



Openwings Overview Ver 1.0 Final

COPYRIGHT ©2000-2003 General Dynamics Decision Systems, INC.
ALL RIGHTS RESERVED
This document is subject to "Terms of Use" as described at <http://www.openwings.org>.

PREFACE

About the Open Specification Process

A brief overview of the community process for developing Openwings Specifications is discussed below. This process was developed by General Dynamics Decision Systems and is similar to the community process for developing Java specifications.

A Process Management Organization (PMO) oversees development and maintenance of all Openwings Specifications. Any company, organization or individual can join the Specification development team; team members are called Participants. The latest Openwings Specifications will be maintained on a Public Web Site.

Participants can submit requests to the PMO to modify existing or develop new Specifications. If the PMO accepts the request, the PMO will then begin the process of forming an Expert Team and will appoint a Specification Lead.

The Specification Lead will direct the Expert Team in developing the new Specification. Once the team agrees on a draft of the Specification, it will be posted on the Public Web Site for public review. A member of the Expert Team will also begin development of a Reference Implementation and Compatibility Test Suite for the new Specification.

Based on public comments, a final release of the Specification will be produced by the Expert Team. Concurrently, the Reference Implementation and Compatibility Test Suite will be completed. Once the Final Public Release of the Specification is completed and posted on the Public Web Site, the Expert Team will disband.

This process is intended to provide rich consensus based Specifications.

Contributors

We would like to recognize and thank the following people for their contributions to this document.

Author(s):

Jeffrey Carpenter, Guy Bieber

Expert Team Members:

Stuart Lewin, BAE Systems North America

Ramesh Nagappan, Sun Microsystems

Ayal Spitz, Mitre Corp

Dr. Dave Usechak, Ocean Systems Engineering Corporation

Other Contributors:

Pat Vessels

1.	INTRODUCTION	1
2.	GOALS AND REQUIREMENTS	2
3.	ARCHITECTURE.....	3
3.1	RELATIONSHIP TO OTHER TECHNOLOGIES	4
4.	OPENWINGS DOCUMENTATION.....	6
4.1	SPECIFICATION FORMAT	7
4.2	DOCUMENT QUICK GUIDE.....	7
4.3	OPENWINGS WHITEPAPER.....	7
4.4	INTRODUCTION TO SERVICE-ORIENTED PROGRAMMING WHITE PAPER	7
4.5	OPENWINGS COMMUNITY OVERVIEW	7
4.6	ARCHITECTURE DESCRIPTION SPECIFICATION	7
4.7	INTERFACE SPECIFICATION	8
4.8	AVAILABILITY SPECIFICATION	8
4.9	SECURITY SPECIFICATION	8
4.10	COMPONENT SERVICES SPECIFICATION.....	8
4.11	CONNECTOR SERVICES SPECIFICATION	8
4.12	CONTAINER SERVICES SPECIFICATION	8
4.13	INSTALL SERVICES SPECIFICATION	9
4.14	POLICY SPECIFICATION	9
4.15	MANAGEMENT SERVICES SPECIFICATION	9
4.16	CONTEXT SPECIFICATION.....	9
4.17	DATA SERVICES SPECIFICATION.....	9
4.18	OPENWINGS TUTORIAL	9
4.19	OPENWINGS API JAVADOCS	10
5.	REFERENCES AND FURTHER READING.....	11

FIGURE 1:	OPENWINGS TOP LEVEL REQUIREMENTS	2
FIGURE 2:	SERVICE-ORIENTED PROGRAMMING ASPECTS AND ELEMENTS	3
FIGURE 3:	OPENWINGS ARCHITECTURE	3
FIGURE 4:	RELATIONSHIP TO JAVA PLATFORMS.....	5
FIGURE 5:	SUPPORTING TECHNOLOGIES.....	5
FIGURE 6:	SPECIFICATION DEPENDENCIES	6

1. Introduction

Openwings™ is an open community, non-proprietary effort to define specifications for self-forming, self-healing systems. Motorola (General Dynamics Decision Systems) and Sun Microsystems established the Openwings™ consortium in June of 1999. Since then, over 100 companies have registered to help mature its development, and more companies are registering daily.

