

# **NESI Part 5: Net-Centric Developer's Guide**



# Table Of Contents

<b>NESI implementation framework</b> .....	<b>1</b>
References .....	2
Overview .....	3
Releasability statement .....	4
Vendor neutrality.....	5
Disclaimer .....	6
Contributions and comments.....	7
Open-source site .....	8
NESI development guidance .....	9
Documentation structure .....	10
<b>Technical guidance and tactics</b> .....	<b>11</b>
High-level guidance .....	12
Publish and insulate public interfaces.....	12
Implement a component-based architecture.....	12
Automate the software build process.....	13
<b>Middle tier</b> .....	<b>15</b>
J2EE environment .....	18
Guidance .....	19
Best practices.....	19
Examples .....	19
References .....	21
Web service models.....	22
Key characteristics .....	24
Guidance .....	25
Examples .....	26
Navy operational example: Exposing web services for METOC .....	26
References .....	27
References .....	28
SOAP .....	29
Guidance .....	31
Examples .....	32
Web services .....	35
Guidance .....	37
Examples .....	37
References.....	38
.NET framework.....	39
Web services.....	39
<b>Data tier</b> .....	<b>41</b>
Decouple databases from applications .....	42
Guidance .....	42
Database implementations .....	43
Guidance .....	43
Guidance .....	45
Best practices .....	46
RDBMS internals .....	47
Guidance .....	47
Best practices.....	47
XML .....	49
References.....	49
Wrapping XML parsers .....	49

Parsing XML strategies .....	52
<b>Networks and enterprise services .....</b>	<b>55</b>
Discovery .....	57
Directory .....	57
Lightweight Directory Access Protocol (LDAP) .....	58
Example: Java JNDI to LDAP .....	58
Java Naming & Directory Interface (JNDI) .....	62
Universal Description, Discovery, and Integration (UDDI) .....	62
References .....	63
Quality of Service (QoS) .....	64
Overview .....	64
References .....	65
<b>Communications and transport .....</b>	<b>67</b>
Joint Tactical Radio System (JTRS) .....	68
Overview .....	68
Example: SCA-compliant software component .....	74
References .....	96
<b>Reference implementations .....</b>	<b>101</b>
GIS display environments .....	102
Goals .....	102
NESI strategy .....	102
Migration strategies .....	103
OGC WS architecture .....	103
Web Feature and Coverage Services .....	104
OGC API .....	108
Examples: GIS open architecture .....	112
Implementing GIS open architecture .....	150
Migrating to GIS open architecture .....	161
Mobile devices .....	164
Overview .....	164
Best practices .....	164
Wireless cell phone environments .....	164
PalmOS 4 .....	165
<b>Guidance .....</b>	<b>169</b>
Guidance details .....	170
G1001 .....	171
G1002 .....	172
G1003 .....	173
G1004 .....	174
G1005 .....	175
G1007 .....	176
G1008 .....	177
G1010 .....	178
G1011 .....	179
G1012 .....	180
G1014 .....	181
G1018 .....	182
G1019 .....	183
G1020 .....	184
G1021 .....	185
G1022 .....	186
G1027 .....	188
G1030 .....	189
G1031 .....	190

G1032	192
G1035	193
G1037	194
G1043	195
G1044	196
G1045	197
G1049	198
G1050	199
G1052	200
G1053	201
G1055	202
G1056	203
G1058	204
G1060	205
G1071	206
G1073	207
G1078	208
G1079	209
G1080	210
G1082	211
G1083	212
G1084	213
G1085	214
G1086	215
G1087	217
G1088	218
G1090	219
G1091	223
G1093	224
G1094	225
G1095	226
G1101	227
G1117	228
G1118	229
Example	229
G1119	236
G1121	237
G1123	241
G1126	242
G1127	243
G1131	244
G1132	247
G1141	248
G1144	249
G1146	250
G1147	251
G1148	252
G1151	253
G1154	254
G1155	255
G1190	256
G1200	257
G1201	258
G1202	259
G1203	263
G1204	265

G1205	271
G1208	272
G1209	273
G1210	274
G1211	275
G1212	276
G1213	277
G1214	278
G1215	279
G1216	280
G1217	281
G1218	282
G1219	283
G1220	284
G1221	285
G1222	286
G1223	287
G1224	288
G1225	289
G1236	290
G1237	291
G1239	292
G1245	293
<b>Best practices</b>	<b>297</b>
Best practices details	298
BP1038	299
BP1039	300
BP1040	301
BP1041	302
BP1042	303
BP1054	304
BP1075	305
BP1076	306
BP1077	307
BP1097	308
BP1098	309
BP1100	310
BP1109	311
BP1111	312
BP1112	313
BP1116	314
BP1122	315
BP1139	316
BP1140	317
BP1143	318
BP1145	319
BP1177	320
BP1226	321
BP1227	322
BP1228	323
BP1229	324
BP1230	325
BP1231	326
BP1232	327
BP1233	328

BP1234	329
BP1235	330
BP1240	331
BP1241	332
BP1242	333
BP1243	334
BP1244	335
BP1246	337
BP1247	338
BP1248	339
BP1249	340
BP1250	341
BP1251	342
BP1252	343
BP1253	344
BP1254	348
BP1255	349
BP1256	350
BP1257	352
BP1258	353
BP1259	354
BP1260	355
BP1261	356
BP1262	357
BP1263	358
BP1264	359
BP1265	360
<b>Appendices</b>	<b>361</b>
Technical References	362
Books	362
Web sites	364
Automated testing tools	368
Environments	369
Security testing tools	370
Namespace management procedures	371
Mobile code	374
Java developer programs	375
Navy-specific guidelines	376
COE-M build lists	376
COMPOSE software list	386
Network security policy guidance	388
Cross-reference between NESI and other initiatives	389
Navy Enterprise Portal (NEP) architecture	389
Open-source tools	392
Apache Ant	392
Apache Axis	393
Tomcat	395
Xalan-Java	395
Xerces2 Java Parser	396
jUDDI	398
UDDI browsers	409
<b>Glossary</b>	<b>417</b>
<b>Index</b>	<b>467</b>



---

# **NESI implementation framework**

## References

- (a) **DoD** Directive 5000.1, *The Defense Acquisition System*, 24 November 2003.
- (b) DoD Instruction 5000.2, *Operation of the Defense Acquisition System*, 12 May 2003.
- (c) DoD Directive 8100.1, *Global Information Grid (GIG) Overarching Policy*, 21 November 2003.
- (d) DoD Directive 4630.5, *Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS)*, 05 May 2004.
- (e) DoD Instruction 4630.8, *Procedures for Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS)*, 30 June 2004.
- (f) DoD Directive 5101.7, *DoD Executive Agent for Information Technology Standards*, 21 May 2004.
- (g) *DoD Global Information Grid (GIG) Architecture, Version 2.0*, August 2003.
- (h) *DoD Joint Technical Architecture, Version 6.0*, 3 October 2003.
- (i) *DoD Net-Centric Data Strategy*, DoD Chief Information Officer, 9 May 2003.
- (j) **CJCSI** 3170.01D, *Joint Capabilities Integration and Development System*, 12 March 2004.
- (k) CJCSM 3170.01A, *Operation of the Joint Capabilities Integration and Development System*, 12 March 2004.
- (l) CJCSI 6212.01C, *Interoperability and Supportability of Information Technology and National Security Systems*, 20 November 2003.
- (m) *Net-Centric Operations and Warfare Reference Model (NCOW RM) V1.0*, September 2003.
- (n) *Net-Centric Checklist, V2.1.3*, Office of the Assistant Secretary of Defense for Networks and Information Integration/Department of Defense Chief Information Officer, 12 May 2004.
- (o) *A Modular Open Systems Approach (MOSA) to Acquisition, Version 2.0*, September 2004.
- (p) DoD IT Standards Registry (**DISR**), <http://disronline.disa.mil>.
- (q) *Net-centric Attributes List*, Office of the Assistant Secretary of Defense for Networks and Information Integration/Department of Defense Chief Information Officer, June 2004.

# Overview

**Net-centric Enterprise Solutions for Interoperability (NESI)** is a joint effort between the U.S. Navy's Program Executive Office for C4I & Space and the U.S. Air Force's Electronic Systems Center. It provides implementation guidance that facilitates the design, development, maintenance, evolution, and use of information systems for the Net-Centric Operations and Warfare (NCOW) environment. NESI has also been provided to other Department of Defense (DoD) services and agencies for potential adoption.

The NESI Implementation guidance applies to all phases of the acquisition process as defined in references (a) and (b). NESI comprises six parts, each focusing on a specific area of guidance. *NESI Part 1: Net-centric Overview* describes each part in detail.

NESI provides guidance on software development best practices, and examples for developing Net-Centric software. It is aligned with the design principles of reference (o). NESI is not a replacement for references (h), (m), or (n).

The overall goal is to provide common, cross-service guidance in measurable terms for the program managers and developers over the lifecycle of net-centric solutions. The objective is not to replace or repeat existing standards or guidance, but to organize, clarify, and reconcile conflicting mandates around the acquisition process..

NESI subsumes a number of references and directives: in particular, the Air Force *C2 Enterprise Technical Reference Architecture (C2ERA)*[1] and the Navy *Reusable Applications Integration and Development Standards (RAPIDS)* Initial authority for NESI is per the Memorandum of Agreement between Commander, Space and Naval Warfare Systems (SPAWAR), PEO C4I & Space, and the United States Air Force Electronic Systems Center, dated 22 December 2003, Subject: Cooperation Agreement for Net-Centric Solutions for Interoperability (NESI).

In addition to references (a) through (q), Navy PEO C4I has mandated a software maintenance policy for its programs that requires the use of *NESI Part 3: Net-Centric Migration Guidance*.

NESI is intended to help programs comply with the DoD Net-Centric directives, instructions, and other guidance documentation (listed as references (a) through (q)). This guidance will continue to evolve as direction and our understanding of the requirements of net-centricity evolve. NESI will be updated to reflect changes to the guiding documents and new regulations.

[1] Air Force C2 Enterprise Technical Reference Architecture, v3.0-14, 1 December 2003.

[2] RAPIDS Reusable Application Integration and Development Standards, Navy PEO C4I & Space, December 2003 (DRAFT V1.5), <https://nesi.spawar.navy.mil>.

[3] Software Maintenance Policy, Department of the Navy, Navy PEO C4I & Space, 14 June 2004.

## Releasability statement

This document has been cleared for public release by competent authority in accordance with DoD Directive 5230.9 and is granted Distribution Statement A: Approved for public release; distribution is unlimited. You may obtain electronic copies at <https://nesi.hanscom.af.mil> or <https://nesi.spawar.navy.mil>.

## Vendor neutrality

The *NESI* documentation sometimes refers to specific *vendors* and their products in the context of examples and lists. However, NESI is vendor-neutral. Mentioning a vendor or product is not intended as an endorsement, nor is a lack of mention intended as a lack of endorsement.

Code examples typically use *open-source* products, since NESI is built on the open-source philosophy. Since NESI accepts contributions from multiple sources, the examples also tend to reflect whatever tools the contributor was using or knew best. However, the products described are not necessarily the best choice for every circumstance. You are encouraged to analyze your specific project requirements and choose your tools accordingly. There is no need to obtain, or ask your contractors to obtain, the open-source tools that appear as examples in this guide. Similarly, any lists of products or vendors are intended only as references or starting points, and not as a list of recommended or mandated options.

## **Disclaimer**

Every effort has been made to make this documentation as complete and accurate as possible. It is expected that the documentation will be updated frequently, and will not always immediately reflect the latest technology or guidance.

## Contributions and comments

**NESI** is an open-source project that will involve the entire development community. Anyone is welcome to contribute comments, corrections, or relevant knowledge to the guides. For Navy contributions, send email to [nesi@spawar.navy.mil](mailto:nesi@spawar.navy.mil). For Air Force contributions, send email to [nesi@hanscom.af.mil](mailto:nesi@hanscom.af.mil).

## Open-source site

The Navy has established an open-source site to support community involvement. It is located at <https://nesi.spawar.navy.mil>. This evolved from the Navy **RAPIDS** initiative. Use this site for collaborative software development across distributed teams.

# NESI development guidance

This developer's guide provides chief engineers and software developers with detailed implementation guidance for applications, services, and data. This effort leverages current best practices from the software development community to enable the DoD to create *net-centric*, extensible, scalable enterprise applications. The goal is to modernize and improve the development of Net-Centric applications and services as critical warfighter capabilities.

Software developers can choose to use published applications via interfaces and services or build applications and services that interface with the infrastructure. Any application that must interoperate in the DoD Net-Centric enterprise should be built and maintained in accordance with the standards, policies, and processes within this guide.

The tactics described in this document are designed to:

- Permit independent paces of development and change on each side of the enterprise, reducing risk and impacts of changes to application developers.
- Implement connection strategies that extend the life and reach of legacy applications while legacy application developers restructure their systems.

## Documentation structure

This document provides developers with detailed software development guidance, best coding practices, lessons learned, and code samples. *It* is intended as a reference, not a document to be read cover to cover.

The contents follow this basic structure:

- **Overview:** Describes the topic in terms suitable for the entire *NESI* audience, and lists future topics that may be covered in that area
- **Guidance:** Lists contractual statements relating to the topic.
- **Best practices:** Contains lessons learned from industry and the DoD, design patterns, code snippets, and configuration examples; developers can augment their efforts by leveraging and reusing this information
- **Examples:** Provides code samples that illustrate the guidance and best practices. For a statement about the choice of tools, see the *Vendor neutrality* disclaimer.
- **Glossary:** Defines jargon and terms used in a specific sense.
- **References:** Lists of books, web sites, and other sources of information that may assist the planning or development effort.

Program managers and chief engineers will find the overview and guidance sections helpful while:

- Directing their programs and activities to build systems. Use this information in combination with *NESI Part 2: Net-Centric ASD (NII) Checklist Guidance* and *NESI Part 4: Net-Centric Node Design Guidance*.
- Reviewing Statements of Work. (Developers may also use the information for this purpose.)
- Reviewing deliverables for compliance.
- Migrating legacy systems to the net-centric environment. Use this information in combination with *NESI Part 3: Net-Centric Migration Guidance*.

---

# Technical guidance and tactics

This section contains guidance on the following topics:

- *High-level guidance*
- Interface design

Future guidance will include:

- **Design patterns and examples:** Recommended patterns and implementations
- **Developer's Toolkit creation:** Toolkit containing a jumpstart/quick start guide, developer's guide, sample code, automated test drivers and certification tools, and access to open-source sites.
- **Enterprise Checklist:** Overview of actions prior to enterprise deployment.
- **Error handling:** Error management processes and guidelines.
- **Interface management:** Public interface management processes and guidelines.
- **Logging management:** Logging and auditing processes and guidelines.

Note that this guidance may be moved to other sections of the *NESI* documentation, as appropriate.

## High-level guidance

This section lists high-level guidance for developing *Net-Centric* software. The remainder of this document provides more detailed guidance on specific topics. Adhering to the guidance in this document will minimize impacts to programs and help manage change.

- *Publish and insulate public interfaces*
- *Implement a component-based architecture*
- *Automate the software build process*

## Publish and insulate public interfaces

This section lists high-level guidance for implementing public interfaces.

### Guidance

- Define public interfaces in a formal standard. [G1001]
- Separate public interfaces from the implementation of an application to control change between evolving applications and the evolving enterprise. [G1002]
- Separate the contents of application libraries that are to be shared from libraries that are to be used internally. [G1003]
- Make public interfaces backward-compatible within the constraints of a published deprecation policy. [G1004]
- Separate infrastructure capabilities from mission functions. [G1005]
- Ensure that applications use open, standardized, vendor-neutral API(s). [G1007]
- Isolate platform-specific interfaces and vendor dependencies. [G1008]
- Use open-standards logging frameworks. [G1010]
- Insulate public interfaces from compile-time dependencies. [G1022]

## Implement a component-based architecture

A component-based architecture (CBA) is:

*"An architecture process that enables the design of enterprise solutions using pre-manufactured components. The focus of the architecture may be a specific project or the entire enterprise. This architecture provides a plan of what needs to be built and an overview of what has been built already." Succeeding with Component-Based Architecture*

CBA represents a shift from the traditional, custom-development-oriented, "design, code, and test" approach that has been used throughout the DoD in the past to a more business-oriented "architect, acquire, and assemble" approach.

The custom-development approach has been successful in building many systems. However the integration, evolution, reuse and cost of these systems have presented a problem. Consequently,

these custom-developed systems have been labeled as archaic stovepipes that can not plug-and-play with other systems.

CBA promises benefits such as shorter time to market, lower risk, and modular and adaptive systems.

The core of CBA is components. Consequently the term **component** needs to be defined. The following guidance statements capture the essence of components.

### Guidance

- All components must be independently deployable. [G1011]
- Components should expose functionality through a set of services. [G1012]
- Components should be externally configurable. [G1217]

## Automate the software build process

A software build process interfaces with source control, compiles code, creates executables, runs unit tests, packages and deploys, and generates documentation. An automated software build process is a necessary part of every software development project and ensures the software will be built in the same manner each time.

### Guidance

Use a build tool that: [G1190]

- Supports operation in an automated mode. [G1218]
- Checks out files from configuration control. [G1219]
- Compiles source code and dependencies that have been modified. [G1220]
- Creates libraries or archives after all required compiles are complete. [G1221]
- Creates executables. [G1222]
- Is capable of running unit tests. [G1223]
- Cleans out intermediate files that can be regenerated. [G1224]
- Is independent of any Integrated Development Environment. [G1225]

### Best practices

- All application developers should use the **ANT** build tool to build, package, and deploy J2EE applications. [BP1075]

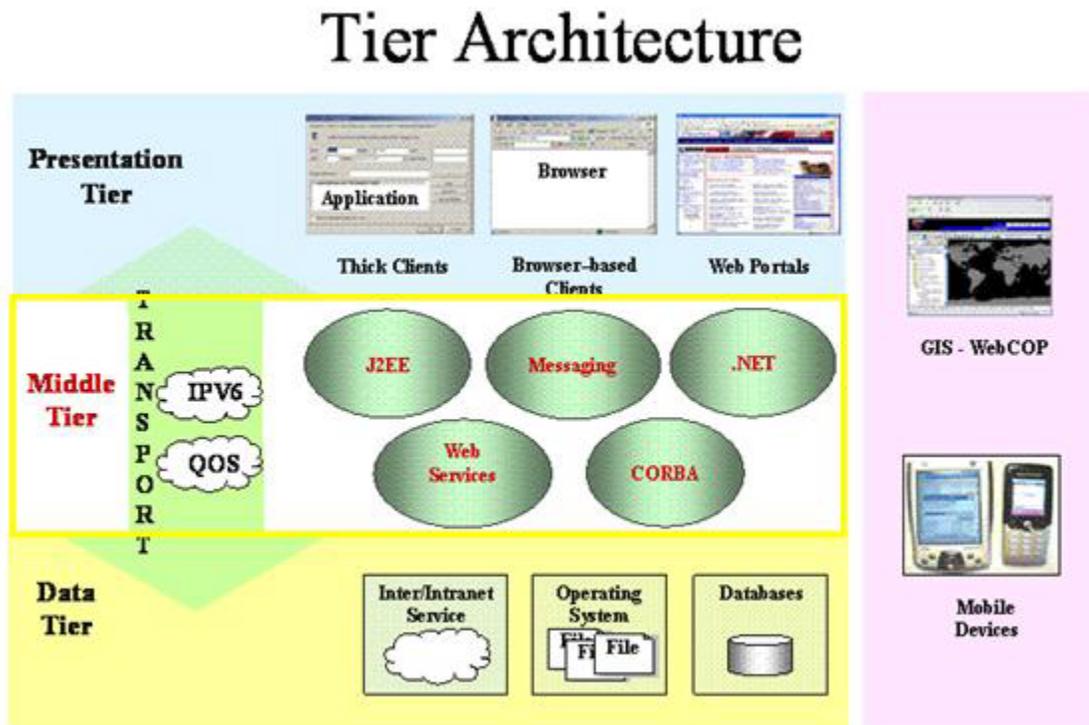
### References

- ANT - <http://ant.apache.org/>
- J2EE - <http://java.sun.com/j2ee/>



# Middle tier

The middle tier provides process management services such as process development, monitoring, and resourcing, that are shared by multiple applications.



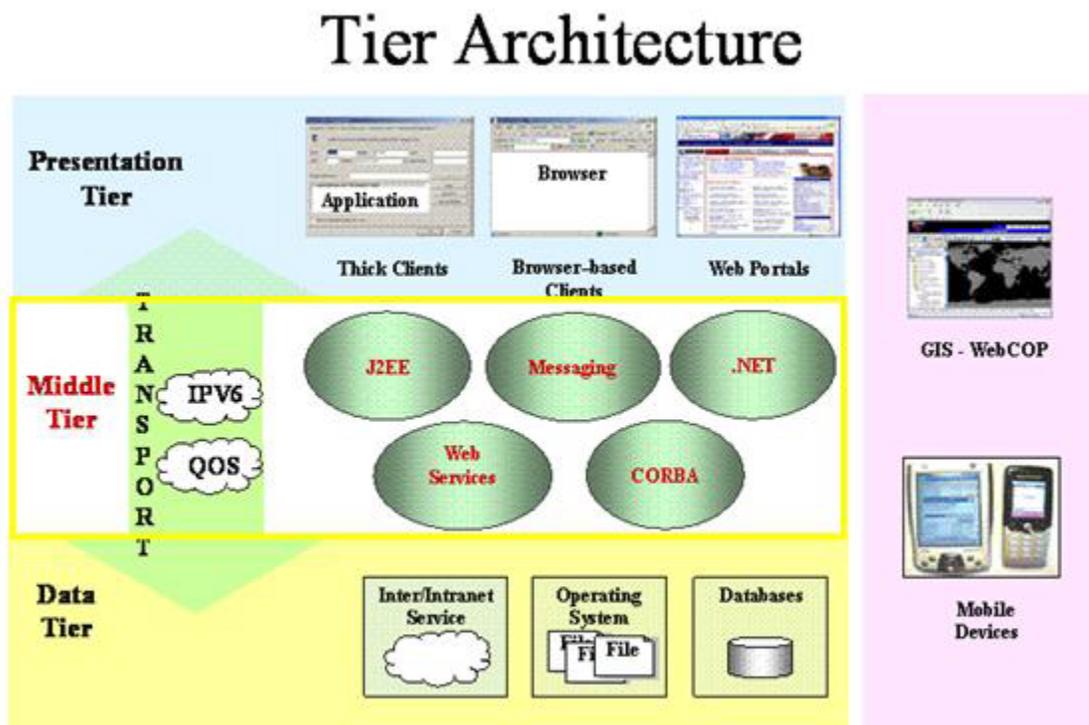
Future guidance will include:

- **Application collaboration/Mediation framework:** Also known as backend integration application communication.
- **Application concurrency control:** Concurrency and locking strategies and guidelines for applications required to operate in a multi-user environment. Transactional strategies for operations with other services in the enterprise.
- **Application server guidelines:** Sybase application server topics, transactions, and data access guidelines.
- **Connection strategies:** Applications written in or using Fortran, Ada, C/C++, Cold Fusion, Java, *J2EE*, Microsoft Office, and *.NET*.
- **CORBA: Real-time** topics, cross-vendor interoperability issues, enterprise connection strategies, and Software Communication Architecture (*SCA*) issues.
- **Design patterns and examples:** Recommended patterns and implementations.

- **Microsoft component model:** .NET, COM/DCOM, COM+, security, and data-access guidelines.
- **Microsoft Office:** Connector strategies to and from the enterprise.
- **Middleware guidelines:** Guidelines on developing connectors to and from the enterprise.
- **Other application server operations:** JBoss, Orion, Sybase EAServer.
- **Security guidelines:** Authentication schemes, secure coding practices, digital certificates, digital signatures, firewall polices, protection mechanisms, and *SSL*.
- **Transactional strategies:** For operations with other services in the enterprise.
- **Web services:** *UDDI* operations and taxonomies.

Note that this guidance may be moved to other sections of the *NESI* documentation, as appropriate.

The middle tier provides process management services such *as* process development, monitoring, and resourcing, that are shared by multiple applications.



Future guidance will include:

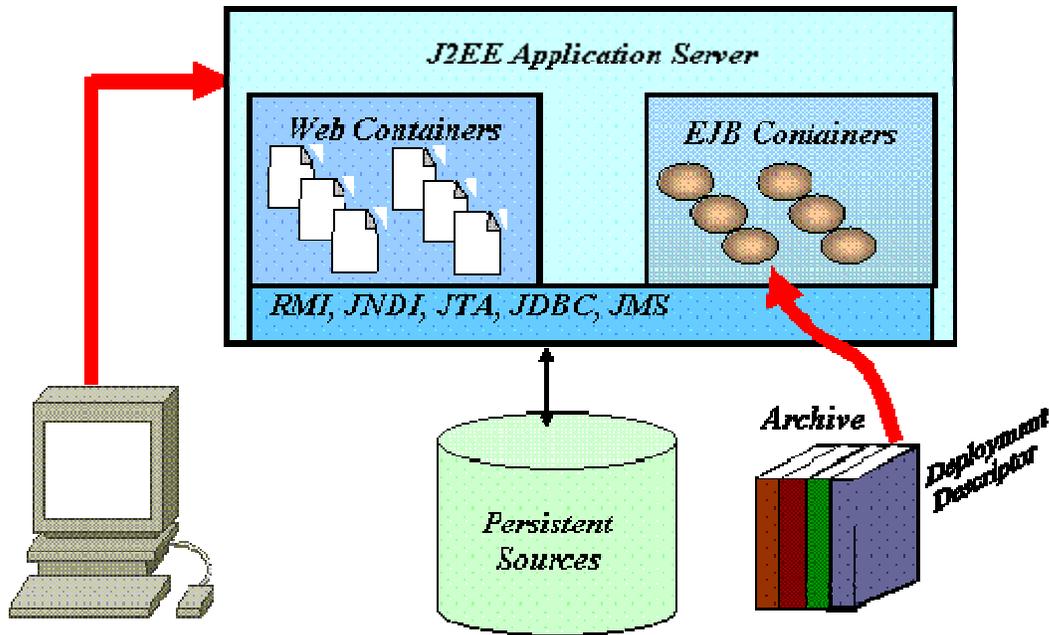
- **Application collaboration/Mediation framework:** Also known as backend integration application communication.
- **Application concurrency control:** Concurrency and locking strategies and guidelines for applications required to operate in a multi-user environment. Transactional strategies for operations with other services in the enterprise.

- **Application server guidelines:** Sybase application server topics, transactions, and data access guidelines.
- **Connection strategies:** Applications written in or using Fortran, Ada, C/C++, Cold Fusion, Java, *J2EE*, Microsoft Office, and *.NET*.
- **CORBA: *Real-time*** topics, cross-vendor interoperability issues, enterprise connection strategies, and Software Communication Architecture (*SCA*) issues.
- **Design patterns and examples:** Recommended patterns and implementations.
- **Microsoft component model:** .NET, COM/DCOM, COM+, security, and data-access guidelines.
- **Microsoft Office:** Connector strategies to and from the enterprise.
- **Middleware guidelines:** Guidelines on developing connectors to and from the enterprise.
- **Other application server operations:** JBoss, Orion, Sybase EAServer.
- **Security guidelines:** Authentication schemes, secure coding practices, digital certificates, digital signatures, firewall policies, protection mechanisms, and *SSL*.
- **Transactional strategies:** For operations with other services in the enterprise.
- **Web services: *UDDI*** operations and taxonomies.

Note that this guidance may be moved to other sections of the *NESI* documentation, as appropriate.

## J2EE environment

In the *J2EE* environment there are multiple deployment descriptors that correspond to the type of modules being deployed. The deployment descriptors are contained in the enterprise archive file (*EAR*). A deployment descriptor is an *XML* file that is inside an archive. It describes the contents of the archive and explicit instructions on how the application server's J2EE-compliant container needs to be configured to run the application.



The table below shows the J2EE standard deployment descriptor files and the specific applications to which they apply. See <http://java.sun.com/dtd/> for details of each XML file.

Component or Application	Scope	Deployment descriptors	Packaging archives
Web application	J2EE	web.xml	.war
Enterprise bean	J2EE	ejb-jar.xml	.jar
Resource adapter	J2EE	ra.xml	.rar
Enterprise application	J2EE	application.xml	.EAR
Client application	J2EE	application-client.xml	

In the J2EE environment, packaging and deployment is done using an archive file. An archive file is a self-contained module that contains all of an application's Java-class files, static files, and deployment descriptor files. Archive files are created using a jar utility.

The format for a deployment descriptor is defined in both the **EJB** specification and the servlet specification. The Sun standards are defined at the following locations:

**J2EE environment applications**      <http://java.sun.com/products/ejb/docs.html>

**Non-J2EE or standard web applications**      <http://java.sun.com/products/servlet/download.html>

**Note:** Some vendors have extensions to these guidelines or have specific descriptors for their products. Refer to the vendor's site for these details. For example, the WebLogic BEA Application Server descriptors are available at [http://e-docs.bea.com/wls/docs70/webapp/WebLogic\\_xml.html](http://e-docs.bea.com/wls/docs70/webapp/WebLogic_xml.html).

