “sites”.

XenServer VM templates are images of virtual machines that contain all the various configuration settings to instantiate a specific VM. Any instance of a VM can be easily converted into a template for further use. Templates facilitate very fast provisioning and deployment of new VMs and thus are especially useful as a form of UrbanFlood component storage. Components which can be made generic and universal for any foreseen purpose can be easily converted into templates for being deployed and configured to fit the requirements of particular EWS.

Each site communicates with the IntercloudManager in order to report status information and receive management commands. Interaction between a site and IntercloudManager is carried out by a separate application (SiteManager) running on each site on a dedicated VM or on physical host, see Figure 1.

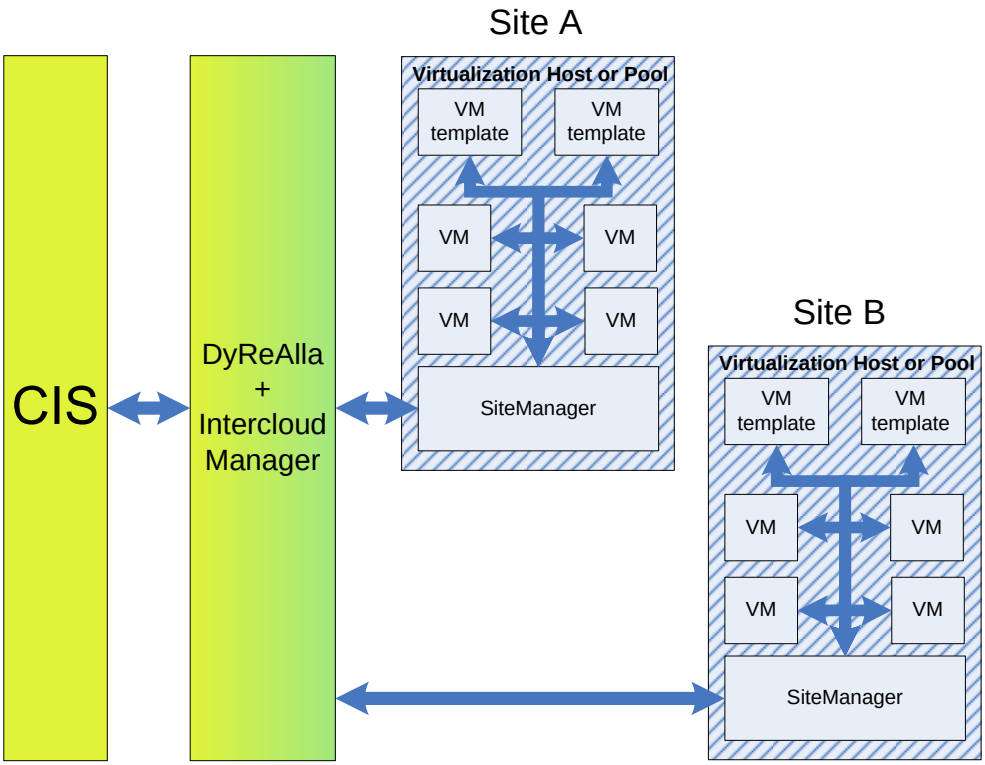


Figure 1 - UrbanFlood Hosting Platform

Status information reported by the SiteManager to the IntercloudManager includes the following:

- Site metrics:
  - Number of CPUs and their usage;
  - Amount of memory and its usage;
  - Total amount and current availability of disk space;
  - List of hosted VMs and VM templates
- VMs metrics and status
  - Current VM state (stopped, running, paused);
  - Number of virtual CPUs and their usage;
  - Amount of allocated memory and its usage;
  - Total amount and current availability of virtual disk space
  - IP address(es)

This information is used by the IntercloudManager and DyReAlla (Dynamic Resource Allocation) CIS component [5] for the purposes of resource management, load balancing and decision making on deployment of new components.