The core Openwings™ framework is designed to incorporate existing commercial standards and is intended for use in both commercial and military environments. Openwings™ is being developed using a community development process modeled after the very successful Java Community Process. Anyone may join the Openwings™ community, participate on expert teams, or use the resulting specifications free of charge by merely signing up at the community web site (<http://www.openwings.org>).

The Openwings™ Architecture provides a framework for building plug-and-play, service-oriented, network-centric, self-forming, self-healing systems that are independent of middleware, databases, platforms, and deployment contexts. Openwings™ has a special focus on issues of availability, security, and interoperability. Openwings™ is the embodiment of a new movement in the software engineering community towards a paradigm known as Service-Oriented Programming (SOP). For an introduction to SOP, refer to <http://www.openwings.org/download/specs/ServiceOrientedIntroduction.pdf>.

2. Goals and Requirements

These are the top-level requirements for the Openwings Architecture. These requirements are allocated to the other specifications.

#	Requirement
1.	The architecture shall be split into a common piece (Openwings) and domains (FreedomC4I for military applications).
2.	The architecture shall be deployable on many hardware architectures, preferably through platform independent code.
3.	The architecture shall work in standalone mode, as part of a LAN, or as part of a WAN.
4.	The architecture shall have no central point of failure.
5.	The architecture shall be database independent and shall be able to talk to flat files, relational databases, and object databases.
6.	The architecture shall be transport layer independent, I.e. working with all types of asynchronous and synchronous middleware.
7.	The architecture shall be plug and play and auto-configuring at all levels (hardware and software components) to make self-forming systems. This component design should provide flexibility, scalability, and testability.
8.	The architecture shall have a minimal administrative overhead (zero admin).
9.	The architecture shall support 99.999% availability through hardware and software.
10.	The architecture shall support high availability in multiple hardware configurations (> 99.999% with redundancy, > 99.99% without redundancy).
11.	The architecture shall provide a 10x reduction in cycle time for the development of new systems.
12.	The architecture shall reduce maintenance costs by a factor of 10x.
13.	The architecture shall provide for network-based deployment, including upgrades and versioning.
14.	The architecture shall maximize reusability of components by externalizing all environmental information from the application business logic.
15.	The architecture shall provide security appropriate for military and non-military uses including: authentication, authorization, privacy, auditing, and multi-level security.
16.	The architecture shall scale from large-scale servers, to handheld devices.
17.	The architecture shall support a CORBA interface and hence legacy languages.
18.	The architecture shall support Java as the primary language and C++ as a secondary language.
19.	The architecture shall be self-tuning
20.	The architecture shall collect all critical software and hardware bit information dynamically.
21.	The architecture shall provide facilities to support hot fail-over of services and live update of services (software updates without a system reboot).
22.	The architecture shall provide facilities for bandwidth awareness.
23.	The application software implementation shall be vendor independent.
24.	The architecture shall be at least level 5 Defense Information Infrastructure (DII) Common Operating Environment (COE) 4.1 compliant. (In the future, this requirement will reference NCES instead of DII COE.)
25.	The Openwings shall be deployable using commercial or DII COE installers.
26.	The architecture shall be Levels of Information Systems Interoperability (LISI) level 4 compliant, I.e. application and data sharing.
27.	The architecture shall use standards specified in the Joint Technical Architecture (JTA 3.0) and achieve Technology Readiness Level (TRA) level 6.
28.	(DELETED)

Figure 1: Openwings Top Level Requirements

3. Architecture

The key value of the Openwings architecture is the ability to create systems composed of completely reusable components. Openwings is an instantiation of the Service-Oriented Programming (SOP) paradigm, which contains certain elements and aspects (see the Introduction to Service Oriented Programming White Paper for a more detailed discussion).