## Guidance

- Document the use of non-J2EE-defined deployment descriptors, and explain how the deployment descriptors are used. [G1078]
- J2EE applications should isolate application configuration data values into the deployment descriptor.[G1079]
  - Define all external resources using a separate resource-ref element for each resource. [G1200]
  - Define configuration data such as environment variables, parameters, and properties using resource-env-ref elements. [G1201]

## Best practices

- When deploying a new application to a WebLogic application server (e.g., ear, war, rar), do not edit the WebLogic startup file to add application-specific information. This file is used for server startup only and should not contain application-specific logic. The system administrator must approve and coordinate all updates to this file. [BP1076]
- Do not edit the **config.xml** file manually. The **config.xml** file is the persistent store used by the WebLogic server to store runtime configuration parameters. Instead, use the WebLogic management console to configure the WebLogic server. Any edits done through the management console will be written to **config.xml**. [BP1077]

## Examples

### Environment entries

Bean environment values are defined in the deployment descriptor using the env-entry element. Use J2EE provider utilities to modify these values during or after deployment.

```
<env-entry>
  <env-entry-name>minQueueBuffer</env-entry-name>
```

```

    <env-entry-type>java.lang.Integer</env-entry-type>
    <env-entry-value>12</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>maxQueueBuffer</env-entry-name>
    <env-entry-type>java.lang.Integer</env-entry-type>
    <env-entry-value>120</env-entry-value>
</env-entry>

```

A bean can access the environment entries with a similar code to the following:

```

InitialContext initialContext
    = new InitialContext();
Context myenv
    = initialContext.lookup
      ( "java:comp/env" );
Integer minQueueBuffer
    = (Integer) myenv.lookup("minQueueBuffer ");
Integer maxQueueBuffer
    = (Integer) myenv.lookup("maxQueueBuffer ");
. . .

```

## Resource references

Use resource references to define and use environment entries. By default, the initial J2EE environment context is `java:comp/env/`. Consequently, it is best to classify all resources into subcontexts of the default. For example, classify all **JDBC** definitions using the default context with a `jdbc` subcontext appended to it. For example:

```
java:comp/env/jdbc
```

In the standard deployment descriptor, the declaration of a resource reference to a JDBC connection factory is:

```

<resource-ref>
    <res-ref-name>jdbc/JTMS</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>

```

And the bean accesses the data source as in the following:

```

InitialContext initialContext
    = new InitialContext();
DataSource dataSource
    = initialContext.lookup
      ( "java:comp/env/jdbc/JTMS" );

```

## Resource environment references

The `resource-env-ref` describes administered objects, as opposed to objects that are better maintained programmatically. Administered objects help define objects that are likely to change between implementations: for example, **JMS** or database implementations. It is best to administer these objects along with other administrative tasks that vary from provider to provider and not within the application. This makes the code more portable.

```

<resource-env-ref>
    <resource-env-ref-name>jms/ConnectionFactory</resource-env-ref-
name>
    <resource-env-ref-type>javax.jms.Queue</resource-env-ref-type>
</resource-env-ref>

```

The code to access the administered object follows:

```

InitialContext initialContext
    = new InitialContext();
ConnectionFactory connectionFactory = (ConnectionFactory)
    ctx.lookup("jms/ConnectionFactory");

```

## Example deployment descriptors

### ***ejb-jar.xml***

```

<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>TestClient</ejb-name>
      <home>TestClientHome</home>
      <remote>TestClient</remote>
      <ejb-class>MyLogicBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
  .
  .
  .
</ejb-jar>

```

### ***web.xml***

```

/* Descriptor for Application named: HelloWorld.jsp */
MyWebApp/ (public directory)
  HelloWorld.jsp
WEB-INF/
  Web.xml
  Classes/myBean
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <display-name>HelloWorldJSP</display-name>
  <servlet>
    <servlet-name>HelloWorld</servlet-name>
    <display-name>HelloWorld</display-name>
    <jsp-file>/HelloWorld.jsp</jsp-file>
  </servlet>
  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
  <ejb-ref>
    <ejb-ref-name>ejb/helloejb</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>HelloHome</home>
    <remote>Hello</remote>
  </ejb-ref>
</web-app>
  Contact.class

```

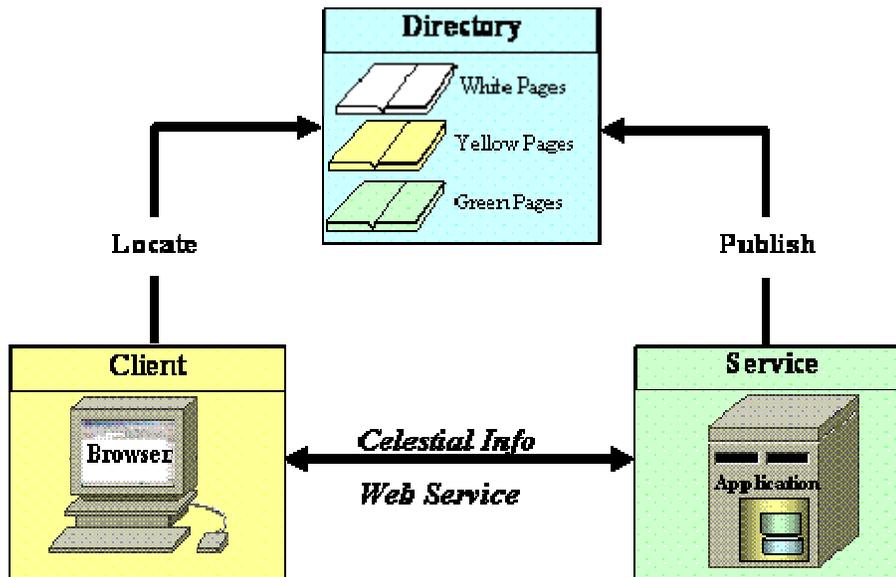
## References

- J2EE - <http://java.sun.com/j2ee/>
- EJB - <http://java.sun.com/products/ejb/>

- .jar - <http://java.sun.com/developer/Books/javaprogramming/JAR/>
- .war - <http://java.sun.com/webservices/docs/1.0/tutorial/doc/WebApp3.html>
- .ear - [http://java.sun.com/j2ee/tutorial/1\\_3-fcs/doc/Overview4.html](http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Overview4.html)
- .rar - [http://java.sun.com/j2ee/tutorial/1\\_3-fcs/doc/Connector2.html](http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Connector2.html)

A web service is an application that exists in a distributed environment, such as the *Internet*. A web service accepts a request, performs its function based on the request, and returns a response. The request and the response can be part of the same operation, or they can occur separately, in which case the consumer does not need to wait for a response. Both the request and the response usually take the form of XML, use a portable data-interchange format called SOAP, and are delivered over a wire protocol, such as *HTTP*.

Web services can reside on top of existing legacy applications and expose services to the net. The web services architecture illustrated below implements the *service-oriented architecture* pattern. For more information on design patterns, see *Web Service Patterns: Java Edition* by Paul B. Monday.

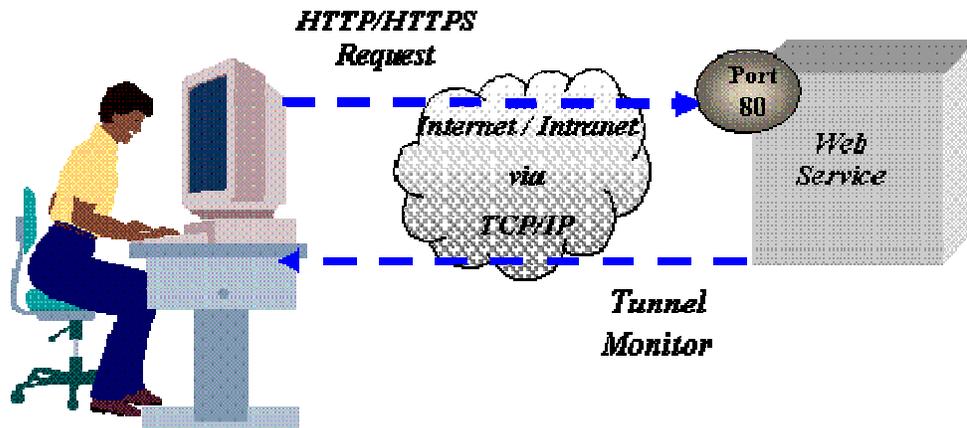


## Web service models

Web services have traditionally been used to connect people to services. However, as the web-service infrastructure has matured, a new model has emerged, the service-to-service model.

### Traditional model

In a classic web service, a request is usually made to a web service using a browser. The request is submitted to the web service using HTTP or HTTPS over the Internet or an *intranet*. The web service processes the request and returns an HTML page that can be displayed in a browser.

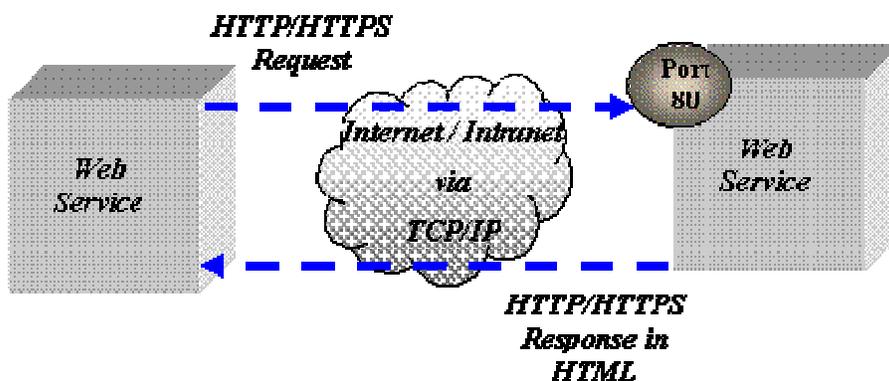


A classic web service has the following characteristics:

- Web page appears via a *browser*
- Connection is via *TCP/IP*
- Transport is *HTTP/HTTPS*
- Message format is *HTML*

### Service-to-service model

*Application servers* used to be responsible for providing machine-to-machine services. Now *web servers* can handle similar work. The web server can pass a request as an XML payload embedded in a TCP/IP and HTTP request, process the data, and respond. The response is typically in the form of an HTML web page or an XML payload that a client application can use.



Machine-to-machine web services have the following characteristics:

- Two independent applications
- Two independent servers
- Connection is via TCP/IP
- Transport is HTTP (port 80)

- Message format is XML payload in SOAP format

## **Key characteristics**

Some key characteristics of web services are:

- High-overhead interactions; may be too heavy for some applications
- Loosely coupled collaborators (e.g., client/server)
- Multiple layers of parsing, marshalling, and unmarshalling
- Non-standard content
- Standard interaction protocol
- No support for services such as messaging and security
- Infant technology
- No support for pass-by-reference

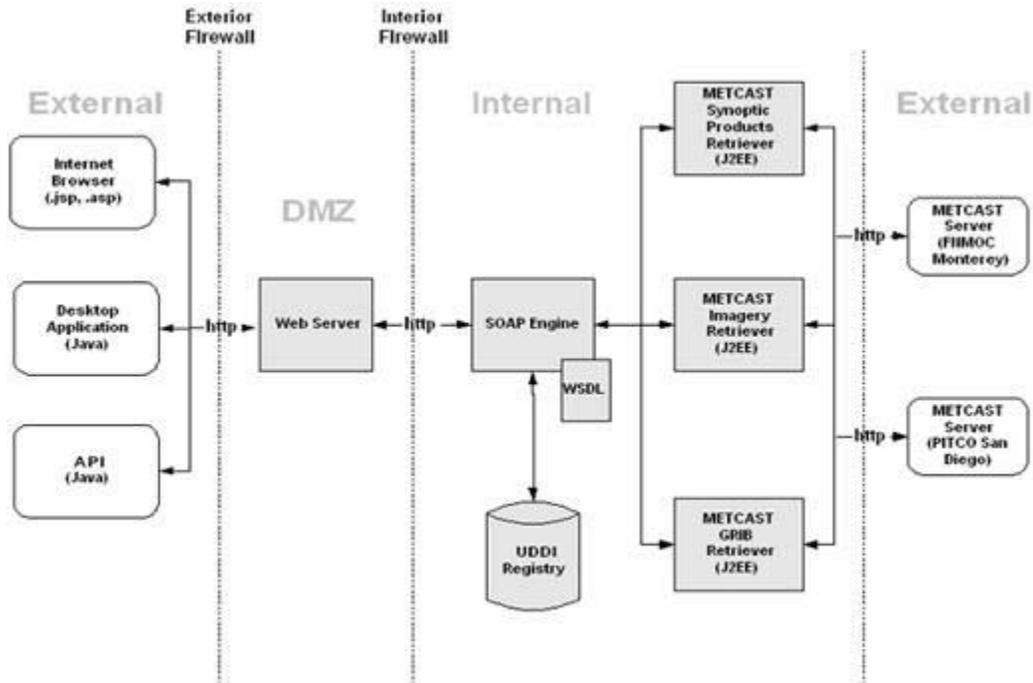
## Guidance

- *WSDL* files used to describe web services must be validated. [G1087]
- Use isolation design patterns such as *façade*, *proxy*, or *adapter* to isolate the application from the connection and manipulation of SOAP messages. [G1088]
- Do not hard-code a web service's *endpoint*. [G1090]

# Examples

## Navy operational example: Exposing web services for METOC

The following figure shows a simplified example of the architecture, illustrating a METOC metcast application that uses SOAP as a proxy to legacy content.



## References

- SOAP definition - <http://sbc.webopedia.com/TERM/S/SOAP.html>
- Web Service Definition Language (WSDL) - <http://www.w3.org/TR/wsdl>
- Adapter pattern - <http://c2.com/cgi/wiki?AdapterPattern>
- Design patterns: Proxy - <http://www.dofactory.com/Patterns/PatternProxy.aspx>
- Façade pattern - <http://c2.com/cgi/wiki?FacadePattern>

**.NET** web services use ASP.NET to expose the middle tier's **API** via **SOAP**. .NET web services also support the **WSDL** 1.1 specification and uses a WSDL document to describe it. In this case, however, the WSDL document contains an **XML namespace** that uniquely identifies the web service's endpoints. .NET provides:

- A client-side component that lets an application invoke web service operations described by a WSDL document.
- A server-side component that maps web service operations to COM-object method calls as described by a WSDL and a Web Services Meta Language (WSML) file, which is needed for Microsoft's implementation of SOAP.

## References

- For information on .NET vs. **J2EE** web services, see <http://www.webservicesarchitect.com/content/articles/hanson01.asp>.

# SOAP

**SOAP** is an **XML** message-based protocol. It uses HTTP to send text commands to web services across the internet. SOAP is lighter weight and requires less programming than similar protocols such as CORBA and DCOM. The extensible messaging framework is independent of programming models and other implementation-specific semantics.

The World Wide Web Consortium provides this description of SOAP:

*SOAP Version 1.2 (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics.*

Two major design goals for SOAP are simplicity and extensibility. SOAP attempts to meet these goals by omitting distributed-system features from the messaging framework. Such features include but are not limited to reliability, security, correlation, routing, and Message Exchange Patterns (MEPs). While it is anticipated that many features will be defined, this specification provides specifics only for two MEPs. Other features are left to be defined as extensions by other specifications.

## Key characteristics

- SOAP is **RPC**-based. It offers an XML-RPC with extensibility mechanisms; for instance, it allows schemas to define types.
- SOAP is an XML document.
- SOAP is text-based, providing a standard mechanism for passing through firewalls via the HTTP ports.
- There are many SOAP language bindings, and new ones are frequently announced.
- SOAP is a wire protocol and does not have an activation mechanism. It is inherently stateless.
- SOAP does not implement security.
- SOAP is case-sensitive and white-space-sensitive.

## Message formats

### Message styles

The W3C's **WSDL** 1.1 Specification identifies two message styles: Document and RPC. The purpose of the styles determines how the content of the SOAP message body is formatted.

<b>Document</b>	<p>The SOAP Body contains one or more child elements called parts. There are no SOAP formatting rules for what the SOAP Body contains; it contains whatever the sender and the receiver agree upon.</p> <p><b>Note:</b> There is a Wrapped form of this style that is required to interoperate with Microsoft web services using Document style.</p>
-----------------	--

	There is no specification that defines this style.
<b>RPC</b>	RPC implies that SOAP Body contains an element with the name of the method or remote procedure being invoked. This element in turn contains an element for each parameter of that procedure.

**Serialization formats**

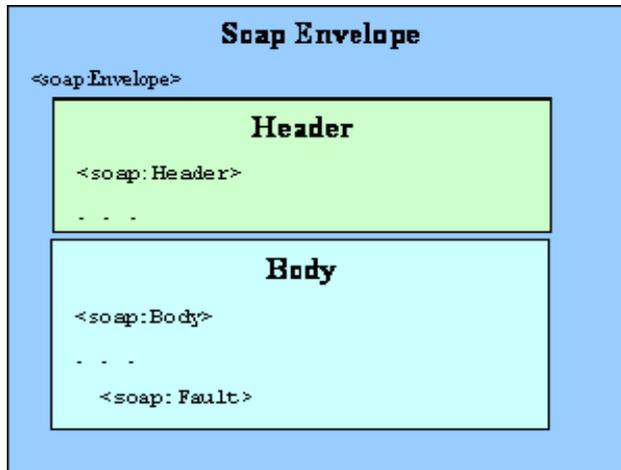
For applications that use serialization/deserialization to abstract away the data wire format, there is one more choice to be made: the serialization format. The following table describes the two most popular serialization formats today.

<b>SOAP encoding</b>	SOAP encoding uses a set of rules to serialize the data transferred between the client and the server. The rules are defined in section 5 of the WSDL 1.1 Specification. These rules are also referred to as “section 5 encoding.” The rules specify how to serialize objects, structures, arrays, and object graphs and directly use the predefined XML Schema data types. Generally, an application using SOAP encoding should use the RPC message style.
<b>Literal</b>	Data is serialized according to an independent external schema. There are no preset rules for serializing objects, structures, and graphs, etc. in the literal encoding style. The industry is overwhelmingly embracing <b>W3C</b> XML schemas.

**Note:** Document style can be interpreted as either an XML string or as a W3C **DOM** Document Element. Microsoft has a technique called Wrapped that encapsulates the information being exchanged, regardless of the style.

**Structure**

A SOAP message comprises three parts: an envelope, an optional header, and a required body. The envelope encapsulates the other two elements. The optional header contains one or more header elements that contain meta-information about the method calls.



## Envelope

is the root of the soap request. At a minimum, it defines the SOAP namespace for SOAP 1.2. The envelope may define additional namespaces.

## Header

contains auxiliary information as SOAP blocks, such as authentication, routing information, or transaction identifier. The header is optional.

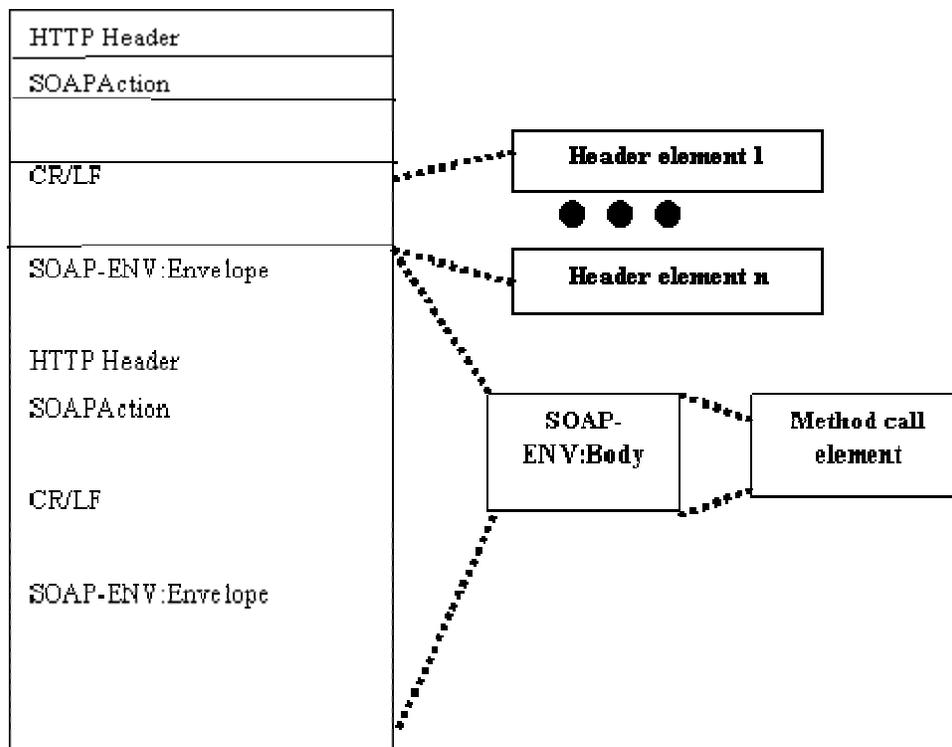
## Body

contains the main information in one or more SOAP blocks; for example, a SOAP block for RPC call. The body is mandatory and it must appear after the header.

## Fault

is a special block that indicates protocol-level error. If present, it must appear within a Body element.

The SOAP payload is encapsulated within the SOAP envelope, which is part of the HTTP payload. The following figure shows an HTTP payload that contains a SOAP message.



## Guidance

- Use the document literal style for all data transferred using SOAP where the document is a W3 Organization's Document Object Model (DOM). [G1082]
- Documents transferred using SOAP should be validated by an **XSD** defined by the Community of Interest (COI). [G1084]

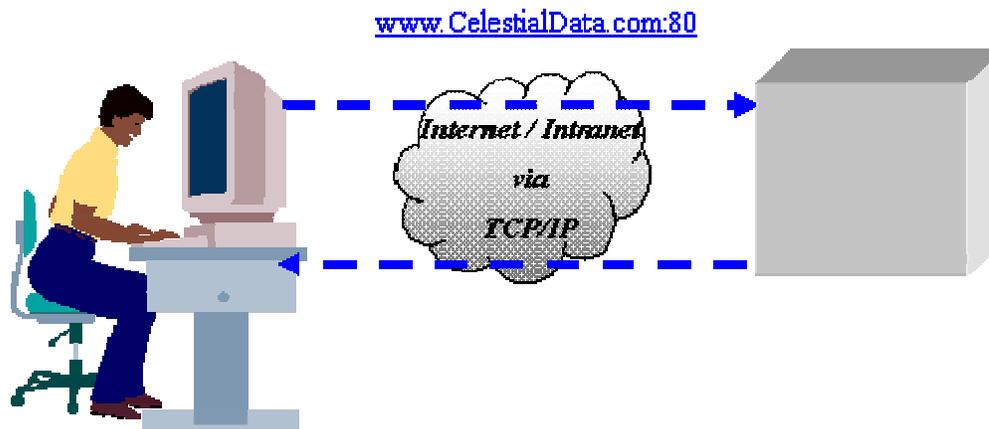
- Use isolation design patterns such as façade, proxy, or adapter to isolate the application from the connection and manipulation of SOAP messages. [G1088]
- Web services must handle SOAP exceptions and SOAP faults. [G1093]
- Use W3C fault codes for all SOAP faults. [G1095]

## Examples

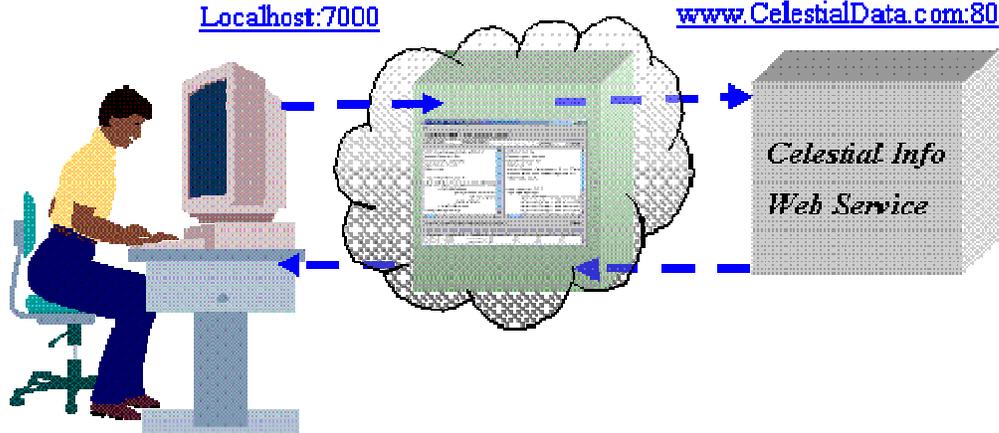
The following is an example of a web service client requesting celestial information about a particular location and receiving the results. Both the request and the response are made using the WSI document literal style of send and receiving XML SOAP messages.

These listings are the results of using a tunnel monitoring utility called NetTool available from the SourceForge site <http://sourceforge.net/projects/nettool/>. The Tunnel monitoring tool basically interjects itself between the web service client and the web service producer. The client connects to the tunnel monitor instead of connecting directly to the producer. The tunnel tool then displays or logs the traffic and forwards it onto the producer. The producer returns the response to the tunnel tool monitor. The response is also displayed or logged and forwarded back to the client.

### Without Tunnel Monitor



## With Tunnel Monitor



## Request

```
POST /DocClientWebProject/BeaServers/CelestialInfoDocDoc.jws HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related,
text/*
User-Agent: Axis/1.1
Host: 192.168.2.8:7003
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 597
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <in0
      xsi:type="ns1:Document"
      xmlns="urn:CelestialInfoDocDoc"
      xmlns:ns1="http://xml.apache.org/xml-soap">
      <DocumentRequestData xmlns="">
        <city>San Diego</city>
        <stateOrProvince>California</stateOrProvince>
        <country>USA</country>
        <documentName>CelestialInfoRpt</documentName>
      </DocumentRequestData>
    </in0>
  </soapenv:Body>
</soapenv:Envelope>
```

## Response

```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2004 17:53:55 GMT
Pragma: no-cache
```

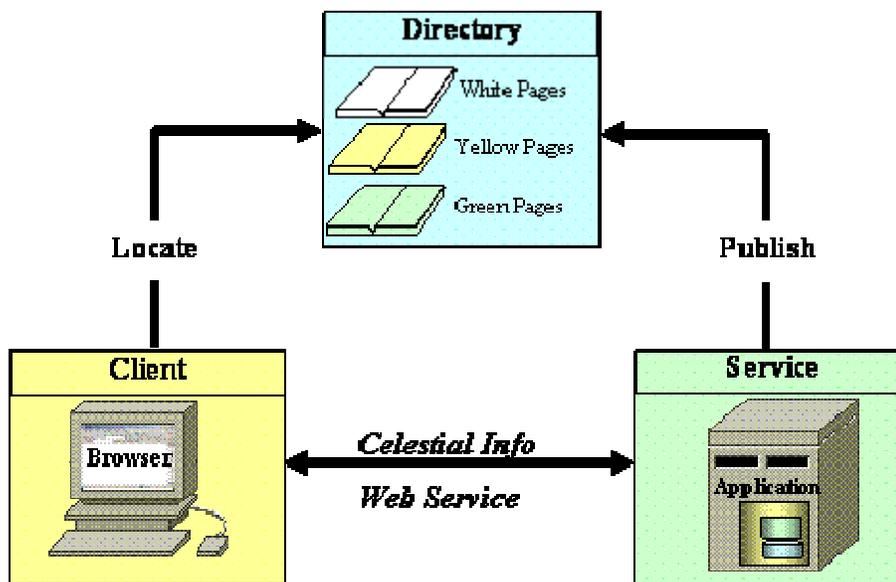
```
Server: WebLogic Server 8.1 SP3 Tue Jun 29 23:11:19 PDT 2004 404973
WebLogic Server 8.1 SP3 Tue Jun 29 23:11:19 PDT 2004 404973 WebLogic
Server 8.1 SP3 Tue Jun 29 23:11:19 PDT 2004 404973
Content-Length: 647
Content-Type: text/xml; charset=UTF-8
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Cache-Control: no-cache
Connection: Close
```

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns:getCelestialInfoReturn xmlns:ns='urn:CelestialInfoDocDoc'>
      <CelestialInfoRpt>
        <description>DOC-DOC: Results returned from :
RAPIDS14(192.168.2.8)
        </description>
        <moonrise>2004-07-12 1:59 AM PDT</moonrise>
        <moonset>2004-07-12 4:22 PM PDT</moonset>
        <sunrise>2004-07-12 5:50 AM PDT</sunrise>
        <sunset>2004-07-12 7:58 PM PDT</sunset>
      </CelestialInfoRpt>
    </ns:getCelestialInfoReturn>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Web services

A web service is an application that exists in a distributed environment, such as the *Internet*. A web service accepts a request, performs its function based on the request, and returns a response. The request and the response can be part of the same operation, or they can occur separately, in which case the consumer does not need to wait for a response. Both the request and the response usually take the form of XML, use a portable data-interchange format called SOAP, and are delivered over a wire protocol, such as *HTTP*.