Management commands are aimed to control VMs stored on site and include the following:

- Create new VM from template;
- Start VM;
- Stop VM;
- Pause VM;
- Reboot VM;
- Migrate VM (to a different site);
- Change VM resources (number of CPUs, amount of memory);
- Delete VM.

The IntercloudManager provides to the CIS a single management entry point to complete a set of platform virtualization hosts. The CIS controls the instantiation of components and EWS-es sending respective commands to the IntercloudManager which distributes them to particular hosts. Thus the IntercloudManager hides from the CIS all details about actual amount and configuration of available virtualization sites making the CIS treat them as an abstract cloud of computational resources.

2.3 Communication Aspects

All communications between CIS, components and sites within the UrbanFlood platform are implemented using TCP/IP-based networking. Thus all sites and virtual machines shall be provided with an IP address either static or dynamic to be able to communicate. Since UrbanFlood platform implies that sites and components might be distributed over the globe and use Internet as a means for communications public IP addresses for every component are preferred as shown in Figure 2.

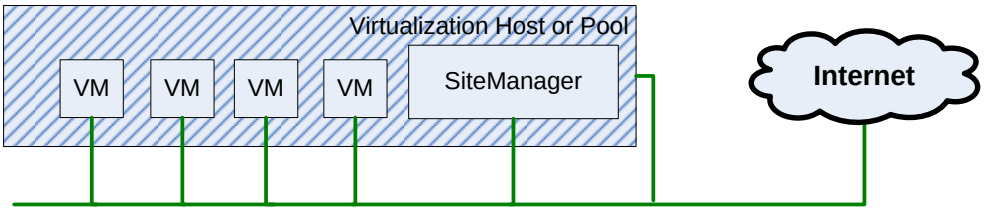


Figure 2 - Network configuration with a series of public IP addresses

In case when there are not enough public IP addresses in the location of a site it is allowed to establish an internal virtual network within the site with a virtual gateway to the external network as shown in Figure 3.

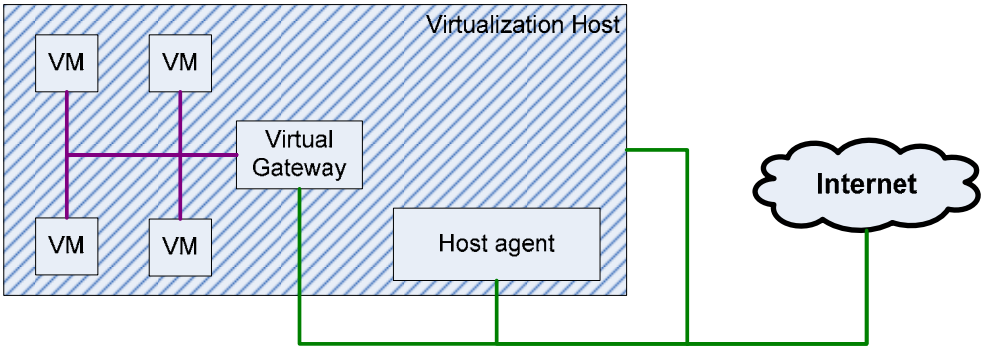


Figure 3 - Configuration with virtual network

This gateway might also act as a firewall, NAT and DHCP server. Such configuration however would aggravate access to components from external networks, thus connection establishment from the component side is preferred if possible and sometimes it is the only way to establish the connection. This will be discussed in more details in the section dedicated to component interfaces.

3 Component Interfaces

3.1 Interface Technologies

Java Message Service (JMS)

The Java Message Service (JMS) API is a messaging standard that allows application components based on the Java 2 Platform, Enterprise Edition (J2EE) to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous. [4]

A JMS application can use either the point-to-point (PTP) and the publish-and subscribe (Pub/Sub) style of messaging. An application can also combine both styles of messaging in one application. These two styles of messaging are often referred to as messaging domains. JMS provides these two messaging domains because they represent two common models for messaging. When using the JMS API, a developer can use interfaces and methods that support both models of messaging.