Elements	Aspects
Contracts	Conjunctive
Components	Deployable
Connectors	Mobile
Containers	Secure
Contexts	Available
	Interoperable

Figure 2: Service-Oriented Programming Aspects and Elements

Openwings provides a framework for SOP that is independent of discovery mechanisms, transport mechanisms, platforms, and deployment environments. The following figure shows a high level view of the Openwings Architecture:

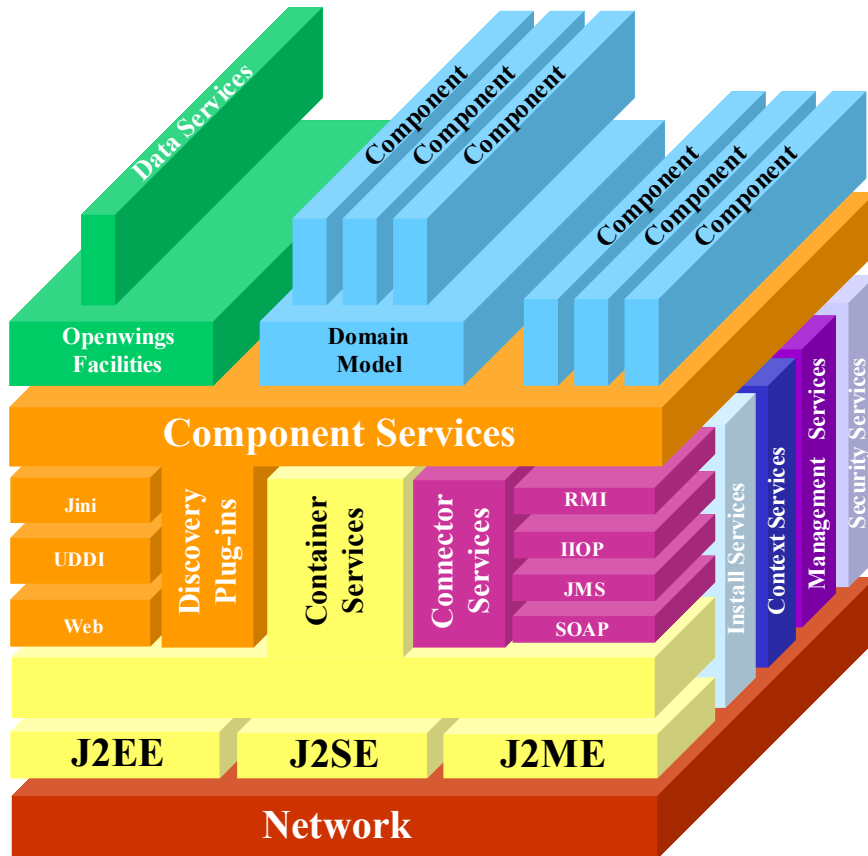


Figure 3: Openwings Architecture

Openwings is a network-centric architecture and hence the diagram rests on a network. The Openwings framework depends on Security, Management, Context, and Install Services. The Security Service provides code, transport, and service security. The Management Service allows management of any component in the framework. The Context Service provides a deployment environment and system boundary. The Install Service provides hands-off installation and propagation of static contextual information. The Container Service handles life cycle management of components and provides a clustering capability. The Connector Service provides protocol independence for asynchronous and synchronous protocols. The Component Service provides a simple programming abstraction for SOP and discovery independence. The unit of interoperability in Openwings is a Java Interface. Well-defined interfaces and code mobility provide the core enablers of Openwings interoperability.

On top of the Openwings framework components and domain models can be built. Openwings provides some common facilities, such as data services. The Data Service is built on the Openwings core and provides an object-based abstraction for persistence mechanisms that is independent of databases.

Each element in the Openwings Core can be used independently, with little or no interdependencies. This provides the greatest flexibility to Openwings users. In cases, where one service uses another, it is designed to integrate the service, but still run without it. For instance the container services can communicate with the install services to find new components.