Web services can reside on top of existing legacy applications and expose services to the net. The web services architecture illustrated below implements the *service-oriented architecture* pattern. For more information on design patterns, see *Web Service Patterns: Java Edition* by Paul B. Monday.

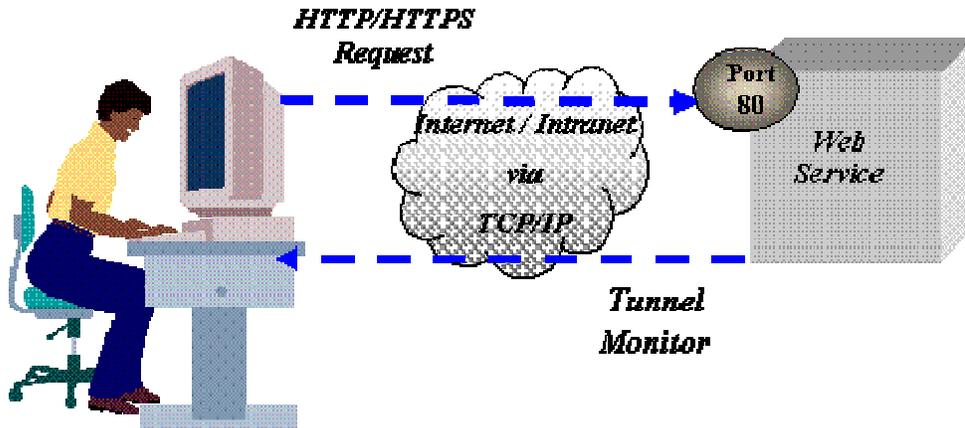


### Web service models

Web services have traditionally been used to connect people to services. However, as the web-service infrastructure has matured, a new model has emerged, the service-to-service model.

#### *Traditional model*

In a classic web service, a request is usually made to a web service using a browser. The request is submitted to the web service using HTTP or HTTPS over the Internet or an *intranet*. The web service processes the request and returns an HTML page that can be displayed in a browser.

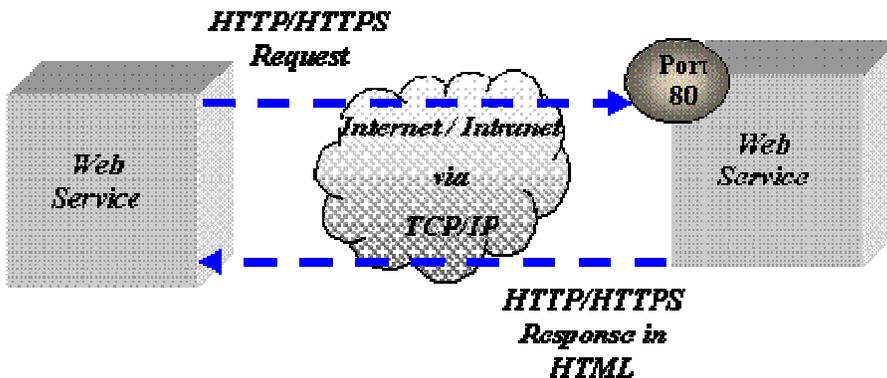


A classic web service has the following characteristics:

- Web page appears via a *browser*
- Connection is via *TCP/IP*
- Transport is *HTTP/HTTPS*
- Message format is *HTML*

### Service-to-service model

*Application servers* used to be responsible for providing machine-to-machine services. Now *web servers* can handle similar work. The web server can pass a request as an XML payload embedded in a TCP/IP and HTTP request, process the data, and respond. The response is typically in the form of an HTML web page or an XML payload that a client application can use.



Machine-to-machine web services have the following characteristics:

- Two independent applications
- Two independent servers
- Connection is via TCP/IP
- Transport is HTTP (port 80)

- Message format is XML payload in SOAP format

## Key characteristics

Some key characteristics of web services are:

- High-overhead interactions; may be too heavy for some applications
- Loosely coupled collaborators (e.g., client/server)
- Multiple layers of parsing, marshalling, and unmarshalling
- Non-standard content
- Standard interaction protocol
- No support for services such as messaging and security
- Infant technology
- No support for pass-by-reference

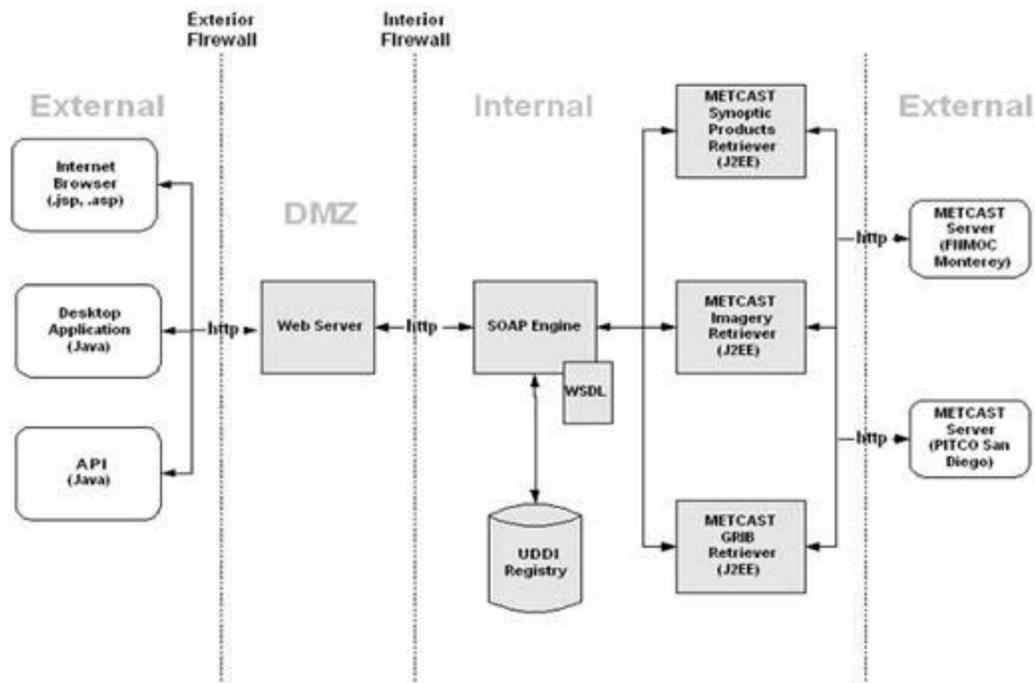
## Guidance

- **WSDL** files used to describe web services must be validated. [G1087]
- Use isolation design patterns such as **façade**, **proxy**, or **adapter** to isolate the application from the connection and manipulation of SOAP messages. [G1088]
- Do not hard-code a web service's **endpoint**. [G1090]

## Examples

### Navy operational example: Exposing web services for METOC

The following figure shows a simplified example of the architecture, illustrating a METOC metcast application that uses SOAP as a proxy to legacy content.



## References

- SOAP definition - <http://sbc.webopedia.com/TERM/S/SOAP.html>
- Web Service Definition Language (WSDL) - <http://www.w3.org/TR/wsdl>
- Adapter pattern - <http://c2.com/cgi/wiki?AdapterPattern>
- Design patterns: Proxy - <http://www.dofactory.com/Patterns/PatternProxy.aspx>
- Façade pattern - <http://c2.com/cgi/wiki?FacadePattern>

# .NET framework

## Web services

**.NET** web services use ASP.NET to expose the middle tier's **API** via **SOAP**. .NET web services also support the **WSDL** 1.1 specification and uses a WSDL document to describe it. In this case, however, the WSDL document contains an **XML namespace** that uniquely identifies the web service's endpoints. .NET provides:

- A client-side component that lets an application invoke web service operations described by a WSDL document.
- A server-side component that maps web service operations to COM-object method calls as described by a WSDL and a Web Services Meta Language (WSML) file, which is needed for Microsoft's implementation of SOAP.

## References

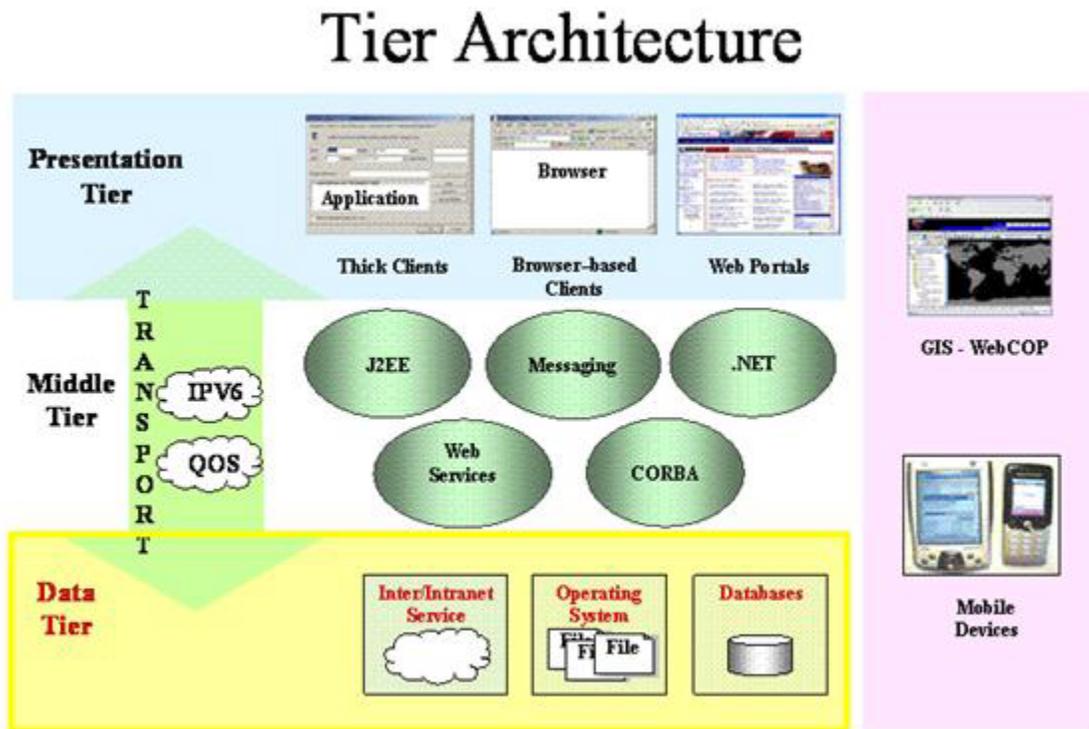
- For information on .NET vs. **J2EE** web services, see <http://www.webservicesarchitect.com/content/articles/hanson01.asp>.



---

# Data tier

The data tier is responsible for storing data. It does not (should not) contain any business logic, and handles only that processing required to access data and maintain its integrity.



Future guidance will include:

- **Database topics:** Lessons learned from Oracle and Sybase, *replication* across database vendors, and database federation concepts.
- **Design patterns and examples:** Recommended patterns and implementations.
- **Security guidelines:** Authentication schemes, secure coding practices, digital certificates, digital signatures, firewall polices, protection mechanisms, and *SSL*.
- **XML:** Coding guidelines, namespaces, XML parser usage, wrapper classes, XML databases, and XML security guidelines like *SAML*.

Note that this guidance may be moved to other sections of the *NESI* documentation, as appropriate.

Most modern multi-tiered systems need to collect, store, retrieve and manage persistent data. This data persistence is the responsibility of the data tier. In essence, the data tier functionality is accomplished with modern *COTS* Database Management Systems (DBMSs) such as MySQL, Oracle, *SQL* Server, or Sybase Adaptive Server Enterprise (ASE).

## Decouple databases from applications

To promote database independence, access the database only through *open-standards* interfaces. The goal is to swap out data sources and/or connect to multiple data sources without affecting the application or increasing software maintenance costs. Data-level adapters allow applications to access data through database calls that are native to the requesting application. At this point, the *business logic* can be shared with other data sources. This positions the application to move business logic from the database to the middle tier, to support database independence.

### Guidance

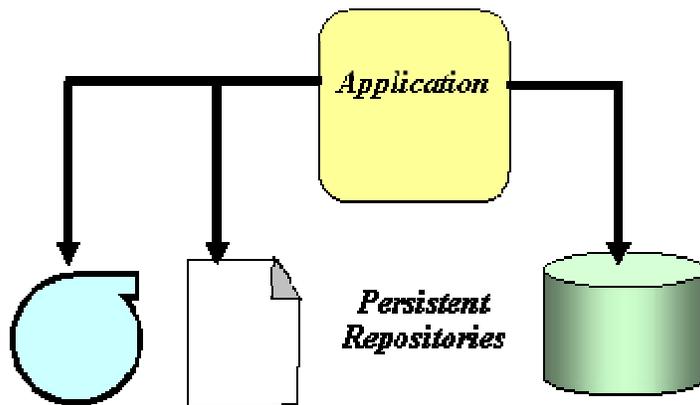
- Access the database only through open-standards interfaces, to promote database independence. [G1014]
  - For Java, use *JDBC*. [G1211]
  - For C/C++ and .NET use *ODBC*. [G1212]

# Database implementations

The data tier is simply a repository for persistent data. There are many ways that data can be persisted:

- *OS file systems*
- *Hierarchical databases*
- *Object-oriented databases*
- *Niche databases*
- *Native XML databases*
- *Relational databases*

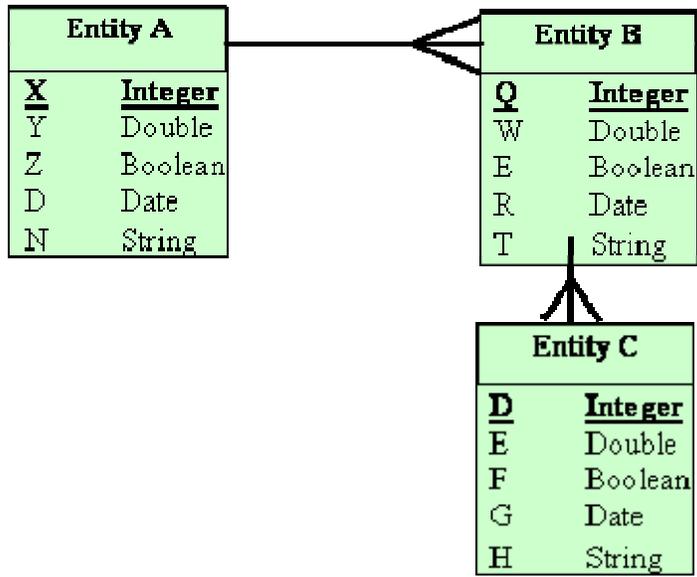
COTS *DBMSs* are mature technical products, the capabilities of which are being continually expanded to adapt to and accommodate new technologies.



## Guidance

- Implement the data tier using readily available COTS *RDBMS* products that are standards-based and provide a rich set of generic capabilities such as ad-hoc queries, backup, recovery, and indexes. [G1132]

Modeling is an essential step in understanding the data that will comprise a system. Before implementing a system, it is important to understand the basic data elements and the relationships of the elements. The end products of *data modeling* can be *XML* schemas or RDBMS schema definitions.



The following guidance applies to the data modeling phase of the data tier.

## Guidance

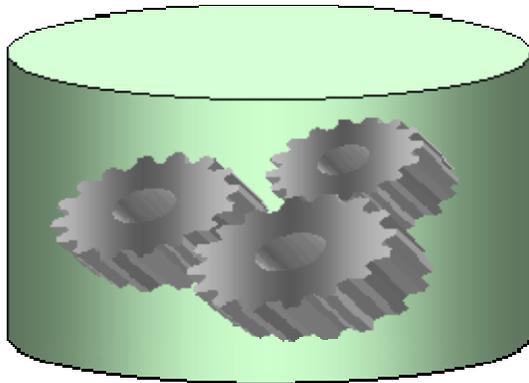
- Use standard data models developed by *Communities of Interest (COI)* as the basis of program or project data models. [G1141]
- Develop a two-level database model; one level captures the *conceptual* or logical aspects, and the other level captures the *physical* aspects. [G1144]
- The conceptual/logical data model should contain information necessary to generate a data dictionary. [G1146]
- *Domain analysis* should define the constraints on input data validation. [G1147]
- The conceptual/logical model should be *normalized*. [G1148]

## Best practices

- Use a database modeling tool that supports a two-level model (Conceptual/Logical and Physical) and ISO-11179 data-exchange standards. [BP1143]
- Conceptual and logical models should be vendor-neutral whenever possible. [BP1145]

# RDBMS internals

A **RDBMS** is a collection of data items organized as a set of formally-described tables. You can access and reassemble data in many different ways without having to reorganize the database tables. It is important to ensure data quality and to access data quickly, using simple, easily understood dynamic queries. Towards these ends, an RDBMS offers such services as **triggers**, **stored procedures**, indices, constraints, **referential integrity**, efficient storage, and **high availability** features.



## Guidance

- Define declarative **foreign keys** for all relationships between tables. [G1151]
- Use stored procedures for operations that are focused on the insertion and maintenance of data. [G1154]
- Use triggers primarily to enforce referential integrity or data integrity and not to perform complex business logic. [G1155]

## Best practices

- Follow a naming convention [BP1248]
  - Do not use generic names for database objects such as databases, schema, users, tables, views, or indices. [BP1249].
  - Use case-insensitive names for database objects such as databases, schema, users, tables, views, and indices. [BP1250]
  - Separate words with underscores. [BP1251]
  - Do not use names longer than 30 characters. [BP1252]
  - Do not use the SQL:1999 reserved words as names for database objects such as databases, schema, users, tables, views, or indices. [BP1253]
  - For command and control systems, use the names defined in the **C2IEDM** for data exposed to the outside communities. [BP1254]
- Use surrogate keys. [BP1255]

- Use surrogate keys as the *primary key*. [BP1256]
  - Place a unique key constraint on the natural key fields. [BP1257]
- All data that are transferred using *XML* should explicitly define the encoding style. [BP1258]
- Use indexes. [BP1259]
  - All tables should have a primary key defined. This is generally enforced via an underlying index. [BP1260]
  - Monitor and tune indexes according to the response time during normal operations in the production environment. [BP1261]
  - In the case of Oracle, define indexes against the FK columns to avoid contention and locking issues. [BP1262]
- Gather storage requirements in the planning phase, and allocate twice the estimated storage space. [BP1263]
- To obtain high availability, use hardware solutions when geographic proximity permits. [BP1264]

# XML

*XML* is a popular new technology that many developers are capitalizing on. For general guidance, use one of the many XML developer's guides available.

This section focuses on interfacing with other applications and enterprise components. It contains the following topics:

- *Wrapping XML parsers*
- *Parsing XML strategies*

## References

- For information on XML schemas and repositories, see <http://diides.ncr.disa.mil/mdregHomePage/mdregHome.portal>.
- For information on the Department of the Navy's XML policies, see <http://quickplace.hq.navy.mil/navyxml> or contact Bob Green, Office of the *DON* CIO, [robert.a.green2@navy.mil](mailto:robert.a.green2@navy.mil).

## Wrapping XML parsers

Wrapping the *parser* promotes interoperability with other systems by reducing coupling and minimizing the impact of enterprise change on the applications.

The enterprise will publish an *API* wrapper to an *XML* parser and an *XSLT* processor. All applications using XML will use these wrapper classes. When they are available, you will be able to download them from the *NESI* open-source site.

## Examples

### *Sample wrapper class*

This figure shows a sample wrapper class for an XML parser:

```
import java.io.*;
import org.w3c.dom.*;
import java.util.*;
import javax.xml.parsers.*;
public class XMLWrapper
{
    private Document document;
    public void initialize( )
    { try
      { System.setProperty
        ( "javax.xml.parsers.DocumentBuilderFactory",
          "org.apache.xerces.jaxp.DocumentBuilderFactoryImpl"
        );
        System.setProperty
        ( "javax.xml.parsers.SAXParserFactory",
          "org.apache.xerces.jaxp.SAXParserFactoryImpl"
        );
        DocumentBuilderFactory dbf
```

```

        = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        document = db.newDocument();
    } // End try
    catch ( DOMException domex )
    { domex.printStackTrace();
    } // End catch DOMException
    catch ( ParserConfigurationException pcex )
    { pcex.printStackTrace();
    } // End catch ParserConfigurationException
} //end init
//public API's
public Node setRootNode
    ( String rootElement )
{
    try
    { Node rootNode = document.createElement( rootElement );
      document.appendChild( rootNode );
      return rootNode;
    } // End try
    catch ( DOMException domex )
    { domex.printStackTrace();
    } // End catch DOMException
    return null;
} // End setRootNode
public Node addChild
    ( Node parentNode, String element )
{ parentNode.appendChild
    ( document.createElement ( element ) );
  return parentNode.getLastChild();
} // End addChild
public void addTextNode
    ( Node parentNode, String element )
{ parentNode.appendChild
    ( document.createTextNode( element ) );
} // End addTextNode
public void addCommentNode
    ( Node parentNode, String element )
{ parentNode.appendChild
    ( document.createComment( element ) );
} // End addCommentNode
public void addCommentNodeDoc
    ( String element )
{ document.appendChild
    (document.createComment( element ));
} // End addCommentNodeDoc
public void addPINodeDoc
    ( String target,
      String value
    )
{ document.appendChild
    ( document.createProcessingInstruction
      ( target,
        value
      )
    );
} // End addPINodeDoc

```

```

public void addPINode
    ( Node parentNode,
      String target,
      String value
    )
{ parentNode.appendChild
    ( document.createProcessingInstruction
      ( target,
        value
      )
    );
} // End addPINode
} // End initialize
} //end XMLWrapper

```

### **Sample object**

This figure shows a sample object using the XML parser wrapper:

```

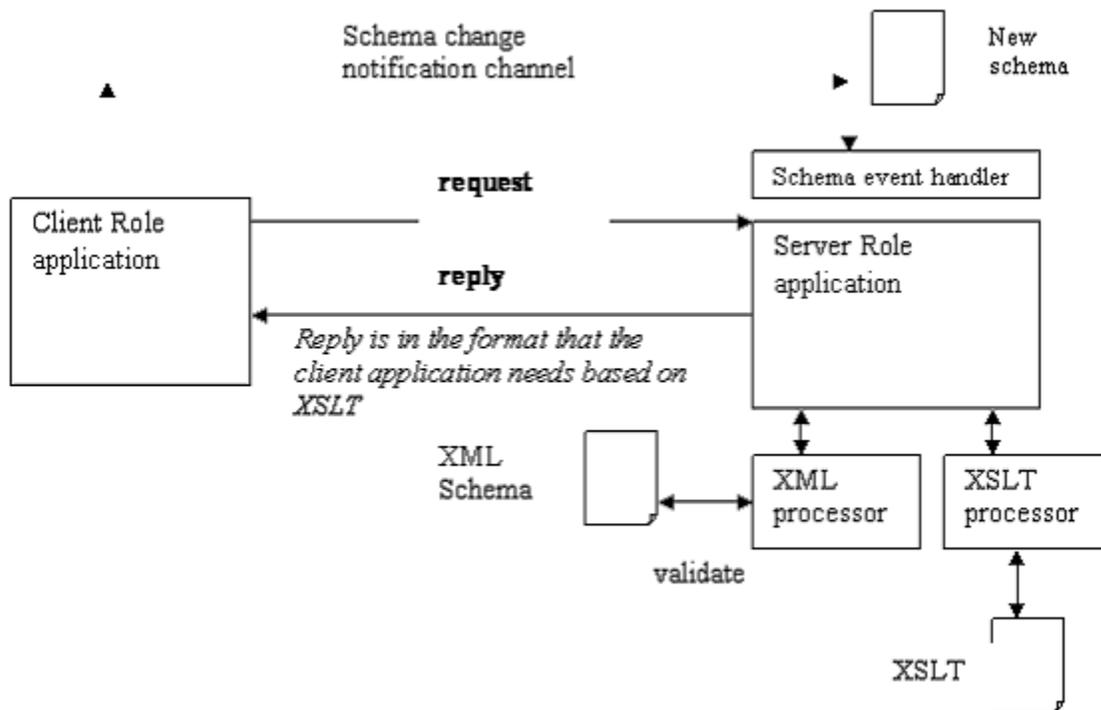
private static void buildXMLDocument()
{ //build up a weather report
  XMLCreator xmlCreator = new XMLCreator();
  xmlCreator.initialize();
  xmlCreator.addCommentNodeDoc
    ( "generate xml from a soap client");
  xmlCreator.addPINodeDoc
    ("xml:stylesheet",
     "type = \"text/xsl\" href = \"weather.xsl\""
    );
  Node weatherNode
    = xmlCreator.setRootNode
      ( "weatherReport");
  xmlCreator.addTextNode
    ( xmlCreator.addChild
      ( weatherNode,
        "location"
      ),
      weatherReport[0]
    );
  xmlCreator.addTextNode
    ( xmlCreator.addChild
      ( weatherNode,
        "wind"
      ),
      weatherReport[1]
    );
  xmlCreator.addTextNode
    ( xmlCreator.addChild
      ( weatherNode,
        "SkyConditions"
      ),
      weatherReport[2] );
  xmlCreator.addTextNode
    ( xmlCreator.addChild
      ( weatherNode,
        "Visibility"
      ),

```

```
        weatherReport[3] );
xmlCreator.addTextNode
( xmlCreator.addChild
  ( weatherNode,
    "Temperature"
  ),
  weatherReport[4] );
xmlCreator.addTextNode
( xmlCreator.addChild
  ( weatherNode,
    "Pressure"
  ),
  weatherReport[5] );
xmlCreator.addTextNode
( xmlCreator.addChild
  ( weatherNode,
    "Humidity"
  ),
  weatherReport[6] );
xmlCreator.addTextNode
( xmlCreator.addChild
  ( weatherNode,
    "Wind2"
  ),
  weatherReport[7] );
weatherDoc = xmlCreator.getDocument();
} //end buildXMLDocument
```

## Parsing XML strategies

Passing *XML* back and forth between systems imposes significant overhead. As more client-side applications use “services,” parsing multiple XML outputs from multiple web services will impact the performance of the client-side application.



## Best practices

- The XML document generator is responsible for validating the XML. [BP1265]

## References

Won Kim. *Introduction to Object-Oriented Databases*. Computer Systems. MIT Press, Cambridge, MA, 1990.

Application Architecture: An N-Tier Approach - Part 1:  
<http://www.15seconds.com/issue/011023.htm>

**SQL:1999**, formerly known as SQL3: <http://dbs.uni-leipzig.de/en/lokal/standards.pdf>

Database Journal: <http://www.databasejournal.com/>

Crossing Chasms Pattern Language Object to RDBMS: <http://c2.com/cgi/wiki?CrossingChasms>

Object Data Management Group (ODMG): <http://www.odmg.org/>

Object Management Group (**OMG**): <http://www.omg.org/>

Native XML database vendors: <http://www.rpbouret.com/xml/XMLDatabaseProds.htm#native>

**C2IEDM** data model specifications: [http://www.mip-site.org/MIP\\_Specifications/Baseline\\_2.0/C2IEDM-C2\\_Information\\_Exchange\\_Data\\_Model/](http://www.mip-site.org/MIP_Specifications/Baseline_2.0/C2IEDM-C2_Information_Exchange_Data_Model/)

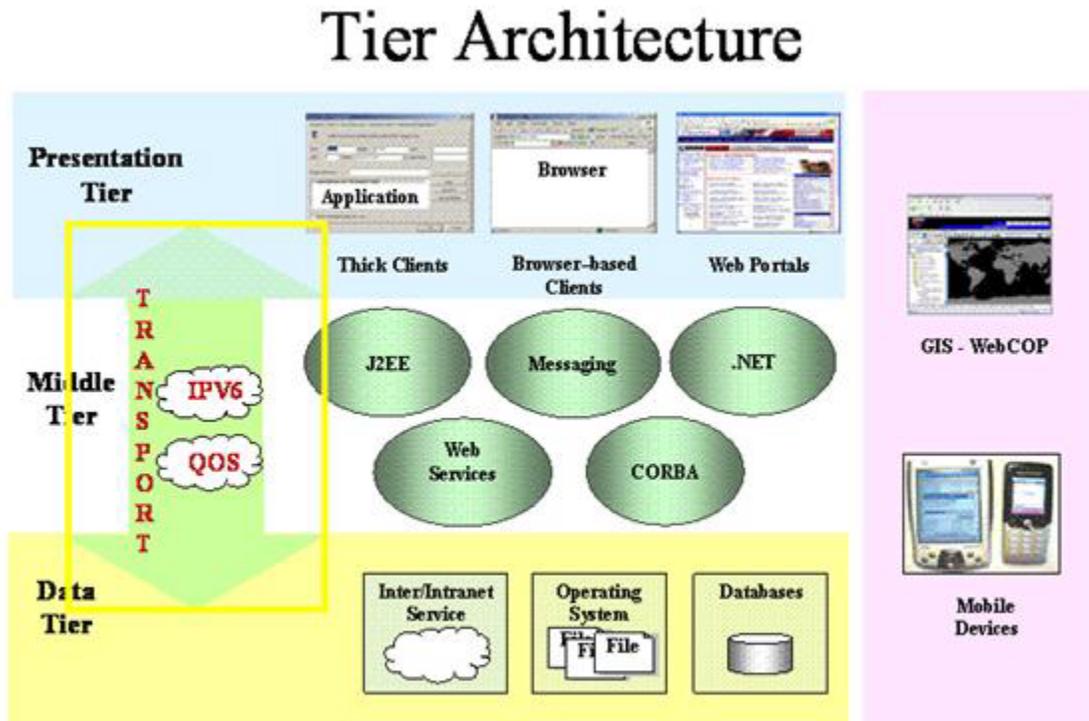


---

# Networks and enterprise services

This section contains information on the following topics:

- *Discovery*
- *Quality of Service*



Future guidance will include:

- **Admin/management framework:** Includes remote monitoring, remote software upgrades, OPS, etc.: all the tools necessary to support content addition, subtraction, and modification.
- **Application instantiation/lifecycle framework:** Describes how applications come and go from the enterprise.
- **Authentication schemes:** Web authentication, enterprise authentication, *PKI*, smart cards, Single Sign-On.
- **Certifications:** SSAAs for services, service IATOs, and ATOs.
- **Connecting to enterprise and *NCES* components**
- **Design patterns and examples:** Recommended patterns and implementations.