In the point-to-point model, a sender posts messages to a particular queue and a receiver reads messages from the queue. The sender knows the destination of the message and posts the message directly to the receiver's queue. It is characterized by the following:

- Only one consumer gets the message
- The producer does not have to be running at the time the consumer consumes the message, nor does the consumer need to be running at the time the message is sent
- Every message successfully processed is acknowledged by the consumer

The publish/subscribe model supports publishing messages to a particular message topic. Subscribers may register interest in receiving messages on a particular message topic. In this model, neither the publisher nor the subscriber know about each other. The following are characteristics of this model:

- Multiple consumers (or none) will receive the message
- There is a timing dependency between publishers and subscribers. The publisher has to create a subscription for clients to subscribe. The subscriber has to remain continuously active to receive messages, unless it has established a durable subscription. In that case, messages published while the subscriber is not connected will be redistributed whenever it reconnects.

In the UrbanFlood platform the CIS contains the implementation of a JMS provider and application components connect to the provider as publishers to make output data available

to other components and act as JMS subscribers for input data reception. Separation of data streams are to be implemented using JMS topic mechanisms and input message filtering on the subscriber side. JMS connection establishment from component towards the CIS provider will work without any problems in any network configuration- either physical or virtual.

Web Service

The W3C defines a "Web Service" as "a 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" [2]. Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. The purpose of a Web service is to provide some functionality on behalf of its owner – a person or organization, such as a business or an individual. The provider entity is the person or organization that provides an appropriate agent to implement a particular service. A requester entity is a person or organization that wishes to make use of a provider entity's Web service. It will use a requester agent to exchange messages with the provider entity's provider agent. In most cases, the requester agent is the one to initiate this message exchange [3].

In the UrbanFlood platform application components shall include a web service provider while the CIS shall implement a web service requester initiating data exchange. The web service of the virtual network configuration component will be not accessible from the outside.

FTP

File Transfer Protocol (FTP) is a standard network protocol used to copy a file from one host to another over a TCP/IP-based network, such as the Internet. FTP is built on client-server architecture and utilizes separate control and data connections between the client and the server. FTP is used with user-based password authentication or with anonymous user access.

FTP can be run in active or passive mode, which determines how the data connection is established. In active mode, the client sends the server the IP address and port number on which the client will listen, and the server initiates the TCP connection. In situations where the client is behind a firewall and unable to accept incoming TCP connections (virtual network configuration), passive mode may be used. In this mode the client sends a PASV command to the server and receives an IP address and port number in return. The client uses these to open the data connection to the server.

In the UrbanFlood platform it is allowable to include either FTP client or FTP server within the component while having the counterpart in the CIS.

Network Share

Network share is a device or piece of information on a computer that can be remotely accessed from another computer via a network. In client–server communications, a client process on one computer takes the initiative to start the communication, while a server process on the file server or print server remote computer passively waits for requests to start a communication session.

On Microsoft Windows, a network share is provided by the Windows network component "File and Printer Sharing for Microsoft Networks", using Microsoft's SMB (Server Message Block) protocol. Other operating systems might also implement that protocol; for example, Samba is an SMB server running on Unix-like operating systems and some other non-MS-DOS/non-Windows operating systems such as OpenVMS. Samba can be used to create network shares which can be accessed, using SMB, from computers running Microsoft Windows. An alternative approach is a shared disk file system (as NFS in Unix-like systems), where each computer has access to the "native" file system on a shared disk drive.

A Server part installed in the component will not be accessible in case of a virtual network configuration.