3.1 Relationship to Other Technologies

As a Java-based systems architecture, Openwings has a unique focus on integration of systems that span from the enterprise down to personal devices. Thus Openwings is designed to integrate functionality of the Java platforms for enterprise, device, and desktop, as shown in the figure below.

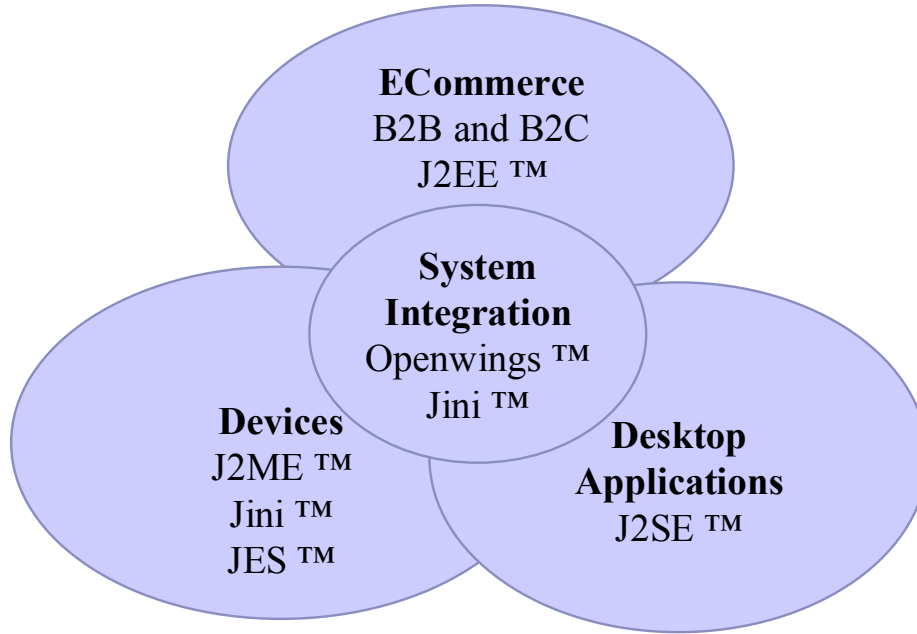


Figure 4: Relationship to Java Platforms

The Openwings framework is not intended as a replacement for existing Java software products and frameworks. Instead, the specifications are written at a higher level of abstraction so that implementations are free to use a variety of different Java based products and frameworks. The following figure shows which standards were especially relevant to each of the core service specifications:

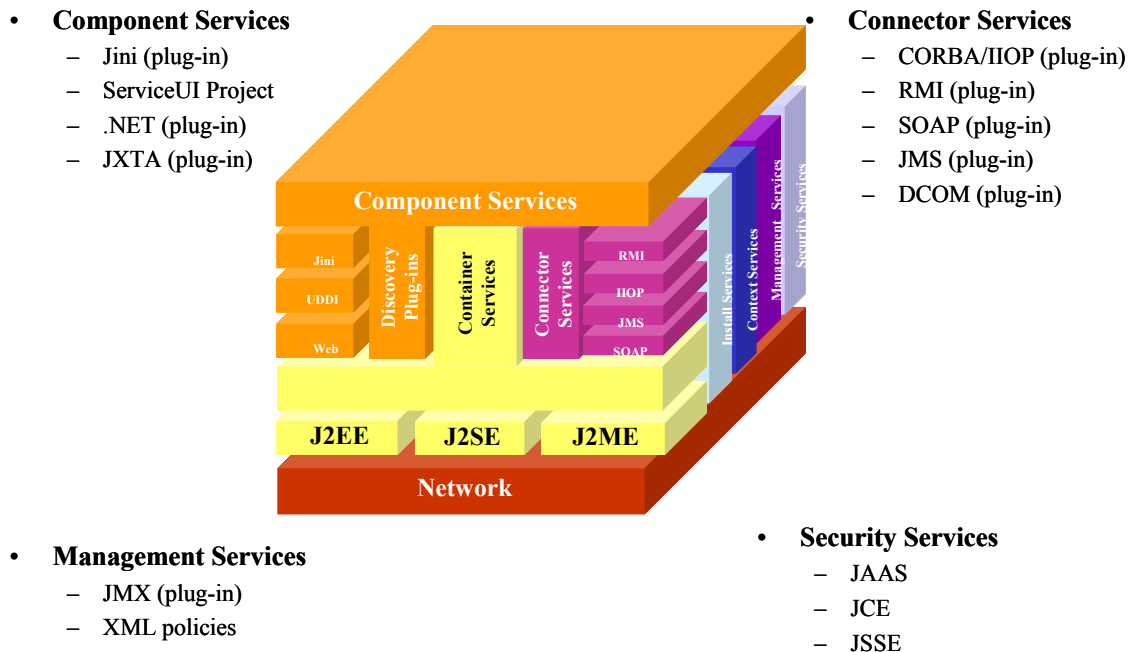


Figure 5: Supporting Technologies

4. Openwings Documentation

The following figure shows all of the specifications that make up the Openwings Architecture and the relationships between the specifications. Generally, documents depend on other documents in the layers above.