- **Directory:** *JNDI*, *LDAP*, Active Directory.
- **Enterprise management**
- **LDAP:** Using LDAP for realms, directory service *replication*, Active Directory connections.
- **Load balancing strategies:** Server configurations to support enterprise services.
- **Mobile code policy:** Guidelines on developing mobile code within the DoD Mobile Code policy.
- **OS setup/configuration:** Lessons learned on various operating systems that the DoD uses.
- **Quality of Service (*QoS*)**
- **Resource management:** Load balancing, caching, and availability strategies.
- **Security:** *XML* and coding standards
- **Security guidelines:** Authentication schemes, secure coding practices, digital certificates, digital signatures, firewall policies, protection mechanisms, and *SSL*.

Note that this guidance may be moved to other sections of the *NESI* documentation, as appropriate.

# Discovery

This section covers:

- *User registries*, which are defined as directory services
- *Service registries*, which are defined as UDDI APIs to RDBMS

Discovery and Directory are a set of repositories and tools for accessing information about people, components, metadata, content, and services from wherever they are, not just from a single machine. These repositories are based on the ability to find people, content, and services using metadata. The Discovery and Directories services perform important but distinct functions in the architecture.

- **Discovery** component provides the repository for published services.
- **Directory** component provides the repository of information concerning people, their roles, organizations, and associated credentials. The directory component is the technology that supports the discovery function.

# Directory

A directory is:

- A service that allows the search of a structured repository of information
- A special-purpose database
- Defined by how users interact with it through its protocol and its APIs
- Used in a context where data are retrieved much more frequently than they are updated
- Not designed to store very large objects, but rather, very large numbers of objects
- Not a relational database

The main differences between a directory and a relational database are:

- Directories are generally intended for environments in which there are more read and search operations than updates.
- Directories do not support the advanced relational queries of a relational database.
- Directories do not support transactional integrity across multiple operations.
- Directories have better support for approximate match searches.
- Directories usually have preconfigured schemas.
- Directory protocols are better suited for WAN use.
- Directories are smaller to maintain and are less expensive.

The rest of this section describes the following protocols and APIs for directories:

- *LDAP*: a protocol for accessing a directory.
- *JNDI*: a layer on top of *LDAP* that enables directories to talk to each other.

## Lightweight Directory Access Protocol (LDAP)

**Lightweight Directory Access Protocol** is an Internet protocol that programs use to look up contact information from a server. LDAP was designed at the University of Michigan to adapt a complex enterprise directory system (called X.500) to the modern Internet. A directory server runs on a host computer on the Internet, and various client programs that understand the protocol can log into the server and look up entries.

LDAP is a hierarchical directory structure accepted through most of the industry. LDAP directories provide a repository for lookup. Security services may also use LDAP directories. Portals and web services use these to maintain user and organizational information, as well as access control and cryptographic certificate information.

### Example: Java JNDI to LDAP

This example shows how Java can use JNDI to search for users. To run this example:

- Install **JDK** 1.4
- Install the Netscape directory server package
- Add JDK and the Netscape package to the classpath.

#### To search for users with JNDI:

1. During Netscape directory server installation, note the configurations for port, username, host, admin user name, and password, as shown in the figures below.





2. Add a user with the Netscape Directory Console.

3. Build a JNDI client that dumps out all the user attributes. The code appears in the sample below.
4. Use a configuration file (properties or *XML*) to store all dynamically configurable settings. Store the following code in a file named **config.properties**.

```
initialContextFactory=com.sun.jndi.ldap.LdapCtxFactory
providerUrl=ldap://localhost:389
contextName=dc=spawar,dc=navy,dc=mil
```

**Note:** In this example, the location of the config file is determined through an command line argument. In a *J2EE* or web environment, these configuration properties would be addressed in a *deployment descriptor*.

## JNDI client

```
import javax.naming.*;
import javax.naming.directory.*;
import java.util.*;
import java.io.*;
public class FindUser
{
    public static void main(String args[])
        throws Exception
    {
        if(args.length < 2)
        {
            System.out.println("Usage: java FindUser <Config File> <User
Name>");
            System.exit(0);
        }
        Properties configSettings = loadConfigurationFile(args[0]);
        if(configSettings == null)
        {
```

```

        System.out.println("Failed to load configuration file: " +
args[0]);
        System.exit(1);
    }
    Hashtable env = new Hashtable();
    env.put(Context.INITIAL_CONTEXT_FACTORY,
configSettings.getProperty("initialContextFactory"));
    env.put(Context.PROVIDER_URL,
        configSettings.getProperty("providerUrl"));
    DirContext ctx = new InitialDirContext(env);
    SearchControls ctls = new SearchControls();
    ctls.setSearchScope(SearchControls.SUBTREE_SCOPE);
    NamingEnumeration results =
ctx.search(configSettings.getProperty("contextName"),
    "cn=" + args[1], ctls);
    while(results.hasMore())
    {
        SearchResult searchResults = (SearchResult)results.next();
        System.out.println(searchResults.getName());
        Attributes attrs = searchResults.getAttributes();
        if(attrs != null)
        {
            for(NamingEnumeration enum = attrs.getAll(); enum.hasMore(); )
            {
                Attribute attrib = (Attribute)enum.next();
                System.out.print(attrib.getID() + "=");
                for(NamingEnumeration e = attrib.getAll(); e.hasMore(); )
                System.out.print(e.next() + " ");
                System.out.println("");
            }
        }
    }
    ctx.close();
}
private static Properties loadConfigurationFile(String filename)
{
    File configFile = new File(filename);
    if(configFile.exists() == false)
    {
        return null;
    }
    else
    {
        try
        {
            Properties config = new Properties();
            config.load(new FileInputStream(configFile));
            return config;
        }
        catch(Exception e)
        {
            return null;
        }
    }
}
}
}

```

## Java Naming & Directory Interface (JNDI)

The **Java Naming and Directory Interface (JNDI)** is an API for directory services in a **J2EE** environment. It allows clients to discover and look up data and objects using a name. JNDI is portable and independent of the actual implementation. Additionally, it specifies a service provider interface (SPI) that allows directory service implementations to be plugged into the framework. The JNDI service implementations are hidden from the user, and may make use of a server, a flat file, or a database. The choice is up to the JNDI provider.

### Guidance

- Connections to the enterprise (e.g., **LDAP**, JNDI, **JMS**, databases) should use vendor-neutral interfaces. [G1071]
- J2EE applications should isolate tailorable data values into the deployment descriptor. [G1079]
  - Define all external resources by using a separate resource-ref element for each resource. [G1200]
  - Define configuration data such as environment variables, parameters, and properties by using resource-env-ref elements. [G1201]

### Best practices

- If using Java-based messaging (e.g., JMS), register destinations in JNDI so message clients can use JNDI to look up these destinations. [BP1116]

### Examples

```
// Create a hashtable that contains the parameters used to initialize
// JNDI.
Hashtable contextParams = new Hashtable();
// Specify the context factory to use. The context factory is provided
// by the
// implementation.
contextParams.put( Context.INITIAL_CONTEXT_FACTORY,
"com.jndiprovider.ContextFactory");
// The next parameter is the URL specifying the location of the JNDI
// provider's data store
contextParams.put( Context.PROVIDER_URL, "http://jndiprovider-database"
);
// Create the JNDI provider's context.
Context navyCurrentContext = new InitialContext ( contextParams );
// Look up the desired bean using its full name.
Object reference = navyCurrentContext.lookup ( "mil.us.navy.NavyBean"
);
// Cast the located bean to the desired type.
MyBean navyBean = (NavyBean) PortableRemoteObject.narrow (
```

## Universal Description, Discovery, and Integration (UDDI)

**Universal Description, Discovery, and Integration** is a service registry that defines a standard way to publish and discover information about **XML web services**. The XML schemas associated

with UDDI define four types of information that would enable a developer to use a published XML web service.

For UDDI guidance, see UDDI guidance in the web services guidance section.

## References

*<http://docs.sun.com/source/816-6696-10/dirintro.html>*

*<http://raleigh.pm.org/ldap-talk.html>*

*<http://www.gracion.com/server/whatldap.html>*

*<http://java.sun.com/products/jndi/overview.html>*

# Quality of Service (QoS)

## Overview

**Quality of Service (QoS)** refers to a network's ability to provide better service to selected network traffic. This requires sufficient network resources (computing resources, storage, buffer space, communication bandwidth, etc.) to ensure that the selected traffic meets its requirements for throughput, latency, jitter, and data loss.

Effective implementation of QoS requires the cooperation and participation of all components in the network that implement traffic management and control functions. QoS traffic management techniques include:

- Classification
- Compression
- Packet shaping and *TCP* optimization
- Prioritization
- Differentiated services
- Integrated services and policing

## Differentiated and guaranteed services

The two most prevalent types of QoS services are **differentiated** and **guaranteed** services. This section discusses QoS capability *as* differentiated services, priority, packet shaping, compressions, and TCP optimization.

Guaranteed services will not be addressed at this time because RSVP will not be supported. This is because the HAIPIS specification does not allow passing of the RSVP signaling from the red to black side, and thus *it* is not available for use in the black core. Applications will not be able to use RSVP in this environment.

## Implementations

Most QoS traffic management techniques are implemented using the information in the traffic's *IP* packet header. This could include:

- *IPv4* source and destination addresses
- IPv4 DSCP field (or IP-precedence field or TOS field)
- *IPv6* source and destination addresses (plus associated masks)
- IPv6 traffic class field
- IPv6 flow label field
- IPv6 next header field

The DSCP field supports differentiated services. This field consists of a 6-bit value used to determine QoS capability. This packet value is set or marked using network components such as packet shaper. Packet shaper provides packet marking/shaping, optimization and compression.

Inspecting and applying predetermined rules for the handling of the packet make QoS routing decisions.

The source and destination addresses, the IPv4 DSCP/TOS field, IPv6 traffic class, and the IPv6 flow label fields are the only fields that the hosts and applications need to specify to implement QoS in future **GIG** architectures. The use of the IPv6 flow label field is still under development by the **IETF**. It may be used for path control.

## References

- RFC 1633, "Integrated Services in the Internet Architecture: an Overview," June 1994.
- RFC 2211 "Controlled-Load Network" and RFC 2212 "Guaranteed Quality of Service" describe services that a network can provide and the mechanisms it would use.
- RFC 2386, "A Framework for QoS-Based Routing in the Internet," August 1998.
- RFC 791. Internet Protocol Specification
- RFC 1809, "Using the Flow Label Field in IPv6"
- RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification"
- [http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito\\_doc/qos.htm#1020563](http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm#1020563)
- <http://www.cisco.com/warp/public/732/Tech/qos/>

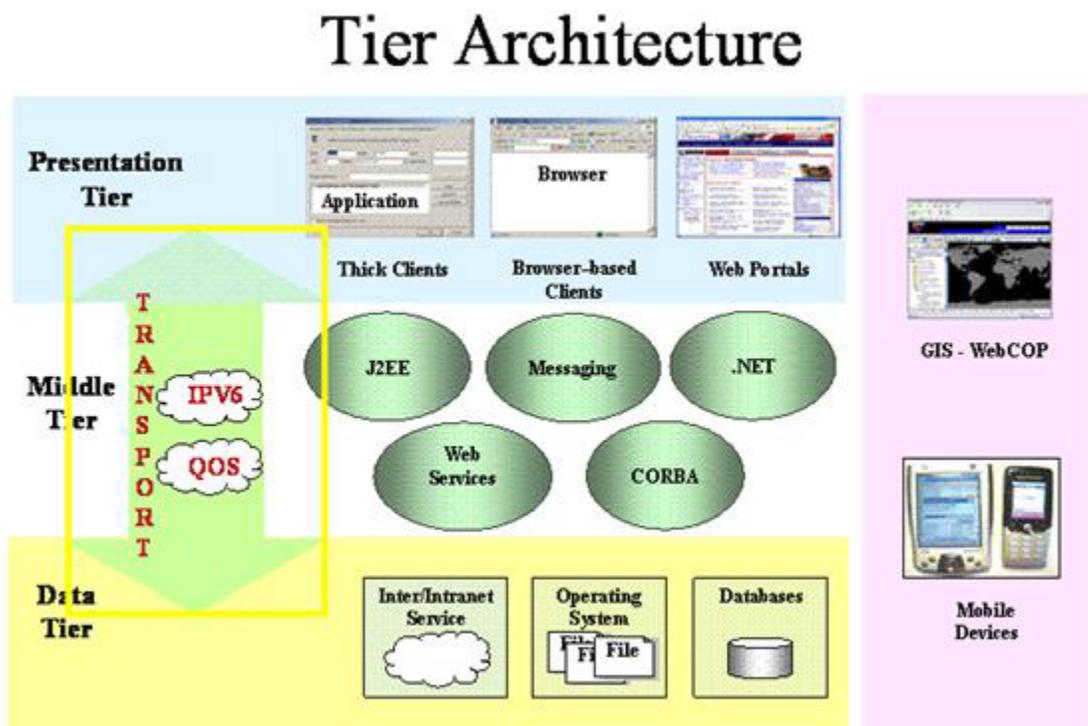


---

# Communications and transport

This section contains information on the following topics:

- *Joint Tactical Radio System (JTRS)*



Future guidance will include:

- **Design patterns and examples:** Recommended patterns and implementations
- **HAIGE**
- **IPv6/IPv4:** Migrating to IPv6; IPv6 and IPv4 coexistence.
- **Software radio:** Developer guidelines for the Joint Tactical Radio System (*JTRS*) and Software Communication Architecture (*SCA*).

Note that this guidance may be moved to other sections of the *NESI* documentation, as appropriate.

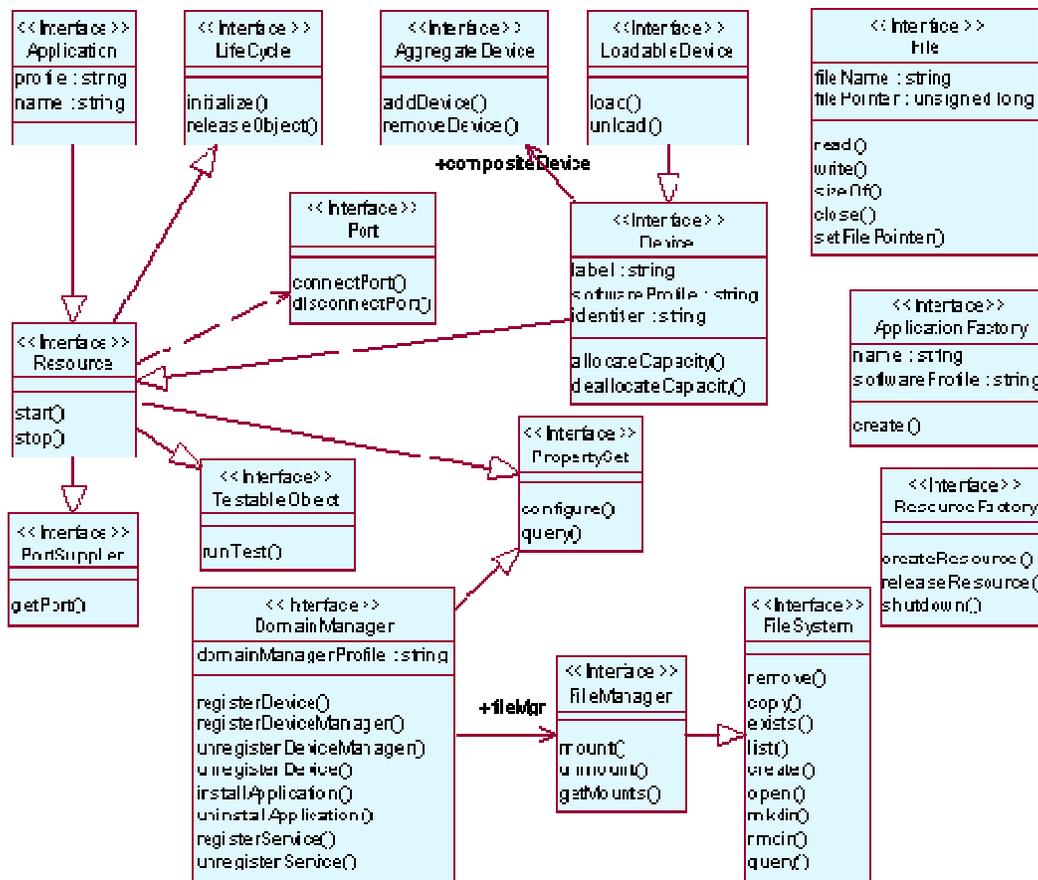
# Joint Tactical Radio System (JTRS)

This section describes the Joint Tactical Radio System (*JTRS*) Software Communications Architecture (*SCA*).

## Overview

A software-defined radio includes a transmitter in which you can alter the operating parameters of the transmitter by changing only the software without making any hardware changes. The operating parameters include the frequency range, modulation type, and maximum radiated or conducted output power.

The **Software Communication Architecture (SCA)** is a complex specification with several different interfaces. The diagram below shows the primary SCA interfaces. An interface is an important concept. The interface is the only exposure of a software component to the outside world. Components in the system can only execute the operations defined in the interface definition.



Primary SCA interfaces

## Member variables

Member variables are not exposed to the outside world. To explain this, consider the device interface shown in the diagram. It provides an interface with both attributes and operations. (The attributes are the first compartment and operations listed in the second compartment.) It is easy to make the erroneous association of CORBA attributes to C++ member variables and CORBA operations to C++ operations. In CORBA, both attributes and operations are operations. Attributes have implicit set and query operations. Again using the device interface in the diagram as an example, the `label` attribute has implicit operation signatures:

- `label(in listString:string):void`
- `label(void):string.`

The software component is responsible for providing the internal storage variable for the `label` string. It is not directly available to the outside world. The CORBA interface provides the implicit operations for changing the variable.

In contrast, the `allocateCapacity()` operation of the device interface has a defined function signature instead of the implicit signatures of attributes. Since operations have improved exception handling capability, many programmers use only operations in an interface definition. However, the SCA uses both attributes and operations in some interfaces.

## Resource interface

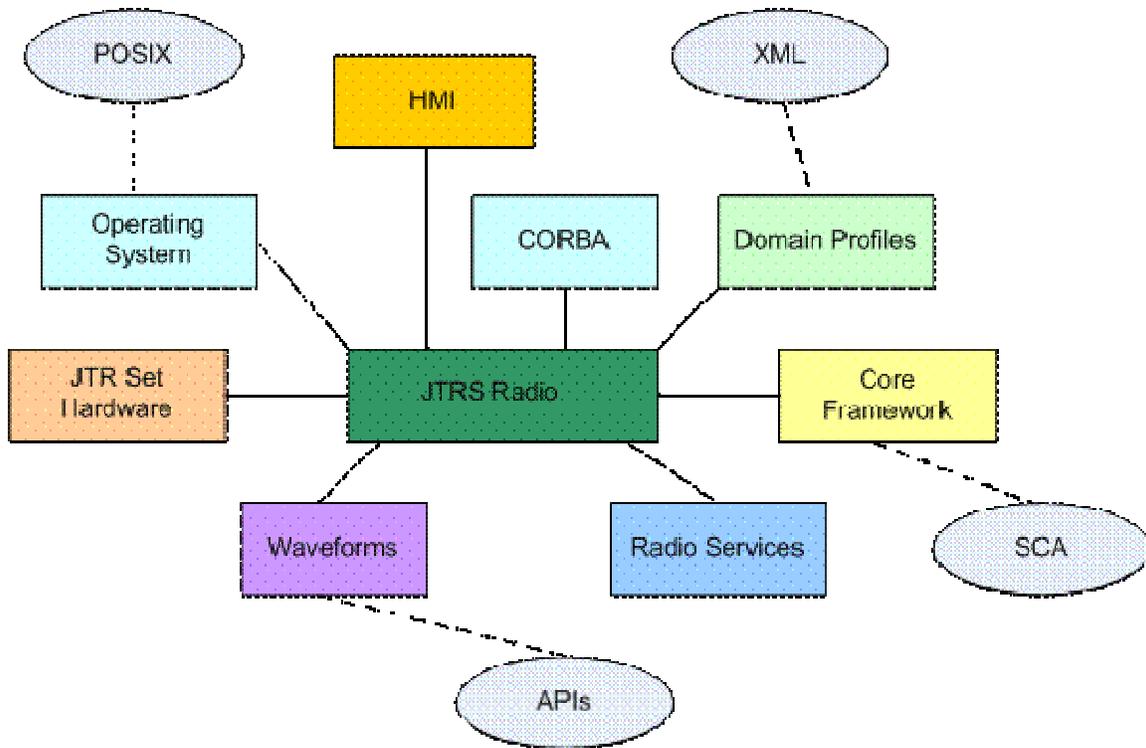
An important interface for SCA components is the resource interface. As shown in the diagram, it inherits interfaces from four other interfaces:

- `TestableObject`
- `PortSupplier`
- `LifeCycle`
- `PropertySet`

The resource interface is inherited by both applications and hardware devices. Because of its importance, the *example* in this section will define a software component that inherits the resource interface. It could inherit other interfaces, but this would add complexity without providing further insight into the development of SCA components.

## Composition of a JTRS radio

The following diagram shows a simple *JTRS* radio composition. The basic concept of JTRS is to define the Application Program Interface (APIs) of all the software elements within the radio using the JTRS Software Communications Architecture (*SCA*). The APIs define the input/output parameters of the software operations and the expected behavior.



### Composition of a JTRS radio

#### **Hardware**

As shown in the diagram, the JTRS radio has a hardware set that provides the processing, cryptographic, and RF resources required to instantiate military waveforms. The hardware supports an operating system which is specified in the SCA as an approximate equivalent to the IEEE-defined POSIX PSE-52.

#### **CORBA middleware**

The CORBA middleware provides the messaging services used to interconnect software components of applications and waveforms. CORBA utilizes *TCP/IP* to send messages between components (objects) and uses the Interface Definition Language (*IDL*) to specify the interfaces for the objects.

#### **Core Framework**

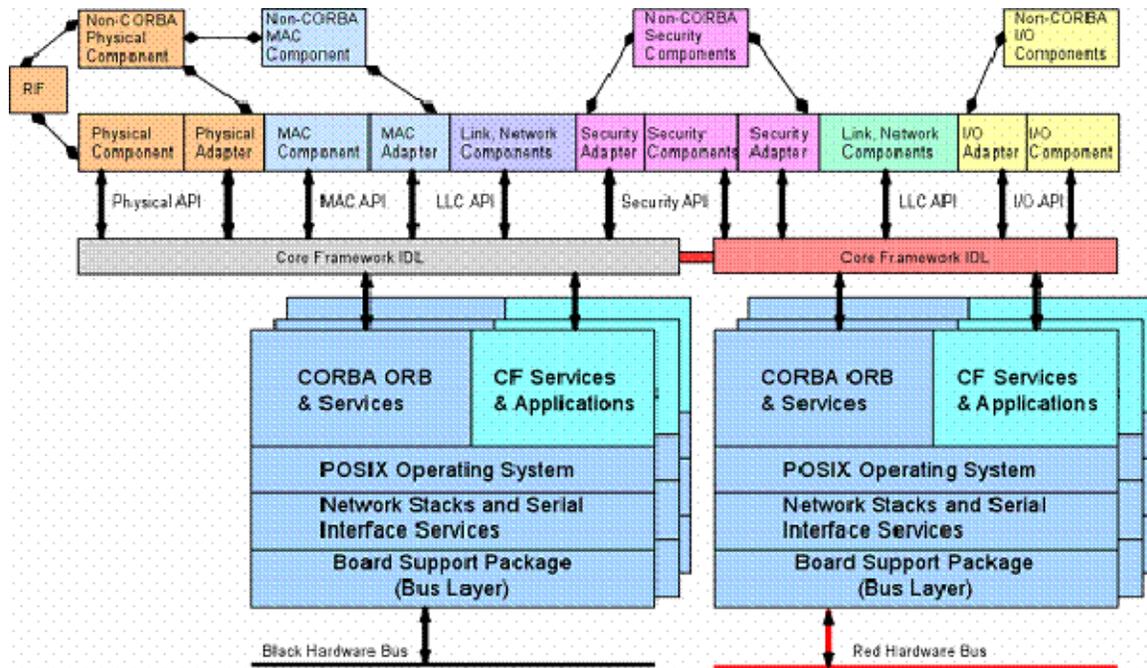
The Core Framework in the diagram can be considered a specialized operating system for radios that executes on top of the native operating system. It provides specific functionality such as the application factory, which can:

- Read the XML-based domain profiles
- Determine from the *XML* which radio resources are required for a waveform
- Load/instantiate all of the required components for a waveform

The interconnection of the software components is completely dynamic and directed by the connections defined in the domain profiles.

## Software

All software within JTRS is both SCA- and CORBA-compliant. This enables more flexibility and scalability than any previous generation of radios. A complete JTRS software architecture is shown in the following diagram. The lower portion of the diagram represents the CORBA, Core Framework, and operating system. This functionality is provided by the JTR set and is named the Application Environment Profile (AEP). Through the SCA specifications and CORBA messaging, all of the software in the upper layers should be able to execute upon any JTR set.

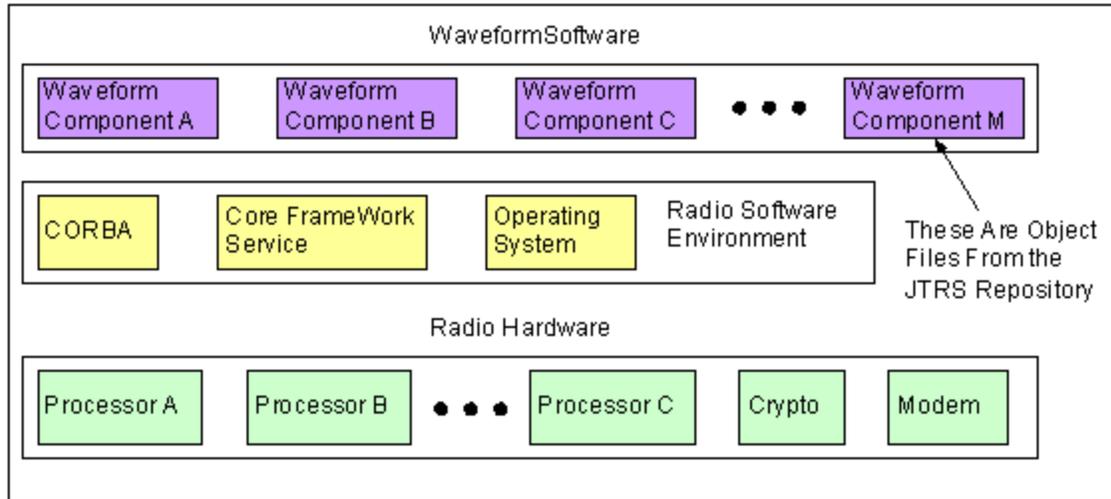


JTRS software architecture

### Component placement

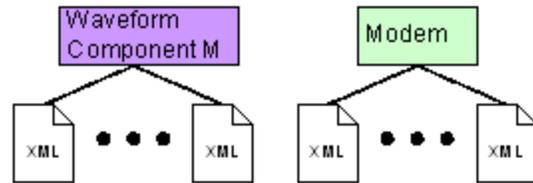
- The *CORBA middleware* allows software components to be distributed anywhere within the radio.
- The *Core Framework* provides distributed object launchers for each processor board within the set.
- The radio's *application factory* launches a waveform or application by providing the object files and execution parameters to the various processors within the radio.

The following diagram shows the configuration after these components are launched.