VNC, Remote Desktop Connection and Web-interface

Remote desktop refers to software or an OS feature allowing applications, often including graphical applications, to be run remotely on a server, while being displayed locally. There are various professional third-party, open source and freeware remote desktop applications, some of which are cross-platform across various versions of Windows, Mac, and UNIX/Linux/BSD. The main remote desktop protocols foreseen to be utilized in UrbanFlood are Virtual Network Computing (VNC) – a cross-platform protocol and Remote Desktop Protocol (RDP) – a Windows-specific protocol. This kind of component interface is intended for human interaction allowing remote administration, observation, debugging, configuration of components in a comfortable manner.

The same functions can be implemented via a Web-interface hosted in the component. Web-interfaces accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Java, AJAX, Adobe Flex, Microsoft .NET, or similar technologies can provide a wide range of visualization and control capabilities.

XenStore

XenStore is a storage space shared between a virtualization server host based on Xen hypervisor (including XenServer) and virtual machines hosted on it. XenStore is meant for the storage / exchange of configuration and status information rather than for large data transfers. Data in XenStore is organized as parameter=value pairs. Since there are no other means of component deployment to communicate parameters to the component, XenStore

is especially useful for initial component initialisation. In order to access XenStore data, the guest operating system shall be supported with paravirtualized drivers by Xen.

3.2 Interface Description

The most common case of a component and its interfaces in interaction with the CIS and the IntercloudManager is shown in Figure 4. No data and control flows concerned to the management of components are shown in the figure.

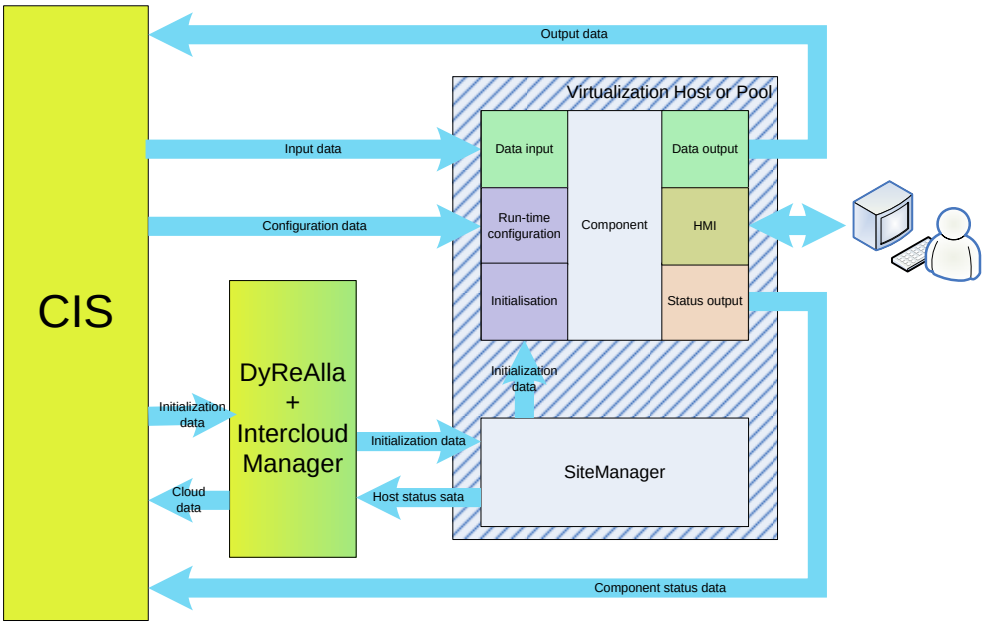


Figure 4 - Component and its interfaces in interaction with CIS

Data Interfaces

Data interfaces including data input and data output are intended to receive input data streams from the CIS and to send back computation or analysis results. They are the main operational interfaces of any component. They do not have to be implemented using the same interface technology. For legacy applications being adapted to become a component the choice of technologies is to be justified by reasonable adaptation efforts (e.g. applications taking input data from data files might be converted into component with FTP or file share data interface quite easily). The most preferable technologies for data interfaces are JMS and Web Service although all other are also allowable (excluding XenStore).

Configuration Interfaces

This group of interfaces include two rather different kinds: the initialization interface and the run-time configuration interface.