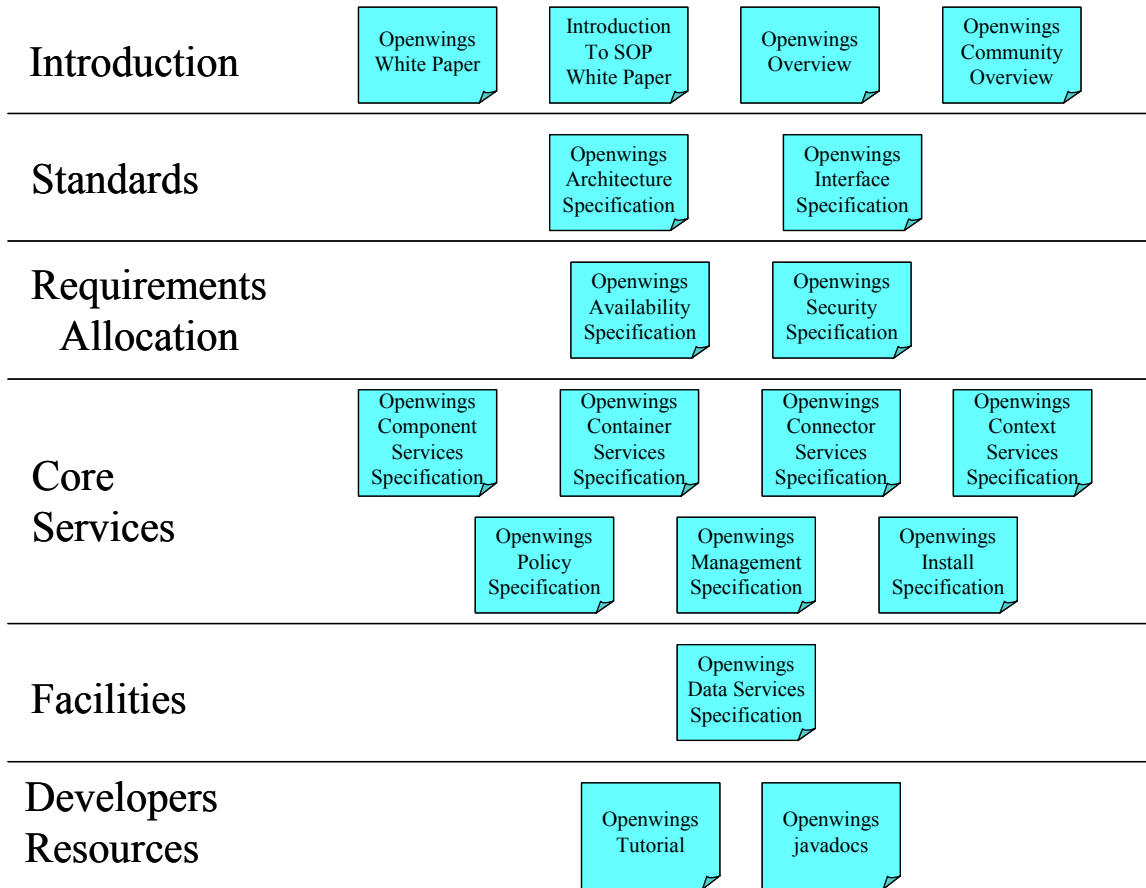


Figure 6: Specification Dependencies

The documents should be read in roughly the order shown in the diagram from the top down. The Openwings White Paper and this document are starting points for learning the architecture. The Architecture and Interface Specifications set standards that are used throughout the rest of the architecture. The Security and Availability Specifications show how these important aspects of the system are handled in hardware and software, and how availability and security requirements are allocated to the core services of the architecture. The Component Service, Connector Service, Container Service, Context Service, Policy, Management Service, and Install Service describe the core services of the architecture. The Data Service is a facility built on top of the core framework. The Openwings Tutorial is an HTML-based training site that is expected to be specific to each implementation of Openwings.

4.1 Specification Format

All of the Openwings specifications follow the same basic format:

- Goals and Requirements – Describes requirements of the specification.
- Use Cases – Describes UML use cases for the specification.
- Architecture – Describes the various architectural aspects of the specification.
- Interfaces – Describes interfaces required for the specification.
- Tools – Describes various tools and utilities provided with the specification.
- Examples – Provides representative real world use cases for the specification.
- Compliance – This section describes what is required to comply with the specification.

4.2 Document Quick Guide

To get up to speed quickly on Openwings the following is the recommended reading order.

1. Openwings Overview – this document
2. Openwings White Paper
3. Introduction to Service-Oriented Programming White Paper
4. Openwings Tutorial
5. Openwings API Javadocs

The other specifications should be read on an as needed basis, for instance if one is trying to implement one of the specifications or get further background.

4.3 Openwings Whitepaper

This document serves as a top-level introduction to Openwings. It develops the military and commercial use cases of the architecture. This document also provides an overview of the core services of the architecture.

4.4 Introduction to Service-Oriented Programming White Paper

This white paper develops the elements and aspects of Service-Oriented Programming and why it is important.

4.5 Openwings Community Overview

This document describes the Openwings Community Process, which was modeled after the Java Community Process.

4.6 Architecture Description Specification

This document describes the Openwings architectural component model for plug and play software / hardware components. The terms and notation for the architecture are defined. Modeling languages used are introduced, including Architecture Description Language (ADL), Unified Modeling Language (UML), and Interface Definition Language (IDL).

4.7 Interface Specification

This document describes the requirements for compliant service interfaces in the Openwings framework. This document describes rules for defining synchronous interfaces, asynchronous interfaces, data interfaces, and attributes. It also describes how to create interface converters for integration of legacy interfaces.

4.8 Availability Specification

This document introduces availability concepts and defines the techniques in the Openwings architecture for achieving high availability systems, focusing specifically on software availability techniques. This document describes how availability requirements are allocated to the core services of the architecture and provides an API for additional issues. Openwings targets High Availability systems by providing component restart, fail-over, load balancing, and upgrades.

4.9 Security Specification

This document describes code, transport, and service security in the Openwings framework. This document portrays the security aspect of the Openwings architecture in whole, and it describes how security requirements are allocated to the core services of the architecture. The basic strategy is to use existing security technology to achieve secure systems. Openwings focuses on transport, code, and service security, as solutions exist for platform and network security already. This document covers all aspects of security including authorization, authentication, confidentiality, integrity, attack detection, and attack response.

4.10 Component Services Specification

This document describes the Openwings component model. Component Services create the ability to use and provide services. Component Services allow plug-in discovery mechanisms. This is the primary abstraction developers must deal with when using Openwings.

4.11 Connector Services Specification

This document describes the protocol abstraction used in Openwings. A connector is an implementation of a specified service interface that uses some protocol to communicate between a component providing a service and the component that is using the service. Connectors can be synchronous or asynchronous. Example synchronous connectors are RMI and CORBA IIOP. Example asynchronous connectors are JMS and CORBA Message Service (CMS). This document covers the generation, storage, and use of connectors.