**Waveform Component M** This Component Expects Other Components and Hardware Devices, but Doesn't Know Their Location or Capacity

Instead of Having Custom Builds for Individual Radios, We Have XML Files Associated With the Waveform Components and Hardware Devices that Allows the Waveform Startup to Find and Allocate Components Dynamically



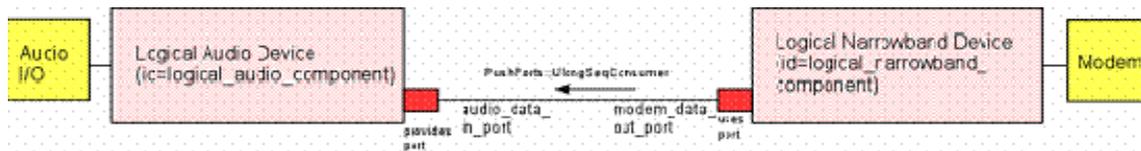
**XML within the JTRS radio set**

After the objects are instantiated, they may be co-located, or distributed among the different processing elements within the radio. These objects do not have any knowledge of other application objects or the hardware resources within the radio.

A set of XML files is associated with each software and hardware object. These files provide information about the objects, including their port references. The application factory parses these files along with an application schematic file, the **Software Assembly Descriptor (SAD)**. The SAD provides the necessary information to connect the hardware and software components together.

**Dynamic software configuration**

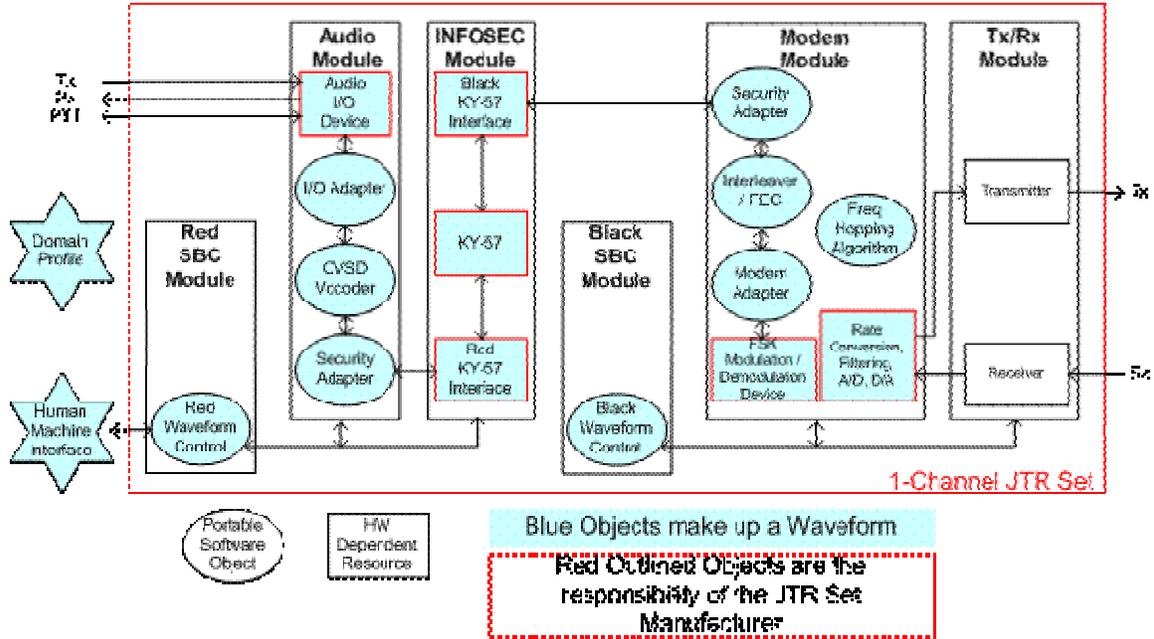
The following diagram shows two software components of a waveform within a **JTRS** radio. As depicted, the two software components have port objects which are dynamically connected by the Core Framework's application factory. After connection, the **CORBA** middleware allows the two objects to pass data or send control information. Because CORBA provides distributed processing the two software components in the diagram can actually reside in different processors within the radio.



**Two software components within a JTRS radio**

The software components shown in the diagram function as *adapters* (drivers) for the hardware components. By providing CORBA adapters for the hardware components, every hardware item and software component can communicate and be controlled via CORBA within the JTRS radio.

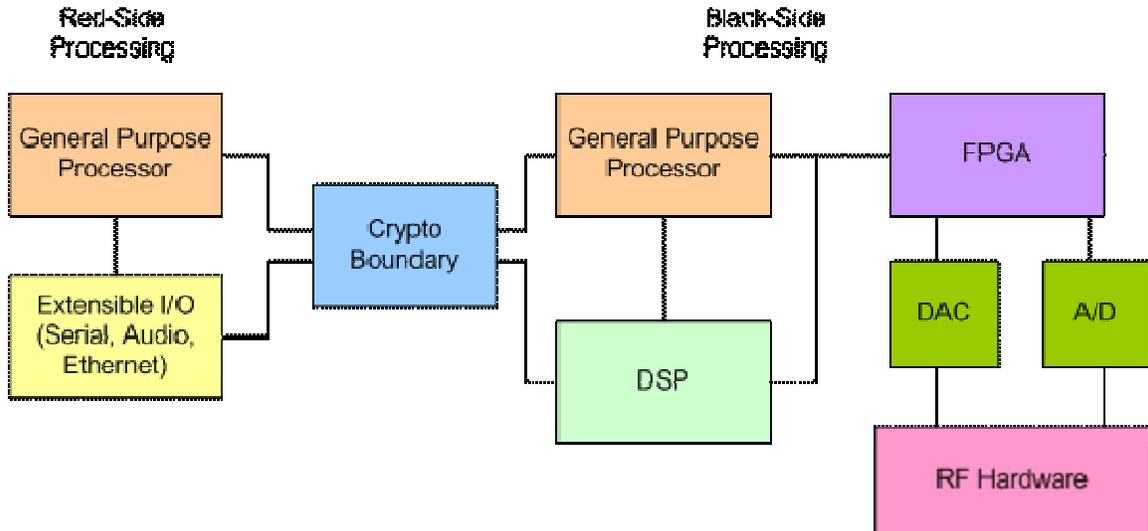
The following diagram illustrates a simple waveform within a JTR set. The oval objects represent software components, and the rectangular objects represent hardware items.



**Waveform within a JTR set**

**Hardware configuration**

The *SCA* does not specify a particular hardware configuration. However, one of the requirements for SCA certification is that the waveform must be ported successfully to a government test platform. These test platforms have a configuration similar to that shown in the following diagram. As would be expected, most waveform software is being designed for such a configuration.



**“Non-Standard” JTRS architecture**

The **DAC** and A/D in this diagram connect directly to the RF hardware. Most previous military radios had a specialized downconverter and modulator integrated circuits. With the non-standard JTRS configuration shown here, the waveform developers must provide FPGA code that can perform the function of operating directly with the A/Ds and D/As. The hardware does not provide direct digital synthesizers and upsamplers typical in previous radios. The waveform designer must provide that functionality in specialized FPGA code that constitutes part of the delivered waveform.

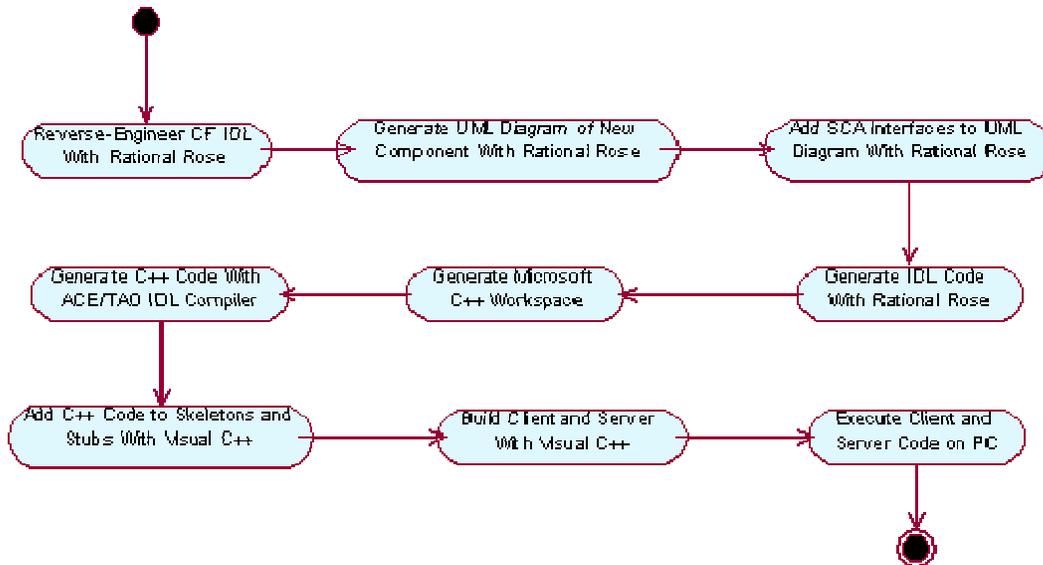
**Example: SCA-compliant software component**

This “Hello SCA” example demonstrates a simple but encompassing development of a SCA-compliant software component on a Windows 2000 platform. This example uses the Rational Rose model of the defined **SCA** Core Framework to generate the **IDL** for the software component. It was built using Microsoft Visual C++.

For this example, you need to:

- Install ACE **ORB**
- Set the ACE\_ROOT environment variable to the ACETAO installation directory
- Change the system Path variable to include %ACE\_ROOT%\bin. For more information, see the *TAO Developer’s Guide* (Object Computing, Inc., Version 1.1a, ociweb.com, 2000).

The activity diagram of the development process appears below.



**Overview of example**

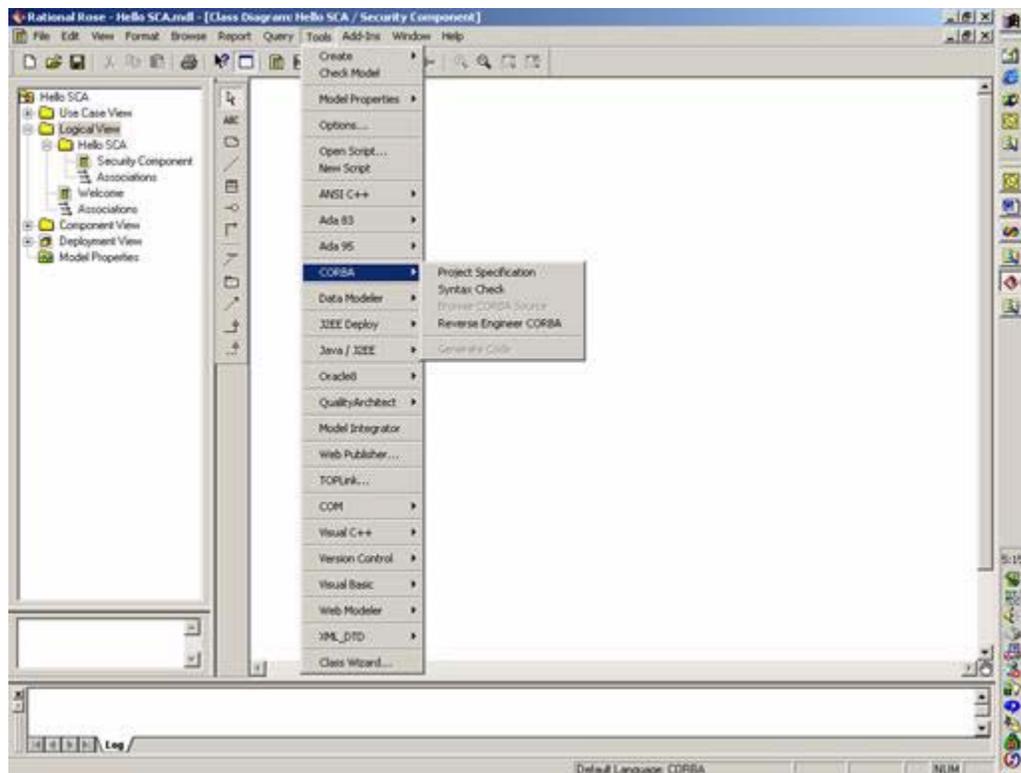
This example involves the following actions:

1. Reverse-engineer the Core Framework Interface Definition Language (IDL)
2. Generate the Rational Rose interfaces.

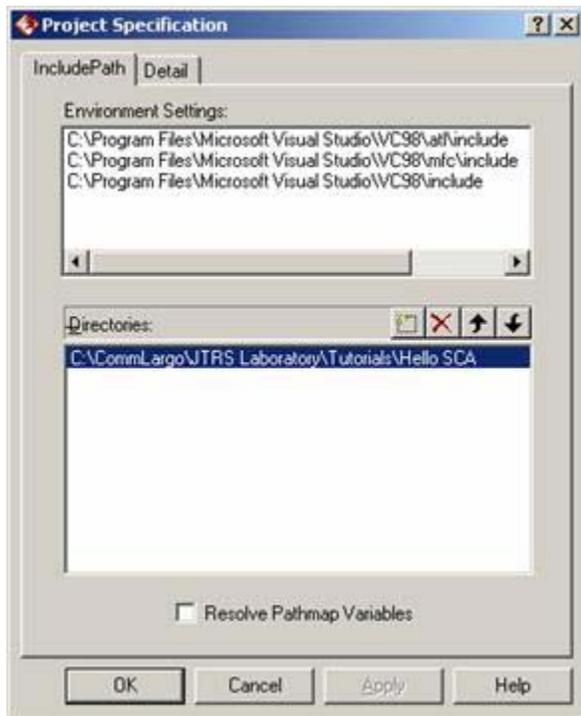
3. Design a simple software object.
4. Add the interfaces to the software object.
5. After completing the object design, generate a new IDL that provides the new object's interface.
6. Take the new IDL and generate skeleton and stub code with the ACE/TAO IDL compiler.
7. Add the object-specific C++ code for the client and server.
8. Execute the objects on a desktop personal computer.

## Develop the Rose model

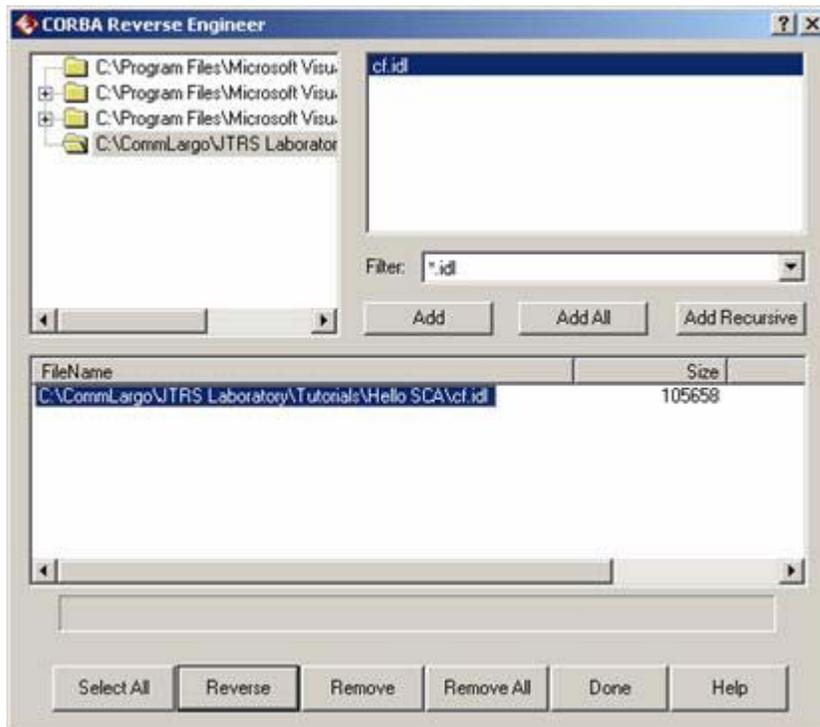
1. Open the Rose Model Browser.
2. Create a new package named Hello\_SCA by right-clicking on the logical package in the model browser on the left-hand side of the window.
3. Open the new package by double-clicking on *it*, and then right-click again on the package.
4. Create a new class diagram named Security\_Component.
5. Double-click **Security Component** in the browser window to open the window.
6. Open the **Tools > CORBA > Specification** menu, as shown below, and specify this folder so that Rose can find the **cf.idl** file.



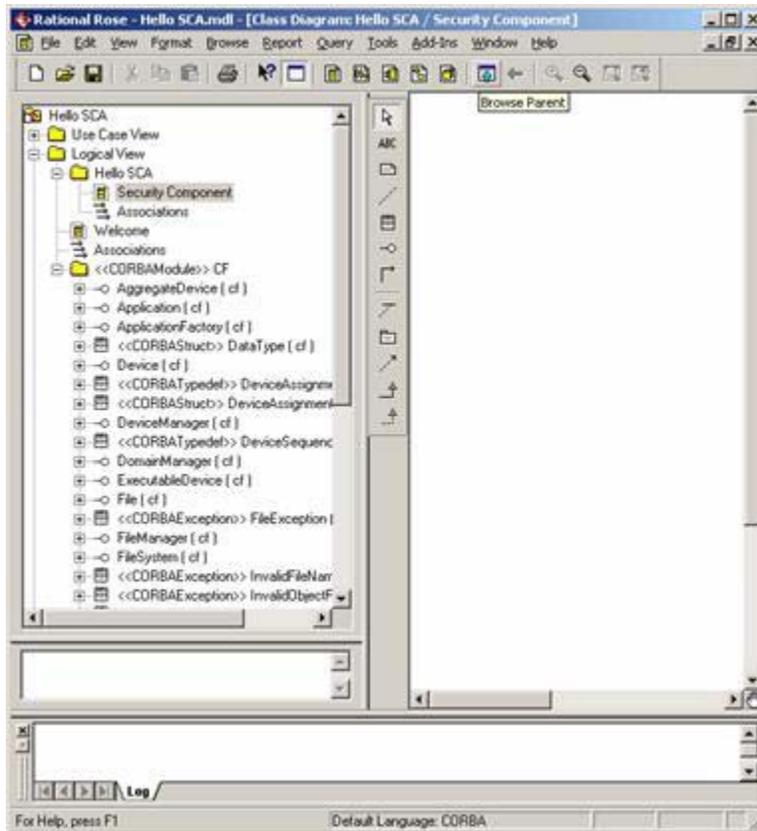
7. Using the **Insert** button in the **Specification** menu, insert the directory containing the **cf.idl** file. Click **OK**.



8. Open the **Tools > CORBA > Reverse Engineer** menu and select the file to be reverse-engineered. The Reverse Engineer window opens.
9. Select the **cf.idl** file in the lower window and click **Reverse** to reverse engineer the **cf.idl** file. This automatically generates a package <<CORBA Module>>CF under the logical view in the Rose browser. Click **Done**.



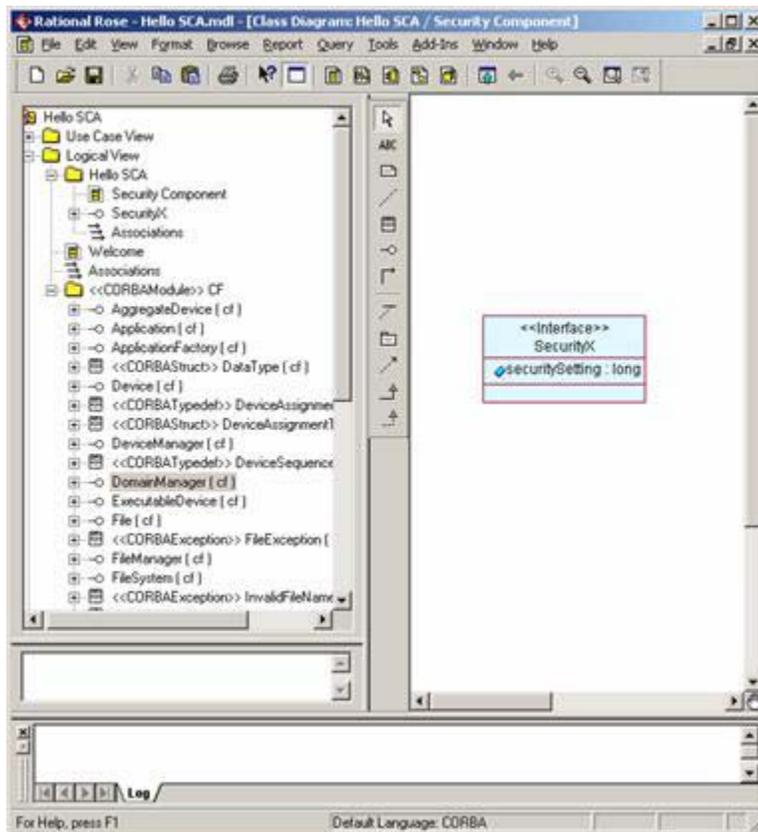
The following screen shows how the reverse engineering created the Core Framework (CF) module in the Rose Model Browser. Note that all of the different Core Framework interfaces (such as AggregateDevice) are defined in the browser and can now be added into a model diagram.



## Core framework

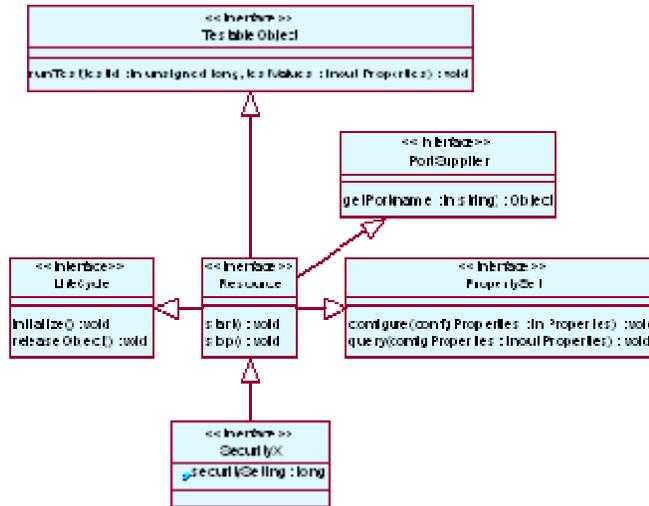
### Construct the software component's class diagram

1. Select the **Class** icon in the center vertical bar of the Rose window.
2. Click anywhere in the right-hand window. An unnamed class object appears in the diagram.
3. Double-click the class object and enter <<Interface>>SecurityX. This defines a CORBA interface named SecurityX.
4. Right-click the SecurityX interface and select **New Attribute**. Enter securitySetting:long.
5. Scroll in the left-hand browser window to the <<CORBA Module>>CF package and double-click the package. The following window appears.

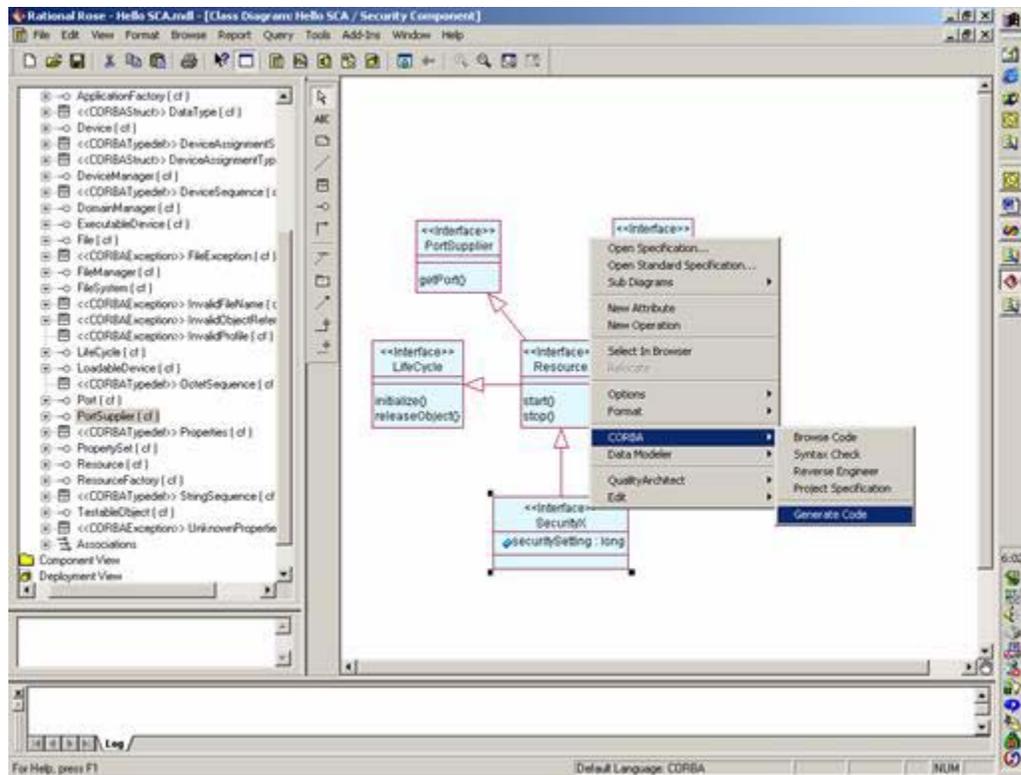


6. Find the Resource interface in the browser window and drag *it* into the window.
7. Select the generalization arrow (solid line with hollow head). In the class diagram window, click and drag from the SecurityX interface to the Resource interface. Now the inherited interfaces of Resource can also be shown in the diagram.
8. Drag the LifeCycle, TestableObject, PortSupplier, and PropertySet interfaces into the window. Note that the inheritance arrows are generated automatically from the associations contained in the core framework file.

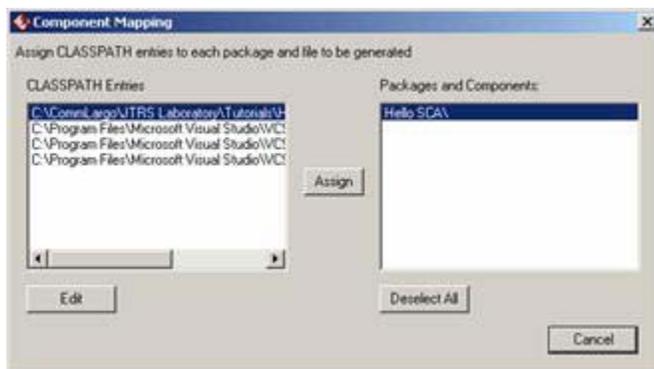
The following diagram shows the *UML* model of the software component. This is the class diagram, which represents a static view of the component. Note that the *SecurityX* component inherits the Resource interface previously defined in the Core Framework. Because of inheritance, the other interfaces inherited by the Resource interface are automatically included in the definition of the *SecurityX* component.



- Right-click the SecurityX component and select **CORBA > Generate Code**, as shown below.



A second menu appears.



10. Select the desired directory in the left window and the package in the right window. Click **Assign**.

Code generation begins and a window alert indicates "Code Generation Completed Successfully." Rational Rose has generated a new folder in the selected directory titled **Hello SCA** and placed the file, **SecurityX.idl**, within it.

## Develop the Visual C++ project

The following topics explain how to develop the Visual C++ project.

The **IDL** file generated by the Rational Rose CORBA compiler is not executable code. It resembles a **.h** header file instead of a **.cpp** source file. In this next section you will prepare a Hello SCA project to generate and execute the example. The example borrows liberally from the *TAO Developer's Guide* (Object Computing, Inc., Version 1.1a, <http://www.ocweb.com>, 2000) in describing the Visual C++ project.

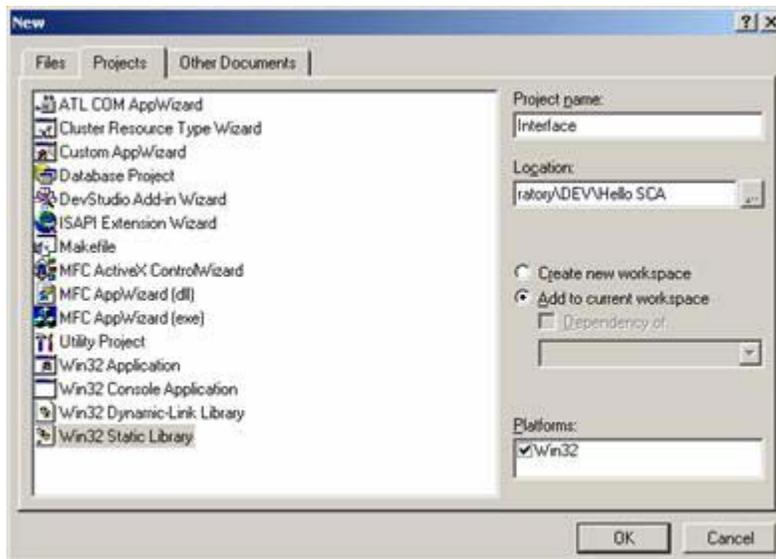
## Configure Visual C++

1. Open the Visual C++ IDE **File** menu and select **New**. A New dialog box opens.
2. Select the **Workspaces** tab.
3. For the Workspace name, enter `Hello SCA`.
4. For the location, select the directory previously used for the Rational Rose *Hello SCA* model.