The initialization interface is used at the component deployment stage to communicate the parameters which are absolutely necessary to establish the connection to the CIS (since without it no further interaction would be possible). Those parameters include some of the following:

- TCP/IP protocol parameters:
    - IP address, network mask (for static IP configuration)
    - Default gateway
    - DNS server addresses
    - IP address type (external/internal)
- CIS connection parameters:
    - JMS provider address, port and topic(s) (in case of JMS-based data interfaces)
    - FTP server address or network share location (in case of FTP or network share based data interfaces with client part implemented within the component)
    - Web service address (in case of web service based data interfaces with service requester implemented within the component)
- Authentication and encryption parameters (if authentication is used):
    - Login
    - Password
    - Encryption method
    - Public keys

The only way to convey these parameters before the component is launched is to use XenStore interface. For that purpose, all the required parameters related to the CIS connection and authentication are sent from the CIS through the InterCloudManager to the SiteManager along with the command on component launch. TCP/IP configuration parameters are under responsibility of the SiteManager and are assigned in accordance of the internal site network configuration. The SiteManager puts all parameters into the XenStore of a particular VM before it is launched

A run-time configuration interface is used to configure or re-configure an application for the needs of a particular EWS or workflow. Thus the set of configuration data is application-specific. Preferred interface technologies are to be chosen on the basis of volume of configuration data. For rather simple sets of parameters JMS or web service is preferred

Comment [AO1]: Do not forget about payloass!!!

while for bulky pieces of data (such as training datasets or retrospective array of historical data) can be communicated by means of FTP or network share. In the case when a component plays a client role in FTP or network share interfaces, the actual server location can be communicated using JMS-based or a web service based configuration interface. A component can have both a simple configuration data interface and a bulky configuration data interface at the same time.

Status Interface

Status data output has two main functions – to report the current status of the component to the CIS and to indicate the component's aliveness as a heartbeat source. Examples of reported component statuses are:

- Waiting for configuration data
- Configuration in progress
- Configuration failed
- Ready for data processing
- Busy processing data
- Paused
- Failed

These statuses are to be used for EWS monitoring purposes and the detection of malfunctioning components. Status information can be reported in a periodic manner or upon request from the CIS. Requests can be addressed to particular component using a run-time configuration interface. Heartbeat messages can also be periodic or poll-based. Preferred interface technologies for this interface are JMS and web service. Transitions between component statuses are shown in 5 and described in Table 1.

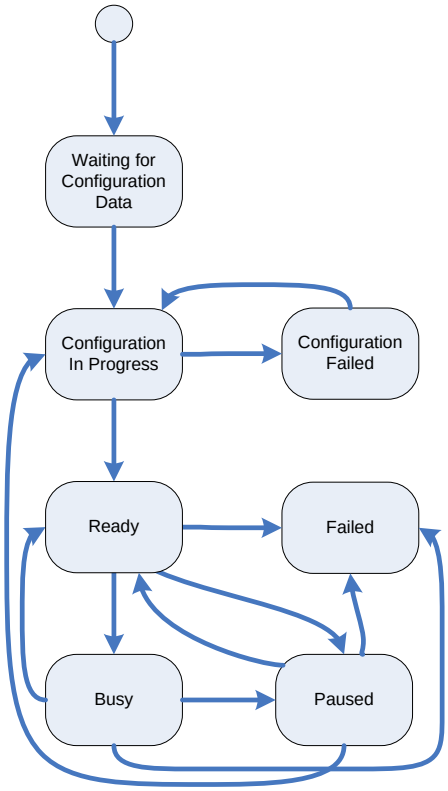


Figure 5 - Component Statuses

Table 1 - Component status transitions

Status	Description	Transition Condition	Target Status
Waiting for Configuration Data	Optional status, needed for components which cannot start working without proper run-time configuration data.	Runtime configuration data received	Configuration In Progress
Configuration In Progress	Optional status, designates that a component performs run-time configuration. This status might be used at startup by components which do not require configuration data, but perform time consuming internal configuration or initialization by themselves.	Configuration successfully completed	Ready
		Configuration failed	Configuration Failed