4.12 Container Services Specification

The Container Service provides the publication of processing as a service. The Container Service is also responsible for component life cycle management, availability features, and enforcement of code security policies. Life cycle management includes start, stop, activation, and mobility. Availability features include fail-over, and rolling updates.

4.13 Install Services Specification

The Install Service is responsible for installation of components on Openwings-enabled systems. This includes the authentication of components, zero-interaction install, dependency resolution, and policy resolution. The Install Service manages the separation of components and the publication of components.

4.14 Policy Specification

This document defines the Openwings policy framework. Policies provide a bridge through which components can interact with their run-time environment, through policy resolution. This effectively de-couples a component from its environment, enabling the component to be deployed in different environments without modification of its core logic. This document describes how policies are represented, deployed, discovered, translated from objects to external representation and back.

4.15 Management Services Specification

This document defines the Openwings framework for management of components and services. The framework makes it possible to automatically manage system operations including security, system formation, and services. The framework provides a standard way to add management plug-ins, called Management Beans (MBeans), to manage different aspects of component behavior. These management beans allow operations to be performed at runtime on components that are outside the components' normal operating interfaces. Management user interfaces may be attached in the same way a service interface is attached to a published service. These may be deployed with an MBean or located separately.

4.16 Context Specification

The Openwings Context specification describes how components are grouped together to form systems. The Openwings Context concept includes boundary security as well as a mechanism to allow services to be shared beyond the context, by establishing relationships with other contexts. This document enumerates the core services needed to form a context.

4.17 Data Services Specification

This document defines the persistence abstraction used in Openwings. Data stores are provided as services to the network through components called Data Servers. This document describes how Data Servers are developed for both relational and object databases, or even flat files. The data server model also provides facilities to centralize queries for optimization.

4.18 Openwings Tutorial

This website contains everything a developer needs to know to begin developing components and systems with Openwings. Example components are presented that demonstrate Openwings features. This tutorial is specific to the reference implementation.

4.19 Openwings API Javadocs

Standard Javadocs are provided for all of the interfaces and classes needed to define Openwings, i.e. those in `net.openwings.*` packages.

5. References and Further Reading

1. Openwings Architecture Whitepaper,
<http://www.openwings.org/download/specs/openwingswp.pdf>
2. Openwings Architecture Specification,
http://www.openwings.org/download/specs/Openwings_Architecture_Description.pdf
3. Openwings Availability Specification,
http://www.openwings.org/download/specs/Openwings_Availability.pdf
4. Openwings Security Specification,
http://www.openwings.org/download/specs/Openwings_Security.pdf
5. Openwings Component Services Specification,
http://www.openwings.org/download/specs/Openwings_Component_Services.pdf
6. Openwings Connector Services Specification,
http://www.openwings.org/download/specs/Openwings_Connector_Services.pdf
7. Openwings Container Services Specification,
http://www.openwings.org/download/specs/Openwings_Container_Services.pdf
8. Openwings Policy Services Specification,
http://www.openwings.org/download/specs/Openwings_Policy_Services.pdf
9. Openwings Management Specification,
http://www.openwings.org/download/specs/Openwings_Management_Services.pdf
10. Openwings Install Service Specification,
http://www.openwings.org/download/specs/Openwings_Install.pdf
11. Openwings Context Specification,
http://www.openwings.org/download/specs/Openwings_Context.pdf
12. DII COE: <http://spider.osfl.disa.mil/dii/>
13. C4ISR: http://www.c3i.osd.mil/org/cio/i3/AWG_Digital_Library/index.htm
14. LISI: http://www.c3i.osd.mil/org/cio/i3/AWG_Digital_Library/index.htm
15. JTA 3.0: <http://www-jta.itsi.disa.mil/>
16. AITS: <http://www-code44.spawar.navy.mil/cpof/>