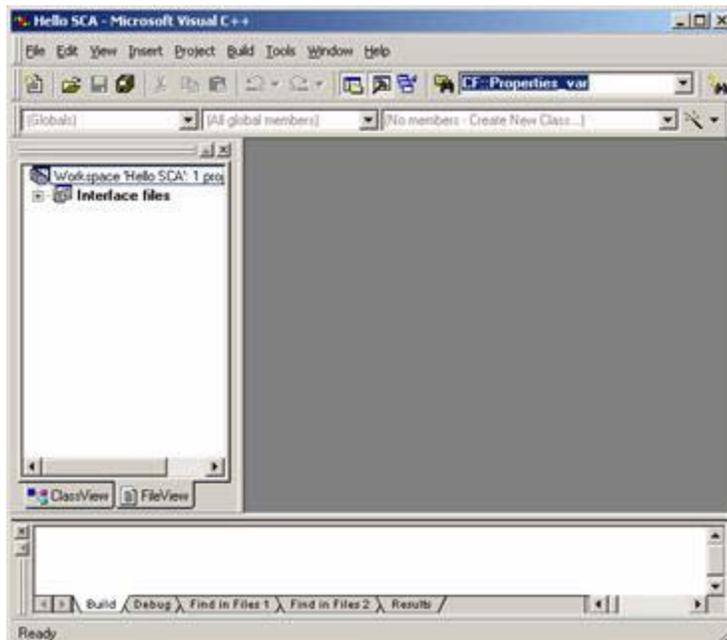
## Generate the interface project

In this step, you will set up the workspace, then relocate some files.

1. In the left-hand browser window, there is a single workspace icon labeled **Workspace 'Hello SCA':0 file(s)**. Right-click the icon to add another project to the workspace. A New dialog box appears.
2. Select **Win32 Static Library**, and enter a project name of **Interface**.
3. Before closing the dialog box, edit the **Location** text box. In the sample dialog box shown below, Visual C++ would automatically create a new directory called **Interface**, which would cause some problems with linking. To avoid this, delete the **/Interfaces** appendix to the directory in the **Location** text box.



4. Click **OK**. A Win32 Static Library wizard appears.
5. Click **Finish** without selecting either of the options. A New Project Information dialog box appears.
6. Click **OK**. The Visual C++ Workspace window appears and shows the interface project.

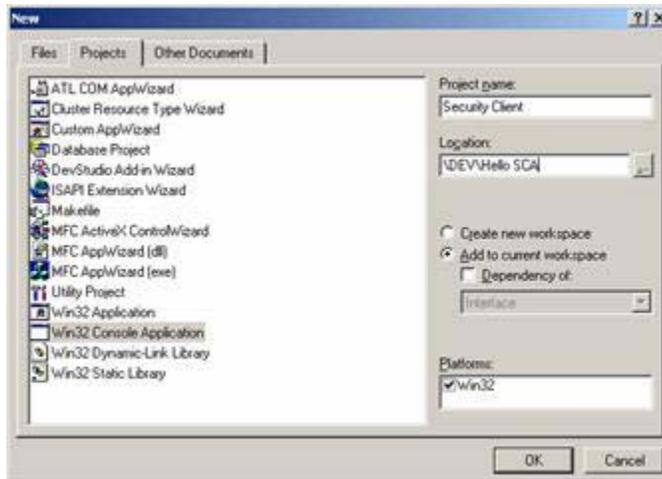


Creating the Win32 Static Library enables you to compile all of the Core Framework interfaces into a library that you can link with the server and client projects. This isolates the various files and serves as a model for larger projects.

### Generate the client project

In this step, you create a new project within the same workspace for the client.

1. In the left-hand browser window, there is a single workspace icon labeled **Workspace 'Hello SCA':1 file(s)**. Right-click the icon to add another project to the workspace. A New dialog box appears.
2. Select **Win32 Console Application**, and enter a project name of `Security Client`.
3. Edit the **Location** text box. In the sample dialog box shown below, Visual C++ would automatically create a new directory called **Security Client**, which would cause some problems with linking. To avoid this, delete the `/Security Client` appendix to the directory in the **Location** text box. Click **OK**.



## Generate the server project

To host the *SCA* object, you must create an additional project for the server.

1. In the left-hand browser window, there is a single workspace icon labeled **Workspace 'Hello SCA':2 file(s)**. Right-click the icon to add another project to the workspace. A New dialog box appears.
2. Select **Win32 Static Library**, and enter a project name of `Security Server`.
3. Edit the **Location** text box. Left as is, Visual C++ would automatically create a new directory called **Security Client**, which would cause some problems with linking. To avoid this, delete the `/Security Client` appendix to the directory in the **Location** text box.
4. Click **OK**. A New dialog box prompts you to enter the type of application you wish to create.
5. Click **Finish** to select the default empty application. A New Project Information dialog box appears.
6. Click **OK**.

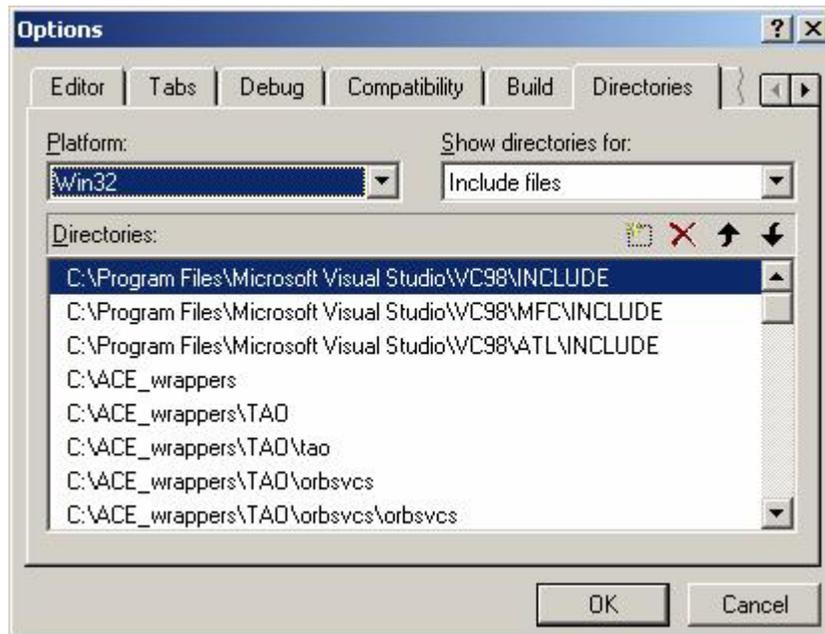
## Configure Visual C++ for the ACE/TAO compiler

In this step, you will modify the Visual C++ settings for the ACE/TAO compiler and establish paths for the Visual C++ linker.

1. Open the **Tools > Options > Directories** menu.
2. Open the **Show directories for** drop-down and select **Include Files**.

3. Inside the **Directories** list box, enter:

```
C:\ACE_wrappers
C:\ACE_wrappers\TAO
C:\ACE_wrappers\TAO\tao
C:\ACE_wrappers\TAO\orbsvcs
C:\ACE_wrappers\TAO\orbsvcs\orbsvcs
```



4. Open the **Show directories for** drop-down and select **Executable Files**.
5. Inside the **Directories** list box, enter:

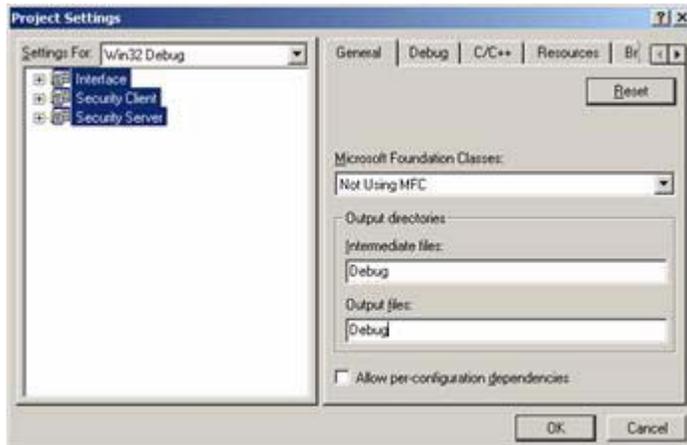
```
C:\ACE_wrappers\bin
```

## Configure the Visual C++ project settings

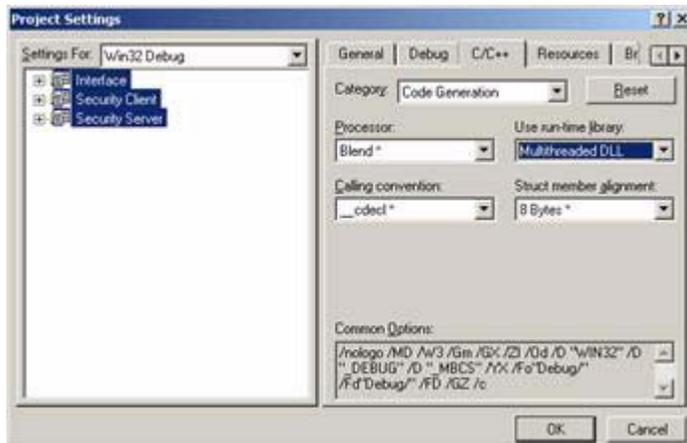
In this step, you configure the workspace.

1. Open the **Projects > Settings** menu.
2. Open the **Settings for** drop-down and select **Win32 Debug**.

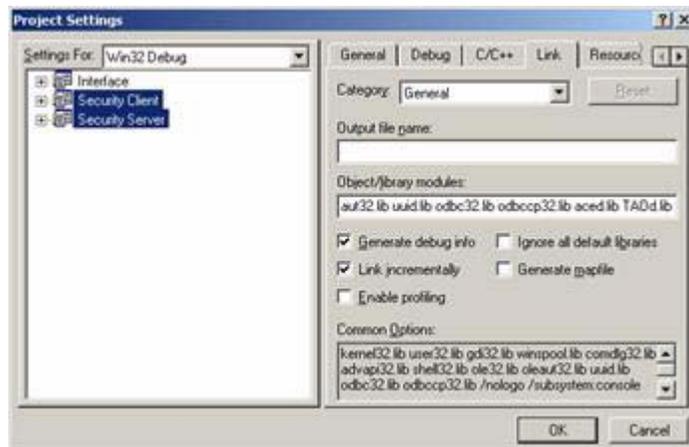
3. Select all three projects in the workspace, as shown below.



4. Select the **General** tab. As shown above, enter **Debug** in the **Intermediate files** and **Output files** text boxes.
5. Select the **C/C++** tab. Open the **Category** drop-down and select **Code Generation**.
6. Open the **Use run-time library** drop-down and select **Multithreaded DLL**.

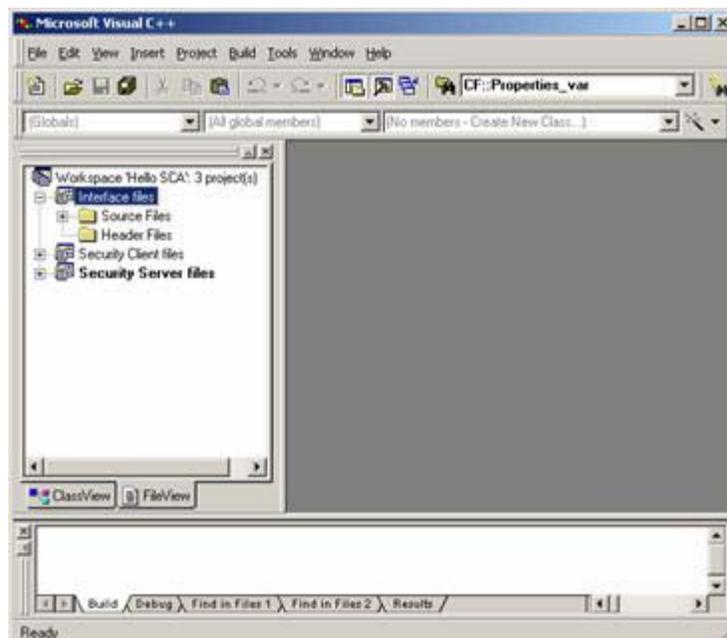


7. Select the Security Client and Security Server projects in the left-hand browser window.
8. Select the **Link** tab. Enter **aced.lib** and **TAOd.lib** in the **Object/library modules** text box.



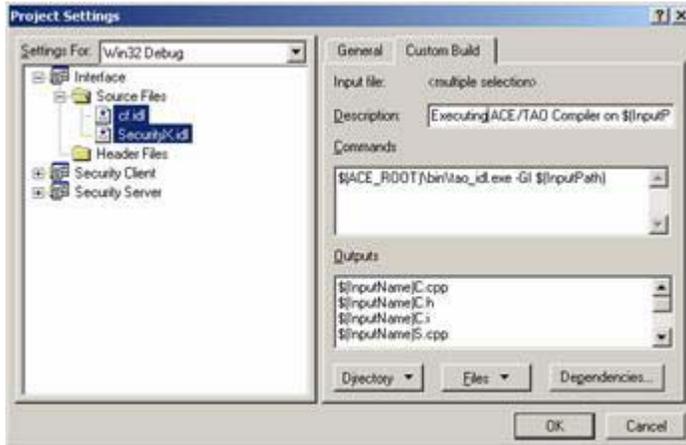
### Add the Rational Rose IDL files to the project

1. Verify that the two *IDL* files generated by Rational Rose are located in the same directory as the Hello SCA Visual C++ project.
2. Click the Interface Files project in the left-hand browser window to expose the Source and Header folders.



3. Right-click Interface Files: Source Files and select **Add Files to Folder**. Select the **cf.idl** and **SecurityX.idl** files generated earlier with Rational Rose. It may be necessary to move the files to the correct directory with Windows Explorer.

4. Select both files, located under Source Files, and right-click to adjust the settings.



5. Open the **Settings for** drop-down and select **Win32 Debug**.
6. Select the **Custom Build** tab.
7. In the **Description** text box, clear the previous text and enter:

Executing ACE/TAO Compiler on \$(InputPath).

8. In the **Commands** box, enter:

\$(ACE\_ROOT)\bin\tao\_idl.exe \_GI \$(InputPath).

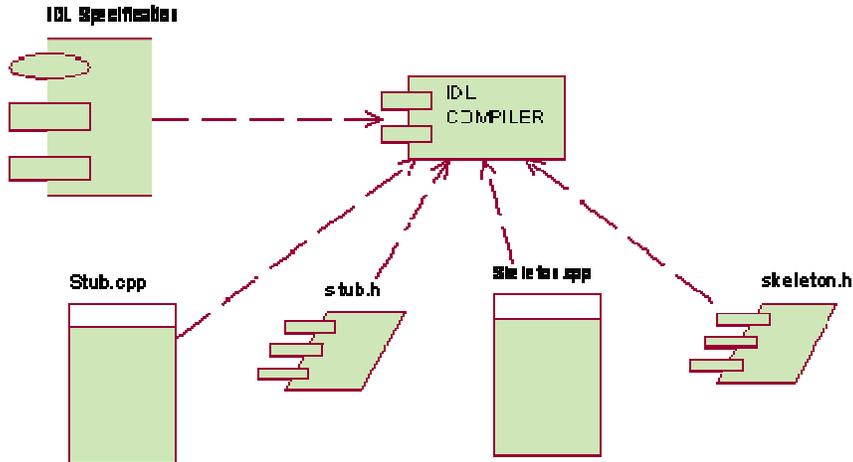
9. In the **Outputs** box, enter:

```
$( InputName )C .cpp
$( InputName )C .h
$( InputName )C .i
$( InputName )S .cpp
$( InputName )S .h
$( InputName )S .i
$( InputName )S_T .cpp
$( InputName )S_T .h
$( InputName )S_T .i
```

The completed window should be similar to the screen shown above.

## Use the ACE/TAO compiler to generate skeletons and stubs

In this step, you use the TAO compiler linked into the Visual C++ environment to automatically generate stub and skeleton code. The following diagram shows the process of automatically generating the code for the component implementation.



### Automatic code generation

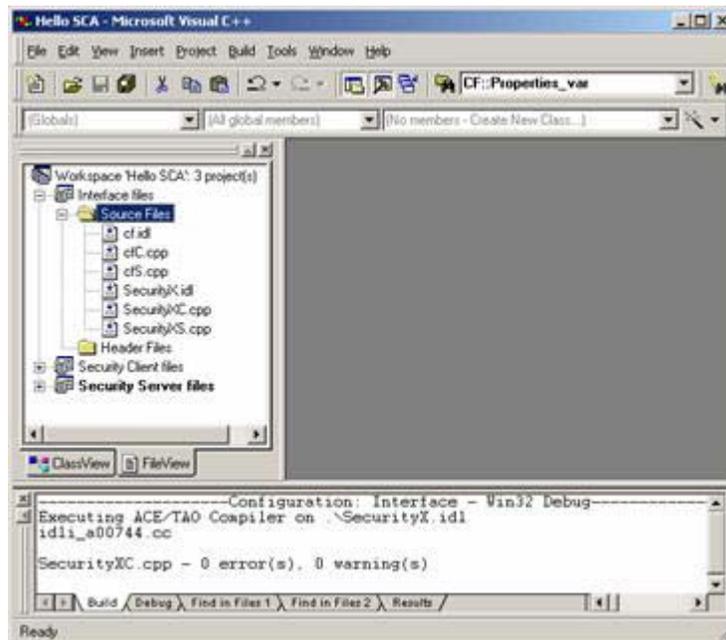
#### To generate files with the ACE/TAO compiler:

1. Right-click the **cf.idl** file and select **Compile cf.idl**. Because you set up the file for a custom build, the ACE/TAO compiler executes and generates all of the files you need for subsequent steps.
2. Right-click the **SecurityX.idl** file. Again, the ACE/TAO compiler executes and generates the necessary files.
3. Because the IDL compiler could be accidentally executed and overwrite the manual code changes you will subsequently make, follow the naming conventions suggested by <http://www.ociweb.com>. Change these file names:
  - **cfI.cpp** to **cf\_i.cpp**
  - **cfI.h** to **cf\_i.h**
  - **SecurityXI.cpp** to **SecurityX\_I.cpp**
  - **SecurityXI.h** to **SecurityX\_i.h**

#### To add the generated stubs and skeletons to the interface project:

1. Right-click Interface files: Source Files in the left-hand browser window and select **Add Files to Folder**.
2. Add these files, *as* shown in the window below:
  - **cfS.cpp**
  - **cfC.cpp**
  - **SecurityXS.cpp**

- **SecurityXC.cpp**



3. Create the Interface static library by right-clicking Interface Files in the browser window and selecting **Build**.

The Visual C++ compiler compiles the four files you just added to the project and builds the Interface library. The client and server projects use this library to access the required Core Framework code. To allow the Client and Server projects to access this library, you must adjust the project dependencies.

4. Open the **Project > Dependencies** menu.
5. Open the **Select project to modify** drop-down and select **Security Client**. In the check boxes under **Dependent upon the following project(s)**, select the box next to **Interfaces**.
6. Open the **Select project to modify** drop-down and select **Security Server**. In the check boxes under **Dependent upon the following project(s)**, select the box next to **Interfaces**.

## Develop the client

In an actual radio, you would not need to develop a specific client for the **SCA** object; the waveform application or one of its components would perform the function of client. This example however, demonstrates how the object operates without requiring a complete waveform infrastructure.

The complete file listing of **SecurityClient.cpp** is shown below. Review the comments for additional insight into the development of SCA-compliant code.

```
// SecurityClient.cpp : Defines the entry point for the console
// application.
//
#include "SecurityXC.h"
#include "cfC.h"
```

```

#include <ace/streams.h>

int main(int argc, char *argv[]){
    CORBA::Long userSecurityLevel;
    CORBA::Object* objectReference;
    char objectName$[15];
    strcpy(objectName$, "JTeL");

    try{
        // Initialize the Orb
        CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);

        // Convert the contents of the file to an object reference.
        CORBA::Object_var obj = orb-
>string_to_object("file://Security.ior");
        if(CORBA::is_nil(obj.in())){
            cerr<<"Nil Security reference"<<endl;
            throw 0;
        }

        // Narrow the object reference to a Security object
reference
        SecurityX_var security = SecurityX::_narrow(obj.in());
        if(CORBA::is_nil(security.in())){
            cerr<<"Not Security object reference" <<endl;
            throw 0;
        }

        // Generate the Properties reference and values needed for the
        // runTest operation. IMPORTANT!!! These variables must be
declared
        // after the ORB is initialized. Otherwise, the ORB will crash!
        CF::Properties_var parameters;
        parameters = new CF::Properties;
        parameters->length(1);
        (*parameters)[0].id = CORBA::string_dup("SPAWAR");
        // Observe the machinations necessary to define a parameter into
the
        // "any" type definition of DataType! Note the insertion operator
        // is used to "place" a value in the any field.
        (*parameters)[0].value <=< (CORBA::Long) 3;

        cout<<"Initial Security Rating"<<endl;

        cout<<security->securitySetting()<<endl;

        cout<< "Increase Security Rating by how much?"<<endl;

        cin>> userSecurityLevel;

        security->securitySetting(userSecurityLevel);

        cout<<"New Security Rating"<<endl;

        cout<<security->securitySetting()<<endl;

        security->runTest(5, *parameters);
    }
}

```

```

    objectReference = (CORBA::Object*) 0x003;
    objectReference = security->getPort(objectName$);

    cout<<"ObjectReference = "<<objectReference<<endl;

    // Release resources
    orb->destroy();
}
catch(const CORBA::Exception &ex){
    cerr<<"Caught a CORBA exception: " << ex <<endl;
    return 1;
}
cout<< "Message was sent" << endl;
return 0;
}

```

## Set up the client project

1. Copy the **SecurityClient.cpp** file to the working Visual C++ directory.
2. Right-click Security Client: Source Files in the left-hand browser window and select **Add Files to Folder**. Select **SecurityClient.cpp**.
3. Right-click the Security Client project icon in the browser window and select **Build**.

Visual C++ compiles, links, and builds the executable for the client that you will use to demonstrate the security object.

## Set up the server project

The server project is a little more complex, because you must provide develop the Security object and provide a host for it. Recall that when you *compiled the IDL files*, you generated skeleton files for the Core Framework and SecurityX object. The skeleton provides the C++ starter code for the project, but you must provide the code to implement the methods.

### **Sample file listing for skeleton**

You may add additional or different implementations. Notice that several of the methods have specific code by the comment `//Add your implementation here` indicated by the ACE/TAO compiler.

You must edit the file generated by the ACE/TAO compiler to match the file listing below.

```

// -*- C++ -*-
//
// $Id$
// **** Code generated by the The ACE ORB (TAO) IDL Compiler ****
// TAO and the TAO IDL Compiler have been developed by:
//   Center for Distributed Object Computing
//   Washington University
//   St. Louis, MO
//   USA
//   http://www.cs.wustl.edu/~schmidt/doc-center.html
//   and
//   Distributed Object Computing Laboratory
//   University of California at Irvine

```

```

//    Irvine, CA
//    USA
//    http://doc.ece.uci.edu/
//
// Information about TAO is available at:
//    http://www.cs.wustl.edu/~schmidt/TAO.html
#include "SecurityX_i.h"
// Implementation skeleton constructor
SecurityX_i::SecurityX_i (void)
{
    //Add your implementation here
    cout<<" -----" <<endl;
    cout<<"Beginning SecurityX object constructor" <<endl;
    securityRating = 0;
}
// Implementation skeleton destructor
SecurityX_i::~~SecurityX_i (void)
{
}
CORBA::Long SecurityX_i::securitySetting (
)
ACE_THROW_SPEC ((
    CORBA::SystemException
))
{
    //Add your implementation here
    cout<<" -----" <<endl;
    cout<<"Beginning implicit securitySetting query" <<endl;
    //Add your implementation here
    return (securityRating);
}
void SecurityX_i::securitySetting (
    CORBA::Long securitySetting
)
ACE_THROW_SPEC ((
    CORBA::SystemException
))
{
    //Add your implementation here
    cout<<" -----" <<endl;
    cout<<"Beginning implicit securitySetting set" <<endl;
    securityRating += securitySetting;
}
char * SecurityX_i::identifier (
)
ACE_THROW_SPEC ((
    CORBA::SystemException
))
{
    char* badIdea = "bad programming";
    return badIdea;
}
void SecurityX_i::start (
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::Resource::StartError

```

```

    ))
    {
        //Add your implementation here
        cout<<" -----" <<endl;
        cout<<"Beginning start Operation" <<endl;
        securityRating = 10;
    }
void SecurityX_i::stop (
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::Resource::StopError
))
{
    //Add your implementation here
    cout<<" -----" <<endl;
    cout<<"Beginning stop Operation" <<endl;
    securityRating = -1;
}
void SecurityX_i::initialize (
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::LifeCycle::InitializeError
))
{
    //Add your implementation here
    cout<<" -----" <<endl;
    cout<<"Beginning initialize Operation" <<endl;
    securityRating = 1;
}
void SecurityX_i::releaseObject (
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::LifeCycle::ReleaseError
))
{
    //Add your implementation here
    cout<<" -----" <<endl;
    cout<<"Beginning releaseObject Operation" <<endl;
    cout<<"We would have released an object if we had one!" <<endl;
    //Add your implementation here
}
void SecurityX_i::runTest (
    CORBA::ULong testid,
    CF::Properties & testValues
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::TestableObject::UnknownTest,
    CF::UnknownProperties
))
{
    //Add your implementation here
    cout<<" -----" <<endl;
    cout<<"Beginning runTest Operation" <<endl;

```

```

    cout<<"Received Test ID: "<<testValues[0].id<<endl;
    // Because the test properties is an "any" field, we do not know
    // the actual data type a priori. To work around this, we use
    // the extraction operators of the any class.
    CORBA::Long receivedValueL;
    CORBA::Short receivedValueS;
    if(testValues[0].value>>=receivedValueL)
    {
        cout<<"Received Test Parameter: "<<receivedValueL<<endl;
    }
    if(testValues[0].value>>=receivedValueS)
    {
        cout<<"Received Test Parameter: "<<receivedValueS<<endl;
    }
}
void SecurityX_i::configure (
    const CF::Properties & configProperties
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::PropertySet::InvalidConfiguration,
    CF::PropertySet::PartialConfiguration
))
{
    //Add your implementation here
}
void SecurityX_i::query (
    CF::Properties & configProperties
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::UnknownProperties
))
{
    //Add your implementation here
}
CORBA::Object_ptr SecurityX_i::getPort (
    const char * name
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::PortSupplier::UnknownPort
))
{
    //Add your implementation here
    cout<<" -----" <<endl;
    cout<<"Beginning getPort Operation" <<endl;
    cout<< "Requested port name was: " << name << endl;
    static CORBA::Object* nilPtr;
    nilPtr = CORBA::Object::_nil();
    return(nilPtr);
}

```

### To set up the server project:

1. Copy the **SecurityServer.cpp** file into the directory.

2. Right-click the Security Server project icon in the left-hand browser window and select **Add Files to Project**. Select **SecurityServer.cpp**, **SecurityX\_i.cpp**, and **SecurityX\_i.h**.

Note that **SecurityX\_i.cpp** and **SecurityX\_i.h** are the files that were generated by the IDL compiler and which you *renamed earlier*.

3. Right-click the **SecurityX\_i.cpp** file and edit it to match the listing provided above.

**Very important:** Don't forget the `#include "SecurityX_i.h"` at the top of the file. (You manually renamed the .h file after the IDL compiler had generated the code.)

Also, make sure that you add in all of the operational code under the `//Add your implementation here comments`.

4. Right-click the **SecurityX\_i.h** file for editing. The skeleton code generated by the IDL compiler does not know how you will implement the object. It is only aware of the defined interfaces.
5. Add a member variable to the SecurityX class and change the definition in the **SecurityX\_i.h** file.
6. At the top of the file, add the `securityRating` variable as shown in the code snippet below.

```
//Class SecurityX_i
class SecurityX_i : public virtual POA_SecurityX
{
    CORBA::Long securityRating;
public:
    //Constructor
    SecurityX_i (void);

    //Destructor
    virtual ~SecurityX_i (void);
```

7. Right-click the Security Server project icon in the browser window and select **Build**.

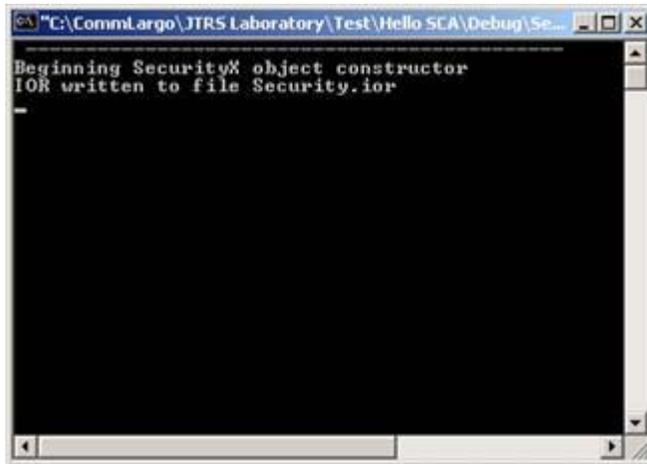
## Test the SCA

There are two methods of testing the **SCA** object:

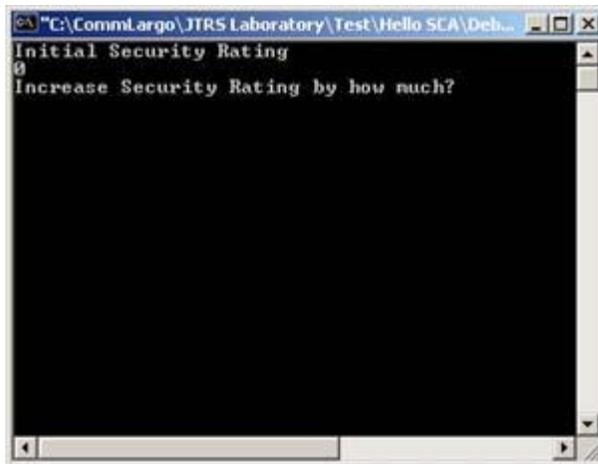
- Execute the \*.exe files from the **Debug** directory
- Execute the files individually from the Visual C++ workspace

Since you need to debug objects in the future, this example demonstrates the latter.