Status	Description	Transition Condition	Target Status
Configuration Failed	Optional status for components supporting run-time configuration, Unexpected failure encountered during configuration or configuration data invalid.	New configuration data received	Configuration in Progress
Ready	Status is to be reported when component is properly configured and ready for data processing.	Input data received	Busy
		Pause command received	Paused
		Unexpected failure encountered	Failed
Busy	Component is busy processing the data.	Data processing finished	Ready
		Pause command received	Paused
		Unexpected failure encountered	Failed
Paused	Component is paused by respective command from CIS. All processing jobs finished.	Resume command received	Ready
		New configuration data received	Configuration in Progress
		Unexpected failure encountered	Failed
Failed	An unexpected failure encountered, component cannot recover by itself.	VM is restarted	The very beginning of component statechart

Human-Machine Interface (HMI)

Human-machine interface is just a VNC, remote desktop connection or even a telnet or SSH connection which can be used for visualisation or for remote administration and configuration. It is not intended for CIS interaction.

4 Component Lifecycle

The lifecycle of a component starts when it is developed, wrapped into a VM and converted to a VM template to be stored on XenServer site awaiting deployment. The CIS stores the information (metadata) about the component in its repository. All component transitions are governed by the CIS. shows the sequence diagram of the component deployment process.

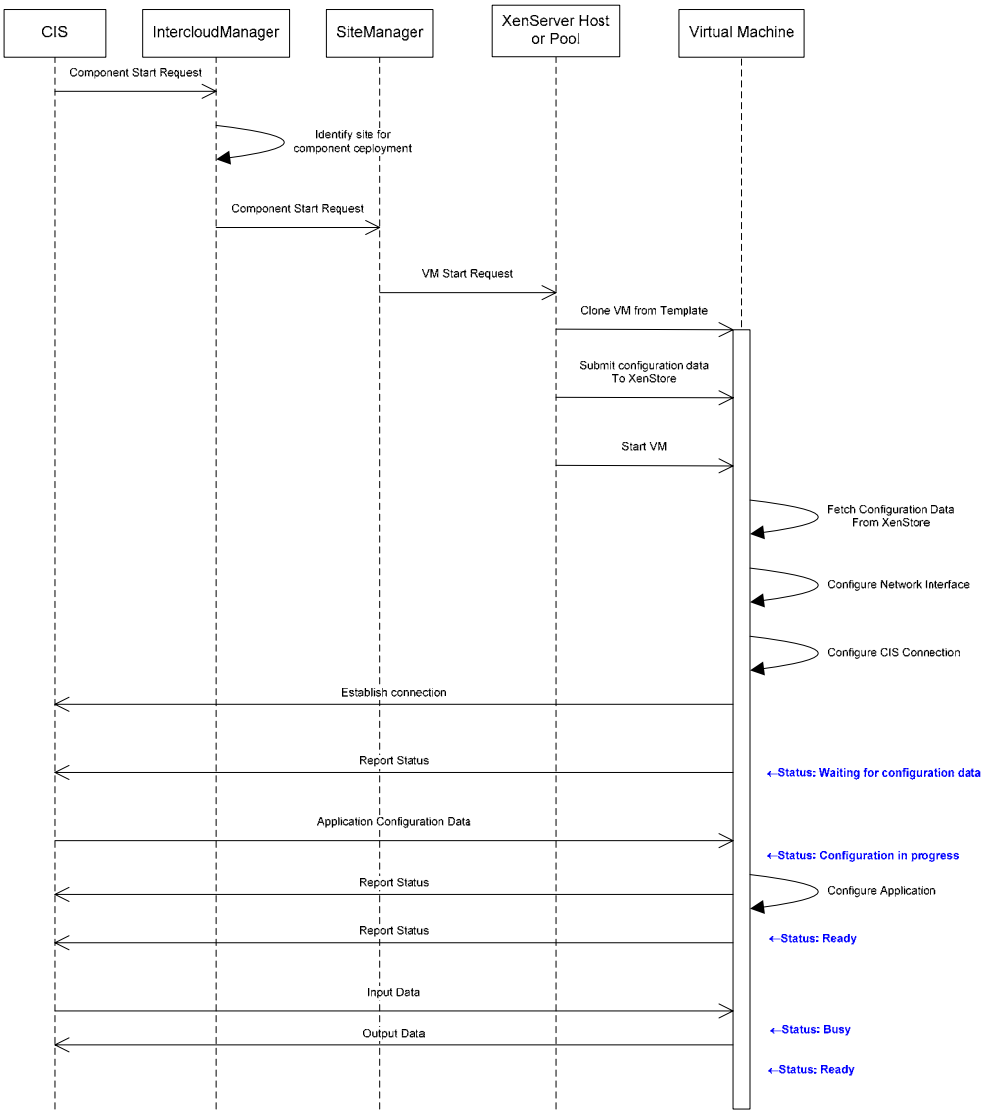


Figure 6 - Sequence Diagram of Component Deployment

After the decision on component deployment is taken in the CIS (e.g. new EWS or workflow is started) the CIS sends a request to the IntercloudManager to create a component instance

(this request shall also contain initial configuration data for CIS connections configuration). The IntercloudManager dispatches this request to the SiteManager of the site considered as most suitable for hosting of newly instantiated component. SiteManager creates a VM from a template and supplements initialization data with network interface configuration parameters. The SiteManager passes the initial configuration data to XenStore and launches the VM. Immediately after the OS boot application gets the data from XenStore, configures network interface for further communication and configures and establishes the connection to the CIS via data interfaces, run-time configuration interface and status interface. The current application status is reported to the CIS as an indication of successful launch of the component. Then the CIS sends run-time configuration data (if required) to adjust the application configuration to the needs of the current EWS or workflow. Configuration data are consumed and processed by the application. The status is reported again to indicate a component's readiness for further data processing. Now the component is ready to consume input data, perform required functions on them and emit output data back to the CIS.

The life of a component ends when it is no longer required by any EWS or workflow as shown in figure 7..

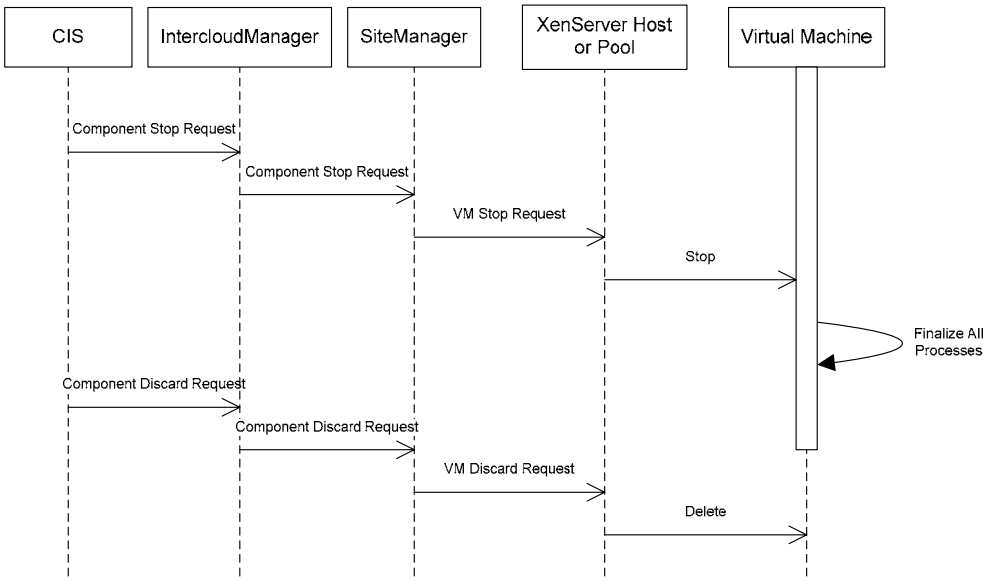


Figure 7 - Sequence Diagram of Component Removal

When such a decision is taken by the CIS it sends a stop request to the IntercloudManager which dispatches the request to the SiteManager of the site where the component is located. The SiteManager in its turn requests the subordinate host or pool to stop the particular VM. If the component is not planned to be used in another EWS or workflow, the CIS requests the IntercloudManager to remove the component. This request, relayed through the SiteManager to the XenServer host or pool in the same manner as described above, removes the instance of a VM.

Besides the start-up configuration, a component might be reconfigured at any time during normal operation as shown in Figure 8.

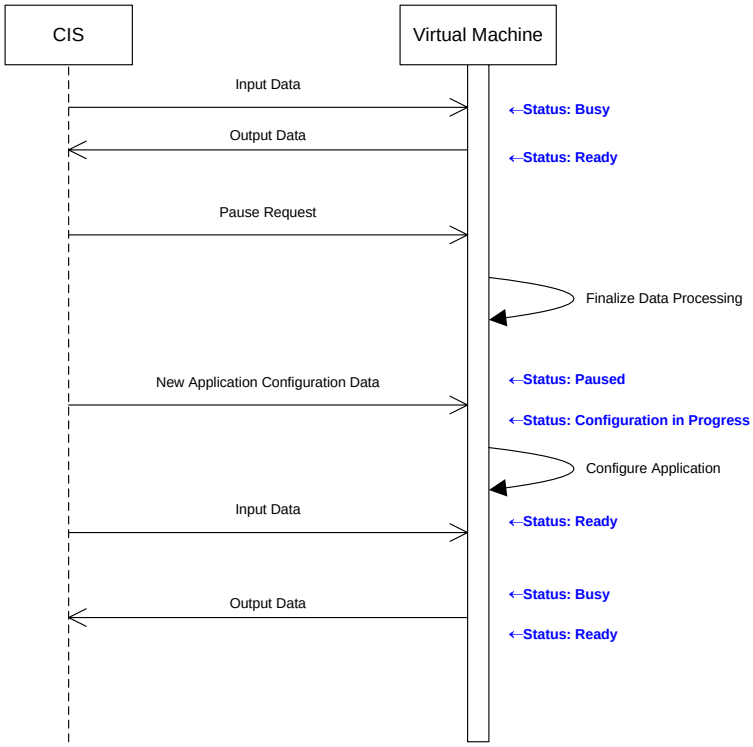


Figure 8 - Sequence Diagram of Component Reconfiguration

The component run-time reconfiguration is carried out under supervision of the CIS, so there are only two parties involved – the CIS and the Component itself. During normal operation (data processing) the CIS sends a stop request via the component configuration interface. The application finalizes processing and stops processing activities although the VM remains running and the application remains connected to the CIS. After the application is stopped and the corresponding status is reported the CIS sends new configuration data. Configuration data is handled in the same manner as upon start-up: configuration is performed and its completion reported as a status message. Then the normal operation continues.

5 References and bibliography

[1] TNO (2009a). UrbanFlood proposal Annex I – "Description of Work"

[2] Web Services Glossary, W3C Working Group Note 11 February 2004, http://www.w3.org/TR/ws-gloss/

[3] Web Services Architecture, W3C Working Group Note 11 February 2004, http://www.w3.org/TR/ws-arch/

[4] Java Message Service Specification - version 1.1, http://www.oracle.com/technetwork/java/docs-136352.html

[5] UrbanFlood Deliverable D5.2 - Specification of the architecture and interfaces of the Common Information Space