1. Right-click the Security Server project in the left-hand browser window and select **Set as Active Project**.
2. Open the **Build** menu and select **Execute Security Server.exe**. The Security Server console window opens.

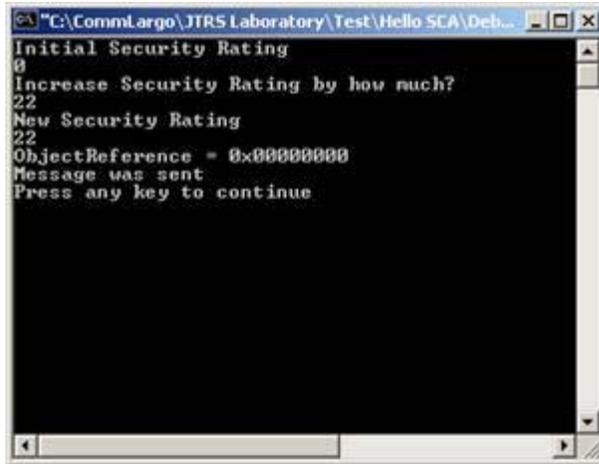


3. Right-click the Security Client project in the browser window and select **Set as Active Project**.
4. Open the **Build** menu and select **Execute Security Client.exe**. The Security Client console window opens.

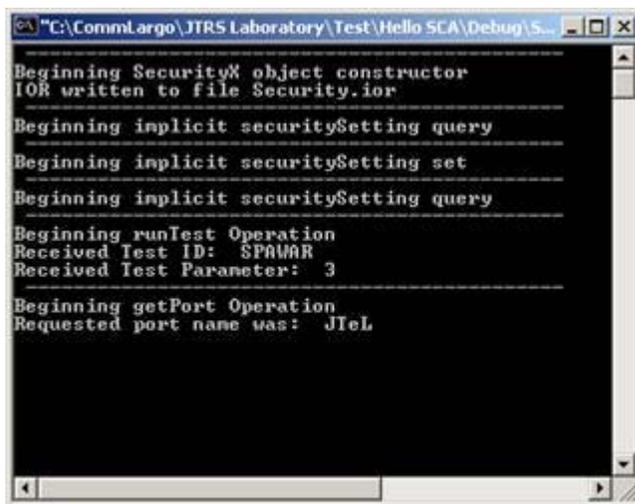


5. Enter a numeric value and the client exercises the Security object. The Security Client and Security Server console windows should now appear as shown below.

The Security Client console looks like this after user entry:



The Security Server console window looks like this after user input to client:



6. Close both windows.

## References

Object Computing, Inc., *TAO Developer's Guide*, Version 1.1a, <http://www.ocweb.com>, 2000.

### Web sites

Site	Description
<a href="http://jtrs.army.mil/">http://jtrs.army.mil/</a>	Joint Tactical Radio Systems, Joint Program Office ( <b>JTRS/JPO</b> )
<a href="http://jtrs.army.mil/sections/technicalinformation/fset_technical_sca.html">http://jtrs.army.mil/sections/technicalinformation/fset_technical_sca.html</a>	<b>SCA</b> technical

	overview
<a href="http://sca.jtrslab.org/">http://sca.jtrslab.org/</a>	SCA change proposal <i>portal</i> : SCA forum/Change proposal (account required)
<a href="http://www.sdrforum.org/">http://www.sdrforum.org/</a>	Software-defined radio forum (commercial SDR )
<a href="http://www.omg.org/">http://www.omg.org/</a>	Object Management Group
<a href="http://www.mprg.org/research/ossie/index.html">http://www.mprg.org/research/ossie/index.html</a>	OSSIE: Open-source SCA Core Framework
<a href="http://www.orcaf.com/">http://www.orcaf.com/</a>	ORCAF: Open-source (limited distribution) Core Framework
<a href="http://www.govcomm.harris.com/dmtk/index.html">http://www.govcomm.harris.com/dmtk/index.html</a>	Commercial SCA Core Framework
<a href="http://www.crc.ca/en/html/rmsc/home/sdr/projects/scari">http://www.crc.ca/en/html/rmsc/home/sdr/projects/scari</a>	SCA Reference Implementation Project (SDR-sponsored implementation project)

## Documents

### Version 2.2

Document (format)	Location
SCA v2.2 (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/V2.2/SCA_v2_2.zip">http://jtrs.army.mil/documents/sca_documents/V2.2/SCA_v2_2.zip</a>

SCA API Supplement (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/API_Supplement_files/API_v2.2.zip">http://jtrs.army.mil/documents/sca_documents/API_Supplement_files/API_v2.2.zip</a>
SCA Security Supplement (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/SECURITY_Supplement_files/SECURITY_V2.2.zip">http://jtrs.army.mil/documents/sca_documents/SECURITY_Supplement_files/SECURITY_V2.2.zip</a>

**Version 2.2.1**

<b>Document (format)</b>	<b>Location</b>
SCA v2.2 (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/V2.2/SCA_v2_2.zip">http://jtrs.army.mil/documents/sca_documents/V2.2/SCA_v2_2.zip</a>
SCA API Supplement (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/API_Supplement_files/API_v2.2.zip">http://jtrs.army.mil/documents/sca_documents/API_Supplement_files/API_v2.2.zip</a>
SCA Security Supplement (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/SECURITY_Supplement_files/SECURITY_V2.2.zip">http://jtrs.army.mil/documents/sca_documents/SECURITY_Supplement_files/SECURITY_V2.2.zip</a>
SCA v2.2.1 (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/V2.2.1/NoChangeBarPDF/SCA/SCA_v2.2.1.zip">http://jtrs.army.mil/documents/sca_documents/V2.2.1/NoChangeBarPDF/SCA/SCA_v2.2.1.zip</a>
SCA API Supplement (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/V2.2.1/NoChangeBarPDF/API/API_v2.2.1.zip">http://jtrs.army.mil/documents/sca_documents/V2.2.1/NoChangeBarPDF/API/API_v2.2.1.zip</a>
SCA Security Supplement (zip)	<a href="http://jtrs.army.mil/documents/sca_documents/V2.2.1/NoChangeBarPDF/Security/Security_v2.2.1.zip">http://jtrs.army.mil/documents/sca_documents/V2.2.1/NoChangeBarPDF/Security/Security_v2.2.1.zip</a>
SCA v2.2.1 Requirements (PDF)	<a href="http://jtrs.army.mil/sections/technicalinformation/req_trace_matrix/2.2.1/SCA">http://jtrs.army.mil/sections/technicalinformation/req_trace_matrix/2.2.1/SCA</a>
SCA SEC v2.2.1 Requirements (PDF)	<a href="http://jtrs.army.mil/sections/technicalinformation/req_trace_matrix/2.2.1/SCA_SEC_2.2.1_Requirements_(JPO_view).pdf">http://jtrs.army.mil/sections/technicalinformation/req_trace_matrix/2.2.1/SCA_SEC_2.2.1_Requirements_(JPO_view).pdf</a>
SCA API 2.2.1	<a href="http://jtrs.army.mil/sections/technicalinformation/req_trace_matrix/2.2.1/SCA">http://jtrs.army.mil/sections/technicalinformation/req_trace_matrix/2.2.1/SCA</a>

<i>Requirements (PDF)</i>	<i>API 2.2.1 Requirements (JPO view).pdf</i>
<i>SCA Requirement Attribute Descriptions (PDF)</i>	<i><a href="http://jtrs.army.mil/sections/technicalinformation/req_trace_matrix/SCA_Reqs_Attribute_Descriptions_v1-1_05jun03.pdf">http://jtrs.army.mil/sections/technicalinformation/req_trace_matrix/SCA_Reqs_Attribute_Descriptions_v1-1_05jun03.pdf</a></i>

**Version 3.0**

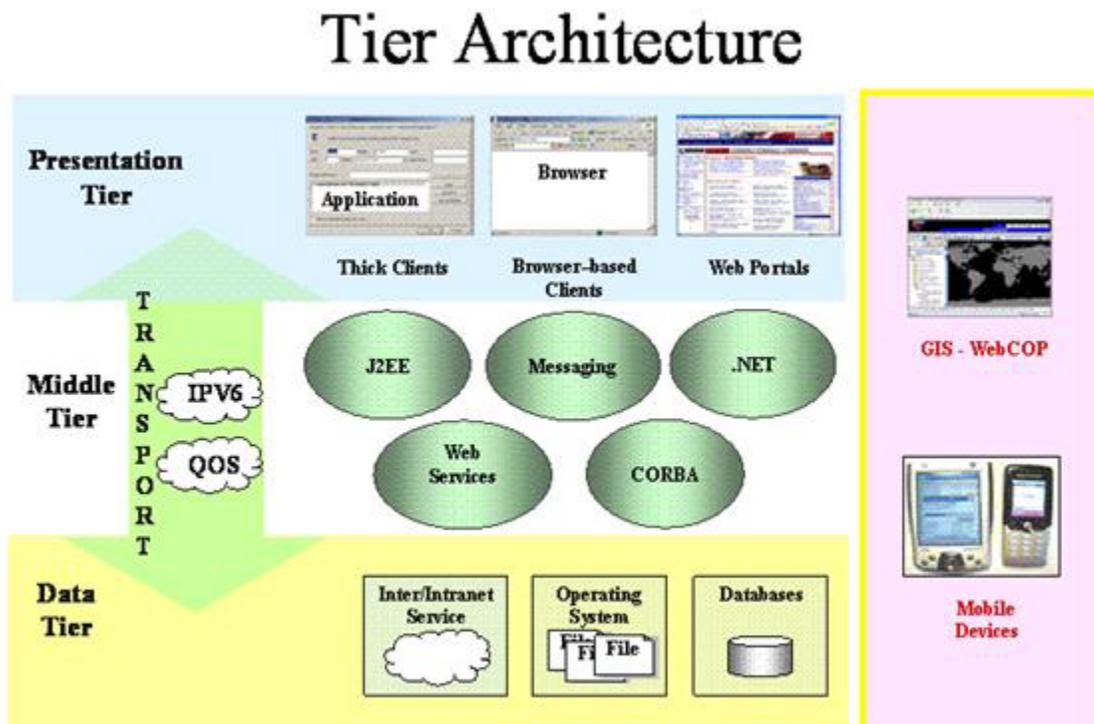
<b>Document (format)</b>	<b>Location</b>
<i>SCA v3.0 (zip)</i>	<i><a href="http://jtrs.army.mil/documents/sca_documents/V3.0/SCA-V3.0.zip">http://jtrs.army.mil/documents/sca_documents/V3.0/SCA-V3.0.zip</a></i>
<i>Specialized Hardware Supplements (zip)</i>	<i><a href="http://jtrs.army.mil/documents/sca_documents/V3.0/SCA-Special-HW-Sup.zip">http://jtrs.army.mil/documents/sca_documents/V3.0/SCA-Special-HW-Sup.zip</a></i>
<i>SCA API Supplement (zip)</i>	<i><a href="http://jtrs.army.mil/documents/sca_documents/V3.0/SCA-V3.0-APIs.zip">http://jtrs.army.mil/documents/sca_documents/V3.0/SCA-V3.0-APIs.zip</a></i>
<i>SCA Security Supplement (zip)</i>	<i><a href="http://jtrs.army.mil/documents/sca_documents/V3.0/SCA-V3.0-Security-Supplements.zip">http://jtrs.army.mil/documents/sca_documents/V3.0/SCA-V3.0-Security-Supplements.zip</a></i>
<i>SCA Developer's Guide (zip)</i>	<i><a href="http://jtrs.army.mil/sections/technicalinformation/developersguide/pdfs/pdfs.zip">http://jtrs.army.mil/sections/technicalinformation/developersguide/pdfs/pdfs.zip</a></i>
<i>API Standardization Process (PDF)</i>	<i><a href="http://jtrs.army.mil/documents/sca_documents/api_policy_files/API_Standardization_Process.pdf">http://jtrs.army.mil/documents/sca_documents/api_policy_files/API_Standardization_Process.pdf</a></i>



---

# Reference implementations

This section discusses applications that incorporate all three tiers.



Future recommendations will include:

- **COE guidelines:** Application structures and development strategies to support orthogonal **NCES** and COE development requirements.
- **Palm OS 5 and above**
- **Wireless environments:** Pocket OS/Windows CE, J2ME

Note that this may be moved to other sections of the **NESSI** documentation, as appropriate.

## GIS display environments

The DoD's net-centric warfare environment requires systems that can be quickly adapted to meet changing requirements. This section explains how to apply the principles of net-centricity to the development and maintenance of GIS systems. Developers must be able to quickly customize, reconfigure, and modify GIS systems to support war-fighting demands. This requires the DoD to use an open-architecture and open-standards approach, and take advantage of new technologies as they arise.

This section contains the following topics:

- *OGC WS architecture, Web Feature and Coverage Services, and OGC API*: Describe an open-source implementation of an open-architecture, open-standards approach for thick and thin **GIS** client applications
- *Examples of GIS open architecture*: Demonstrates this vision using an open-source implementation
- Provides recommendations on how to *implement GIS applications* with this new architecture
- *Migrating to GIS open architecture*: Provides recommendations on how to migrate legacy GIS applications to this new architecture

The examples in this section use open source products, since **NESI** is built around an open source philosophy. However, these products are not necessarily the best for every circumstance. The coding techniques and the guidance provided apply to any open-architecture, open-standards approach.

### Goals

- Support joint interoperability across GIS visualization components through a component-based open standards approach
- Position applications to operate in conjunction with any GIS application with minimal development effort
- Enable applications to take advantage of new technologies in a cost-effective manner.
- React to changing GIS visualization needs while minimizing the impact to programs and budgets
- Facilitate the design, development, maintenance, evolution, and usage of GIS systems that support the **NCW** environment
- Facilitate a cost-effective method to comply with DoD Net-Centric directives as applications migrate into the net-centric environment in the Global Information Grid (**GIG**)

### NESI strategy

- Isolate change and its associated integration and switching costs via a loosely coupled component-based and open-standards approach

- Abstract the data access and rendering portions of GIS applications through an open-standards interface layer
- Make applications agile, so they can use whatever GIS application supports their operational requirements
- Decouple the visualization layer from the rest of the application (when appropriate) with an open-standards GIS layer
- Within applications, decouple the task of rendering control from the task of producing the content; allow each area to evolve independently (see *GIS development communities*)
- Use an open-standards, platform-independent data strategy

## Migration strategies

To minimize the effort of developing an open standards-based architecture, there are two migration strategies:

- *Thin client* development
- *Thick client* development

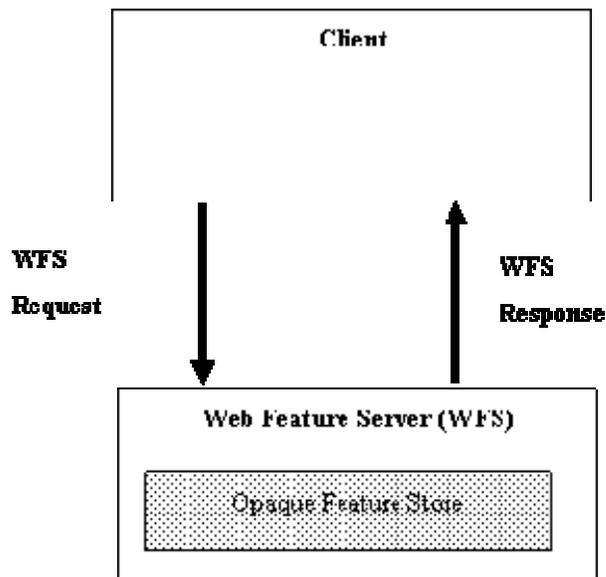
## OGC WS architecture

This section demonstrates how a Service Oriented Architecture (**SOA**) works with a GIS application using the **OGC** open-source framework called OGC (<http://www.opengis.org/>) Web Services (WS) distributed architecture.

There are two complementary elements to this approach:

- OGC Web Services (**OWS**) architecture
- OGC **API** layer called **GO-I**/GEOOBJECTS

The notional OWS architecture appears in the figure below.



These data sources are called *Web Feature Services (WFS)* and *Web Coverage Services (WCS)*.

## Components

This architecture is based on the OGC standards. These standards include five main components:

<i>WFS</i>	Web Feature Server
<i>WMS</i>	Web Map Server
<i>WCS</i>	Web Coverage Server
<i>GRS</i>	GIS Replication Server (WFS and WNS implementation)
<i>GO-1</i>	Geospatial Rendering APIs

## Navy initiatives

There are two Navy initiatives towards this end:

- Joint Open Source WebCOP (*JWC*), based on the Navy Open Source WebCOP (NOSWC)
- Commercial Joint Mapping Tool Kit (C/JMTK)

## Web Feature and Coverage Services

This section discusses the *Web Feature Service (WFS)* and *Web Coverage Service (WCS)*.

## WFS vs. WCS

To help developers choose between **WFS** and **WCS**, **NESSI** identified three broad categories of data based on complexity. To decide what service to program against, evaluate your data and determine where it fits in this chart.

WFS candidates →		← WCS candidates
<b>EASY</b>	<b>MEDIUM</b>	<b>HARD</b>
Point data	Vector data	Gridded data (DTED data, Terrain data)
Icons	Complex styles (WX fronts, Mine field fills)	Imagery data
Tracks	2525 Symbology	
Overlays	Style Layer Descriptions (SLD)	
2525 Symbology	Highly linked data	

## Web Feature Service (WFS)

The **WFS** specification defines interfaces for describing data manipulation operations of geographic features. Data manipulation operations include the ability to:

- Create a new feature instance
- Delete a feature instance
- Update a feature instance
- Get or Query features based on spatial and non-spatial constraints

A WFS describes discovery, query, or data transformation operations. The request is generated on the client and is posted to a web feature server using HTTP. The web feature server then executes the request. The WFS specification uses HTTP as the distributed computing platform, although this is not a hard requirement.

There are two encodings defined for WFS operations:

- **XML** (amenable to HTTP POST/**SOAP**)
- Keyword-Value pairs (amenable to HTTP GET/REST)

## Vendors

A current list of vendors that implement **OWS** services appears on the **OGC** web site:

<http://www.opengis.org/resources/?page=products>

## WFS communication models

The **WFS** specification supports two communication models:

- Stateless Request Reply
- *Pub/Sub*

The WNS is one of the implementation specifications for the Pub/Sub model. Regardless of the model, *URL* format is used and specified in the WFS specification.

At this time there are no open-standard implementations of WNSs. Vendors plan to release implementations once the standard has been ratified.

## WFS data

The Geography Markup Language (GML) passes data back and forth between a Web Feature Server and a client. GML normally communicates geospatial data but also supports other types of data.

## GML

GML expresses feature data in and out of a feature server. The GML standard is at V2.0 and is defined by the *WFS* 1.0 specification. This standard covers the following new topics:

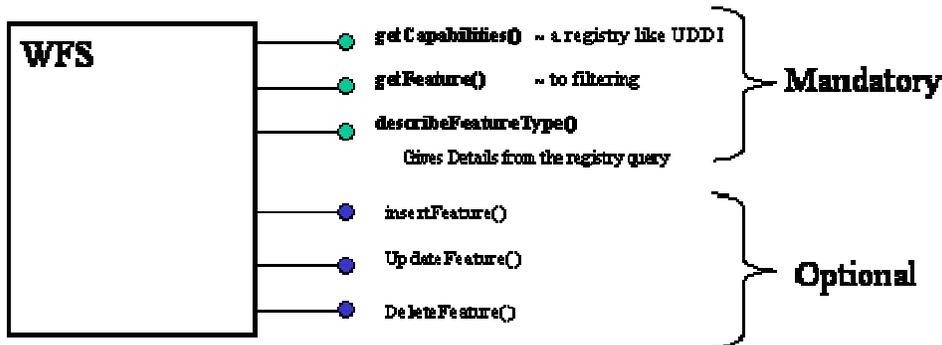
- Coordinates
- Geometry (e.g., polygons)

V3.0 of the specification, which is currently being released, includes Topology, which enables the expression of facts such as "Road A ends at Road B."

## WFS public interfaces

### Static interfaces

The static interface model for the *OGC* Web Service model appears in the figure below.



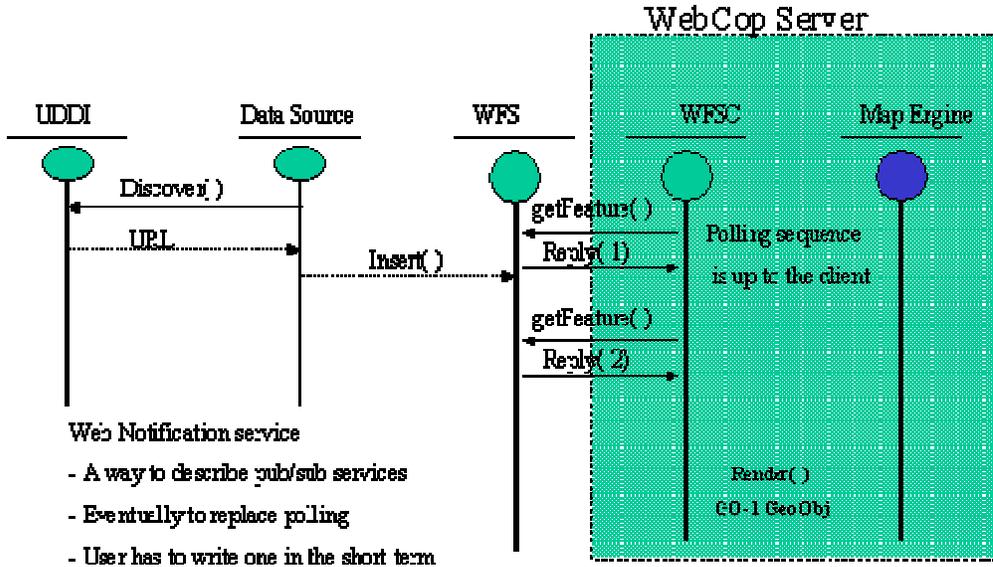
The Transaction and LockFeature operations are also optional.

When writing a *WFS*, you must implement the following operations:

- GetCapabilities
- DescribeFeatureType
- GetFeature

### Dynamic interfaces

The dynamic request/reply interface model for the OGC Web Service model appears in the figure below.



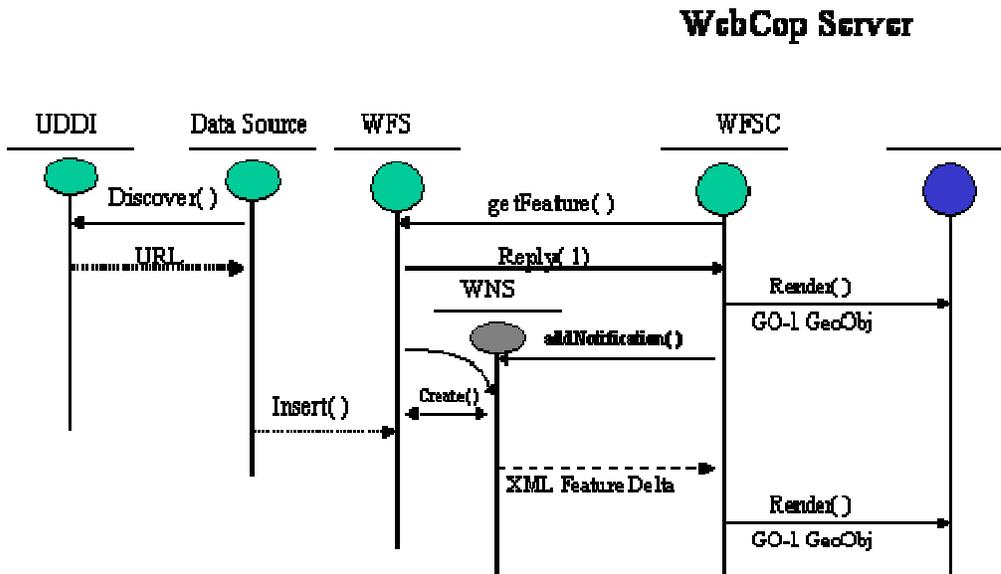
**Dynamic interface updates**

The client gets updates by one of two mechanisms:

- *Notification*: Recommended but not mandatory. Depends on the availability of a WMS implementation.
- *Polling*: Use this method if a WMS implementation is not available.

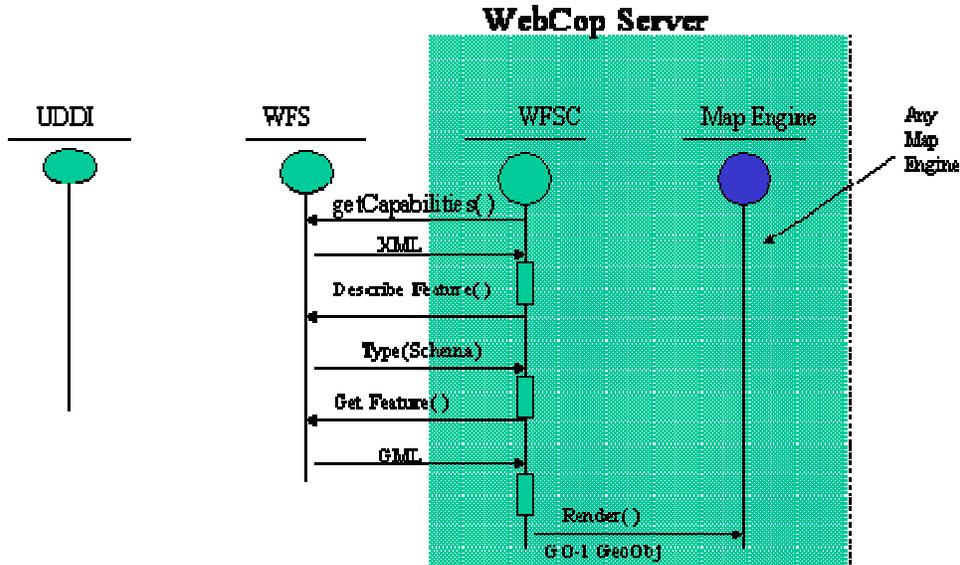
**WFS dynamic interface web notification model**

This model uses the OGC Web Notification Service to send update notifications to registered clients. The Notification interface appears in the figure below.



**WFS dynamic interface web notification polling model**

The polling model interface appears in the figure below.



## Web Coverage Service (WCS)

The **WCS** specification defines interfaces for describing coverage features that associate positions within a bounded space. This service is specifically designed and optimized to handle gridded data such as a raster image or a digital elevation matrix. The operations are similar to the *Web Feature Service*, but handle a different type of data.

For more information, go to <http://www.opengis.org/specs/?page=specs>.

## OGC API

### Overview

**GO-1 APIs** (<http://www.geobject.org>) are the OGC soon-to-be-ratified (expected in Q1CY04) open standards for an interoperable **GIS** rendering abstraction layer.

*Geobject APIs* are the precursor to GO-1 and are not an OGC standard. This initiative was sponsored by **DISA/DARPA JPO** and **SPAWAR**.

### Implementations

Currently there are only Java bindings available for Geobjects. The specifications are available at: <http://www.geobject.org/javaImpl.html>.

Geobject V2.0, under development, covers two new packages for military specific concerns: `org.geobject.mil` and `org.geobject.mil.milstd2525b`. V2.0 will be superseded by the ratified standard and will become GO-1 V1.0. This should be released in Q1CY04.

### Recommendation

Developers should use Geobject V1.3 for development until the GO-1 bindings have been ratified and released.

## Geobject specification

The Geobject specification defines a level of abstraction between the application developers and the implementers of the methods prescribed by the specification. This allows the developers to build components or applications without regard to the underlying implementation.

The Geobject specification:

- Defines abstractions for the drawing canvas and input devices (e.g., mouse and keyboard)
- Supports any coordinate space, such as 2D and 3D Cartesian, polar, and geographical; new coordinate spaces will be added to the specification as they are identified
- Has an extensible, modular design

### Core specification

Currently the core Geobject specification is complete and available in the Unified Modelling Language (*UML*), with a complete Java interface implementation, ready for integration into geographic/geometric products. The core Geobject specification covers key 2D primitive constructs, management of a drawing canvas and input devices, and some support for 3D concepts.

### Subspecifications

Optional subspecifications extend Geobject into areas that some vendor implementations may not support. Open source processes for the following subspecifications are likely to be initiated in the near future:

Subspecification	Probable functionality
3D Geobjects	Supports surfaces, solids, integration of standard 3D models such as VRML, and other 3D concepts
Advanced 2D Geobjects	Supports graphics like Java 2D's General Path, Splines, and other 2D types defined by the Open GIS Consortium ( <i>OGC</i> )
Military Geobjects	Supports at least US DoD graphics standards including MIL-STD-2525x symbology, tactical graphics, and other common military graphics such as unit or naval formations
Immediate Mode Rendering	Adds an optional method for rendering Geobjects using lightweight, transient calls during the physical rendering process; this helps render extensive amounts of graphical information, but is not easily supported by some implementations, such as distributed or client/server map engines

### Implementations

Implementations of language-specific bindings for the Geobject core and subspecifications are being developed in other languages and for other platforms. Other than Java, current proposals

include C++ for Windows, *COM* (which may be the same implementation as C++ for Windows), *.NET*, and Unix/X-Windows.

## Geobject API

Geobject provides a set of common, lightweight abstractions for describing geometric and geographic objects. The following sections provide a high-level overview of the *API*. For more detailed information, consult <http://www.geobject.org/umldoc/1.2/indexLeft.html>.

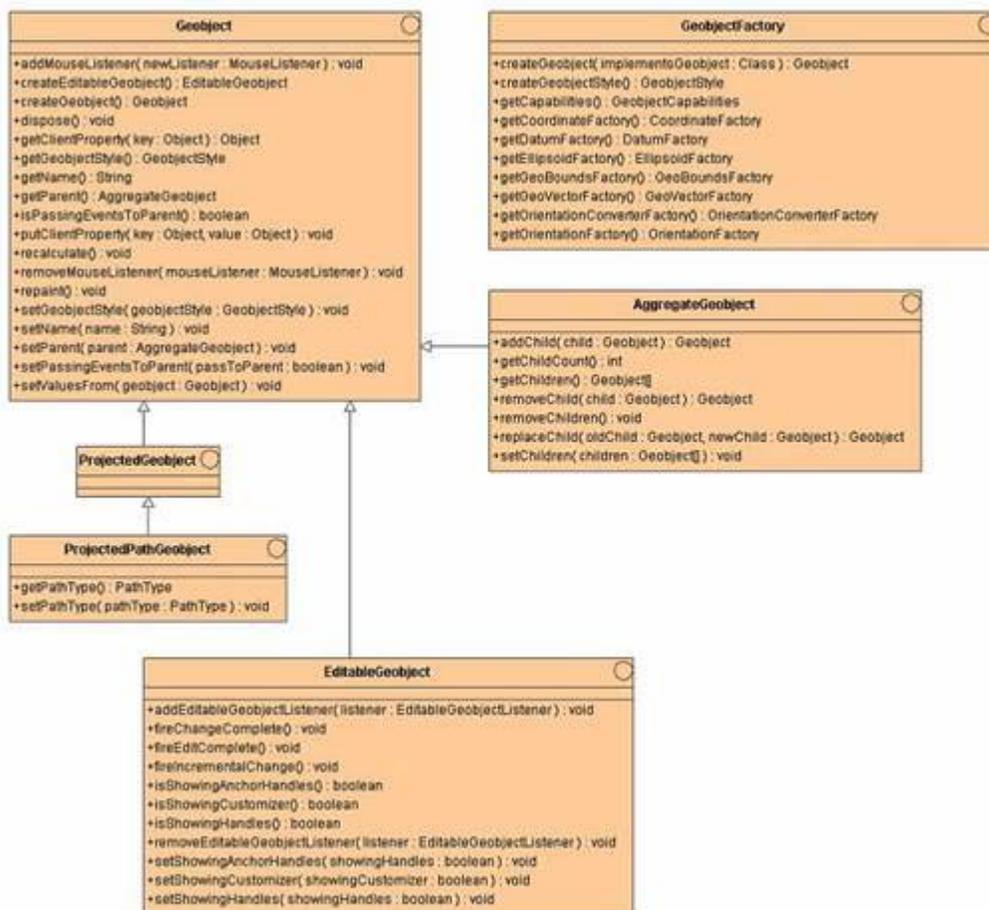
### Geobject package

The main Geobject package contains:

- Five categories of *Geobject abstractions* for describing geometric and geographic objects
- Five categories of *renderable Geobject abstractions*

Specializations of Geobject describe specific geometric/geographic objects such as GeoLabel and GeoPolygon. There are additional abstractions and support classes for organizing, editing, and rendering the objects.

### Class hierarchy UML diagram



**Geometric/Geographic abstraction categories**

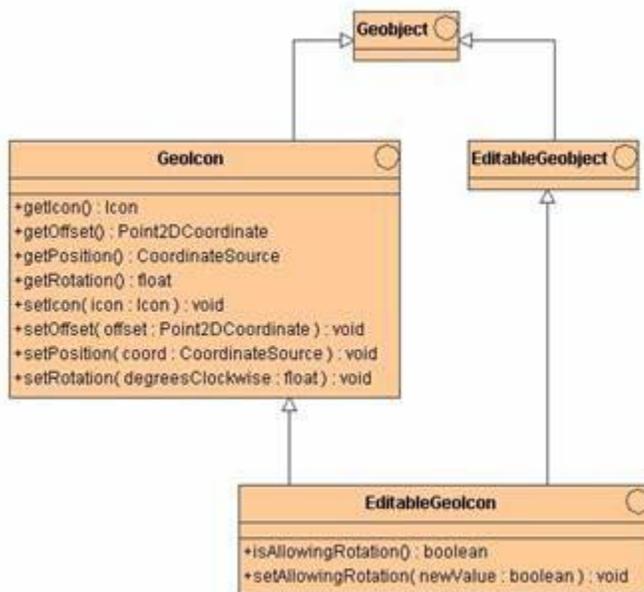
<i>AggregateGeobject</i>	Creates collections of Geobjects so that complex Geobjects can be constructed. For example, an AggregateGeobject containing a GeoIcon, a GeoLabel, and a GeoPolyline could represent a moving object with an icon, a label, and a course/speed indicator.
<i>EditableGeobject</i>	Defines a common abstraction for implementing editable Geobjects, where users can manipulate visual representations of the underlying Geobjects.
<i>OrderedAggregate</i>	Extends the AggregateGeobject interface to enable users to specify a stacking order or Z-order.
<i>ProjectedGeobject</i>	Defines a common abstraction for implementations of projected Geobjects. This means that the line between two vertices will be segmented into multiple sublines and the endpoints of a GeoPolyline line segment will get projected onto the drawing surface as part of the rendering process before a straight line is drawn between the two endpoints.
<i>ProjectedPathGeobject</i>	Defines a common abstraction for objects with vertices. The pixels between vertices can be calculated in several different ways.

**Renderable Geobject abstraction categories**

There are five main renderable Geobject categories:

- GeoIcon
- GeoLabel
- GeoPolygon
- GeoPolyline
- GeoScaledImage

The class hierarchy *UML* diagram for GeoIcon appears in the figure below:



## Examples: GIS open architecture

This section contains examples that demonstrate how the *OGC*'s open architecture works using:

- *Navy Open Source WebCOP (NOSWC)*
- *Commercial Joint Mapping Tool Kit (C/JMTK)*, where the services are ESRI OGC-compliant
- *Command and Control Personal Computer (C2PC)*, which uses an OGC abstraction layer

The examples build on each other to demonstrate *interoperability* across GIS applications that use this OGC framework.

### Navy Open Source WebCOP (NOSWC)

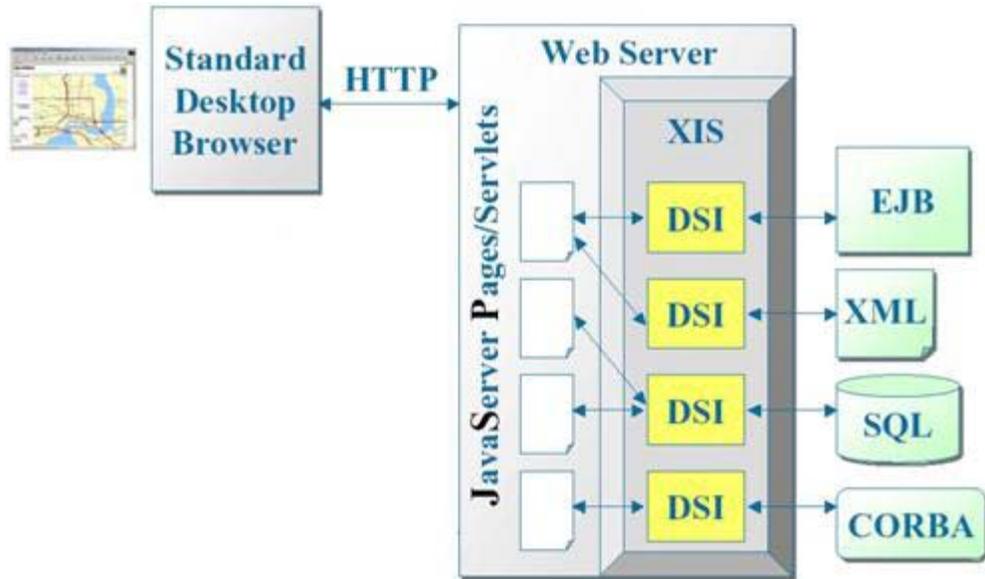
The Navy currently uses the Navy Open Source WebCOP (NOSWC) for command and control. NOSWC is a web-based GIS framework that implements the Web Map Service (*WMS*) specification defined by the Open GIS Consortium.

The WebCOP user interface is based in an Internet browser. Within this request/response paradigm there is no elegant way for the browser to asynchronously respond to server updates. Therefore, the browser has to poll for updates. There is an auto-refresh feature in WebCOP under the Map Options tab above the left navigation tree.

For more detailed information on developing to this WebCOP, go to <https://nesi.spawar.navy.mil/> and navigate to the WebCOP project to download a copy of their developer's manual and the Java *API*.

This example uses the *OGC WFS* implementation that comes with NOSWC. This demonstration uses the OGC WFS with the NOSWC in both a *J2EE* environment using JBoss and a web server environment using Tomcat.

This diagram illustrates the NOSWC architecture:



**Example: WFS to NOSWC in JBoss**

*Disclaimer:* This example uses open-source products, since *NESI* itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

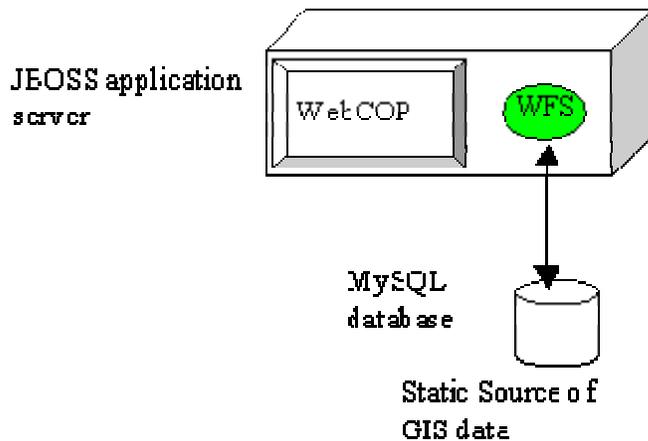
This example demonstrates the use of the *OVS* model in the Navy Open Source WebCOP (NOSWC).

This example uses:

- JBoss v3.2.2.2
- MySQL v4.0.1.6
- WebCOP v3.8.0.2 or later
- Web Feature Server running on Windows 2000 and Windows XP

Before you begin, you must:

- Contact Polexis ([info@polexis.com](mailto:info@polexis.com)) to obtain a copy of the V3.8.0.2 or later WebCOP release, and the the Composeable Feature Server release, which implements the OGC Feature Server Specification
- Download JBoss and MySQL

**Architecture**

With the exception of the NOSWC, the components in this architecture use open source components. This enables you to run the example without purchasing a product and licenses. (This architecture currently operates in a Navy *LOE*.) The WebCOP runs in the same application server as the WFS for ease of running the example.

**Set up the JBoss example**

Perform the procedures in this section to set up the example. You will be installing the non-COE version of the NOSWC in JBoss.

**To set up the environment:**

1. Install JBoss.
2. Create a JBoss server instance called NESIServer.
3. Follow these steps to install NOSWC:
  - a. Create a target installation directory for NOSWC. This directory must not be in the JBoss directory structure.
  - b. From the installation CD, copy the **docs** directory and the non-COE installation files **releaseroot.zip** and **webcop.war** to the target installation directory.
  - c. Using WINZIP, extract the **releaseroot.zip** file in that directory.
  - d. Using WINZIP, expand the **webcop.war** file into a temporary directory.
  - e. Follow the installation instructions located in the **docs** directory for setting up the root files and the data directory.
  - f. When you finish the installation, open **web.xml** and set the `servlet.dir` parameter to the name of the WebCOP server instance directory in the JBoss deploy directory.
  - g. Comment out the `Security-constraint` section, then save and close **web.xml**.
  - h. Open **baselayers.xml** and comment out the TMS live layer that is identified with the tag:

```
<!-- Layer queryable="0">
<Name>TMS Live</Name>
```

- i. Comment out the TDBM live layer that is identified with the tag:
 

```
<!-- Layer queryable="0">
      <Title>TDBM: Live</Title>
      <Abstract>Live TDBM Producers</Abstract>
```
4. Follow these steps to deploy NOSWC to JBoss:
  - a. Create a subdirectory under **<WebCOP server instance dir>\deploy** called **webcop.war**.
  - b. Copy the expanded directory of the **webcop.war** file into the **\deploy\webcop.war** subdirectory of the JBoss server instance folder, as described in the JBoss install section.
  - c. Delete any files in **\tmp\deploy**.
  - d. Delete any files in **\work\MainEngine\localhost**.
5. Follow these steps to run NOSWC in JBoss:
  - a. Start the JBoss application server as described in the JBoss install section.
  - b. Start the web browser and enter the URL `http://localhost:8080/webcop`.  
**Note:** The port number is the port number you used during the JBoss configuration steps, as described in the JBoss install section.
  - c. Once the WebCOP is up, go to the left menu of the WebCOP and click on **TDBM Producers > TDBM replay**, then check the **Replay** check box. The tracks should appear.
  - d. Shut down the JBoss server.

### To install the Web Feature Server:

1. Install the MySQL database.
2. Use either the MySQLGuiClient or the MySQL command interpreter to create a database called `wfs`.
3. Create a new JBoss server instance, called `WebFeatureServer`, and configure the ports as described in Installing JBoss.
4. Follow these steps to load the WFS components into this server instance:
  - a. Create two subdirectories under this server instance: **file.war** and **WFL.war**.
  - b. From the WFS distribution, copy **file.war** to the **file.war** subdirectory. Copy **WFL.war** to the **WFL.war** subdirectory. Expand each **war** file.
  - c. Change to the **WFL.war\WEB-INF** subdirectory.
  - d. Modify the `ABSOLUTE_FEATURES_DIRECTORY` tag in **web.xml** to point to the path of the WebCOP data directory.
  - e. Save the file.
  - f. Change to the **file.war\WEB-INF** subdirectory.
  - g. Modify the `DATADIR` tag in **web.xml** to point to the path of the WebCOP data directory.
  - h. Modify the `LOCALHOST` tag in **web.xml** to your IP address.

- i. Save the file.
5. Follow these steps to select the HPAC feature for this example:
  - a. Change to the WebCOP data root directory.
  - b. Change to the **features** subdirectory.
  - c. Keep the **HPAC** subdirectory and move the other subdirectories to a backup directory .
6. Follow these steps to configure the WebCOP to accept WFS input:
  - a. Change to the JBoss WebCOP deploy directory: **<JBoss install\_dir>\server\NESIserver\deploy\webcop.war\WEB-INF.**
  - b. Make a copy of the existing **baselayers.xml** file as a backup.
  - c. Add the following code to **baselayers.xml**:

```

<Layer queryable="0">
  <Name>Web Feature Server</Name>
  <Title>Web Feature Server</Title>
  <Abstract>Web Feature Server</Abstract>
  <SRS>
    EPSG:4326 AUTO:42400 AUTO:42402 AUTO:42403
    AUTO:42404 AUTO:42405 AUTO:42406 AUTO:42407 AUTO:42408
  </SRS>
  <LatLonBoundingBox
    minx="-180.0"
    miny="-90.0"
    maxx="180.0"
    maxy="90.0"/>
  <DataURL/>
  <Style>
    <Name>default</Name>
    <Title>Default</Title>
  </Style>
  <ScaleHint min="0.0" max="0.0"/>

  <Host>198.253.7.109</Host>
  <Port>8085</Port>
  <Path>/WFS/WFSServlet</Path>
  <ExposeFeaturesOfType>HPAC</ExposeFeaturesOfType>
</WMSLayerLoader>
</Layer>

```

- d. Modify the host IP and port number to match your JBoss environment:
 

```

<Host>198.253.7.109</Host>
<Port>8085</Port>

```

The port number should be the JBoss port number you used to configure this instance of the JBoss server for the web feature server.
- e. Save the file.
7. Make a copy of the existing **web.xml** file as a backup.
8. Add this code to **web.xml**:

```

<context-param>
  <param-name>LOCALHOST</param-name>

```

```

    <param-value>198.253.7.109</param-value>
    <description>The IP address or FULL MACHINE NAME of the
machine this web app
is running on ('localhost' will NOT WORK!!)
    </description>
</context-param>

<context-param>
  <param-name>WFS_PATH</param-name>
  <param-value>http://198.253.7.109:8085/WFS/WFSServlet</param-
value>
  <description>The path to the WFSServlet to post insert layer
requests
  </description>
</context-param>

```

9. Modify the LOCALHOST param to point to your IP address:

```

<param-value>http://198.253.7.109:8085/WFS/WFSServlet</param-
value>

```

10. Modify the WFS\_PATH to point to your IP address. The port number is the JBoss port number for the web feature server instance.
11. Save the file.

The example is now configured and ready to run.

### ***Run the JBoss example***

1. Open a command window.
2. Enter this command to start the MySQL database:

```
C:> net start NESI_SQL
```

3. Enter this command to start the Web Feature Server instance:

```
<JBoss install_dir>\bin\run -c=WebFeatureServer
```

4. Wait for it to completely start up before proceeding.
5. To test basic WFS operation, point the browser to  
<http://localhost:8080/WFS/WFSServlet?REQUEST=GetCapabilities>.

**Note:** The port number is the number you used to configure the JBoss server instance.

6. This example uses the GetCapabilities API call to determine available features. This figure shows the output:

```

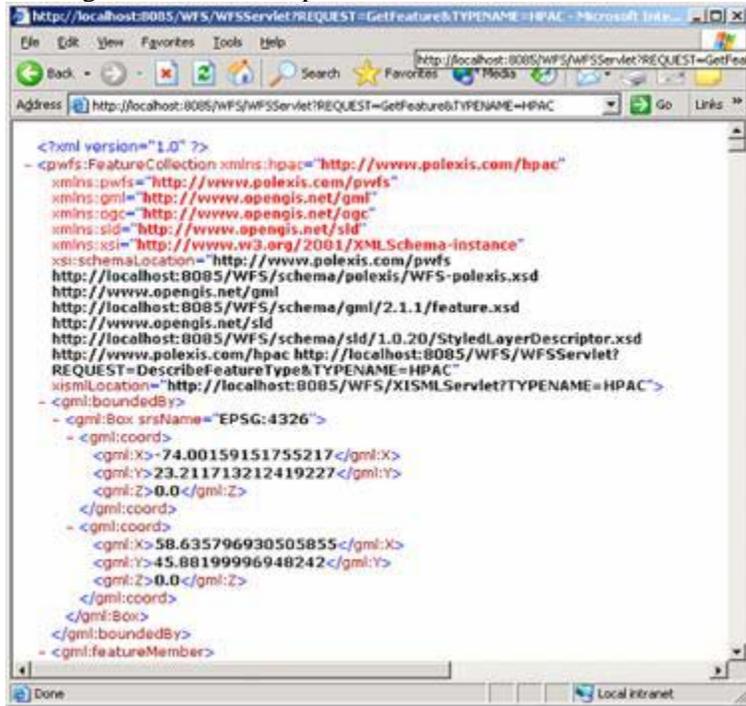
<?xml version="1.0" ?>
- <WFS_Capabilities xmlns="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs
  http://localhost:8085/WFS/schema/wfs/1.0.20/WFS-capabilities.xsd">
- <Service>
  <Name>Polexis WFS</Name>
  <Title>Polexis Web Feature Service</Title>
  <Abstract>Web Feature Service maintained by Polexis, Inc.</Abstract>
  <OnlineResource>http://www.polexis.com</OnlineResource>
</Service>
- <Capability>
- <Request>
  - <GetCapabilities>
  - <DCPType>
  - <HTTP>
  <Get
    onlineResource="http://localhost:8085/WFS/WFSServlet" />
  </HTTP>
  </DCPType>
  - <DCPType>
  - <HTTP>
  <Post
    onlineResource="http://localhost:8085/WFS/WFSServlet" />
  </HTTP>
  </DCPType>
</GetCapabilities>
- <DescribeFeatureType>
- <SchemaDescriptionLanguage>

```

7. To test the example in the WFS, HPAC, point the browser to:  
<http://localhost:8080/WFS/WFSServlet?REQUEST=GetFeature&TYPENAME=HPAC>

**Note:** The port number is the number you used to configure the JBoss server instance.

8. This figure shows the output:



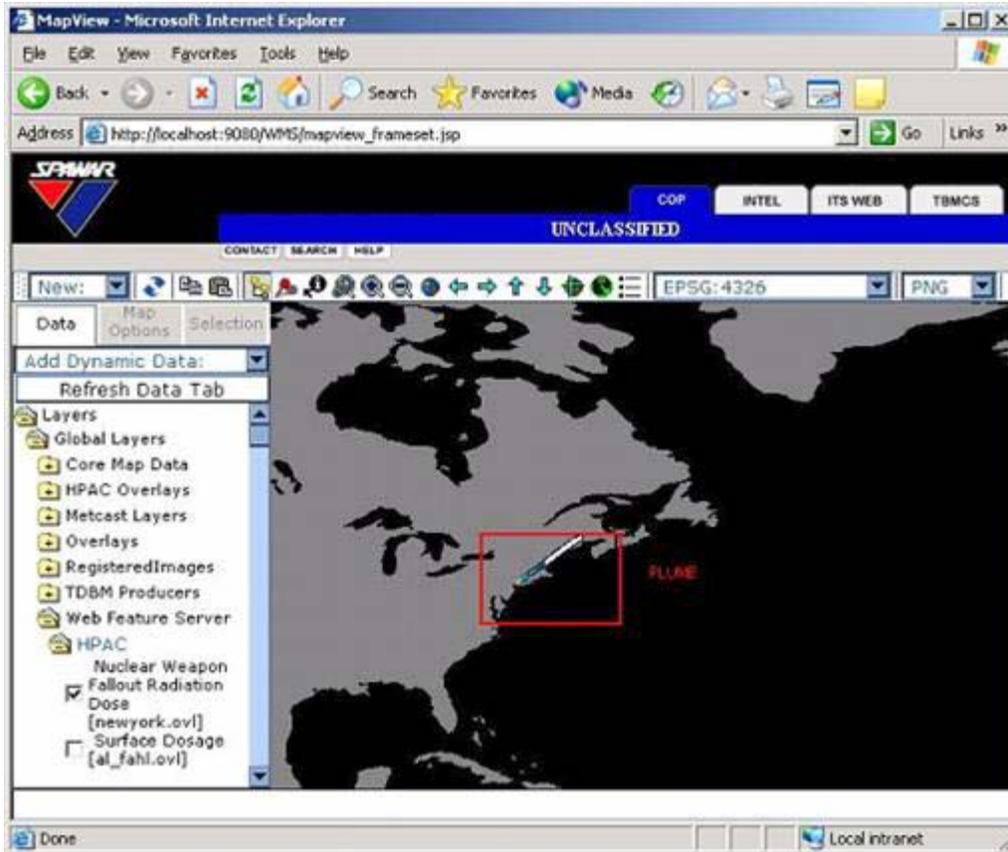
9. Enter this command to start the WebCOP server instance:

```
<JBoss install_dir>\bin\run -c=NESIServer
```

10. Wait for it to completely start up before proceeding.
11. Open a web browser and access the WebCOP with this URL:  
<http://localhost:8080/webcop/>

**Note:** The port number is the number you used to configure the JBoss server instance.

This figure shows the output as it appears after you zoom in:

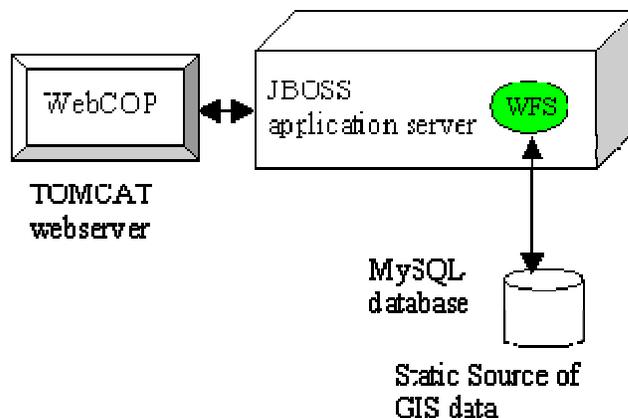


**Example: WFS to NOSWC in Tomcat**

*Disclaimer:* This example uses open-source products, since *NESI* itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

This example demonstrates the use of the open standards APIs and *OWS* model in the NOSWC. It builds off the *JBoss example* and uses the same components, but the WebCOP runs in the Tomcat web server (v4.1.18) rather than the JBoss server.

**Architecture**



### Set up the Tomcat WFS example

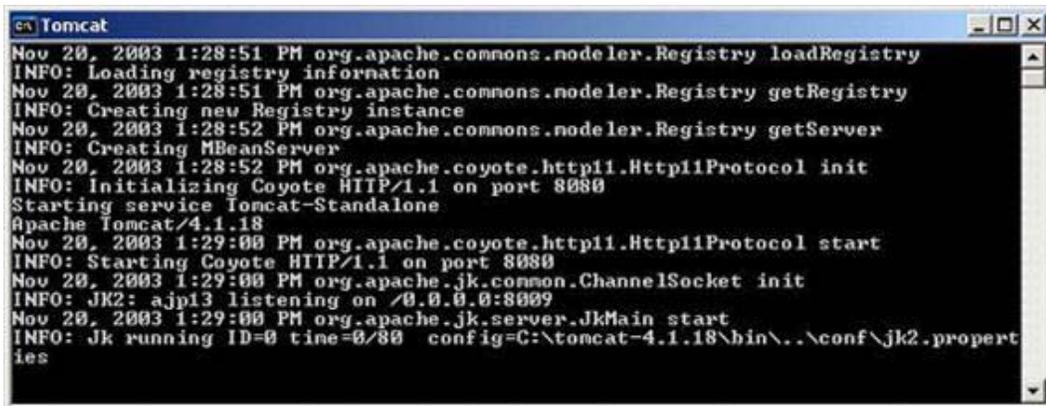
Perform the procedures in this section to set up the example.

#### To set up the Tomcat environment:

1. Go to <http://jakarta.apache.org> and download the Tomcat binaries.
2. Expand the downloaded zip file to a target installation directory.
3. Set the JAVA\_HOME environment variable to point to your Java environment.
4. Open **conf/server.xml** and configure the Tomcat ports.
5. Open a command line and start Tomcat by executing the command:

```
<Tomcat install_dir>\bin\startup.bat
```

This figure illustrates a successful startup:



```

Tomcat
Nov 20, 2003 1:28:51 PM org.apache.commons.modeler.Registry loadRegistry
INFO: Loading registry information
Nov 20, 2003 1:28:51 PM org.apache.commons.modeler.Registry getRegistry
INFO: Creating new Registry instance
Nov 20, 2003 1:28:52 PM org.apache.commons.modeler.Registry getServer
INFO: Creating MBeanServer
Nov 20, 2003 1:28:52 PM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on port 8080
Starting service Tomcat-Standalone
Apache Tomcat/4.1.18
Nov 20, 2003 1:29:00 PM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on port 8080
Nov 20, 2003 1:29:00 PM org.apache.jk.common.ChannelSocket init
INFO: JK2: ajp13 listening on /0.0.0.0:8009
Nov 20, 2003 1:29:00 PM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 tme=0/80 config=C:\tomcat-4.1.18\bin\..\conf\jk2.properties
  
```

6. Open a web browser to test the Tomcat installation. Enter this URL:  
<http://localhost:8080/>.

The Apache home page should appear as shown in the figure below:



7. Stop the Tomcat web server by entering **Ctrl-C** on the command line.
8. Follow these steps to install the Navy Open Source WebCOP into Tomcat. This enables you to operate with the **WFS** from the JBoss example.
  - a. Change to the `<JBoss>\server\NESIserver\deploy\webcop.war` directory.
  - b. Create a **WAR** file that contains the WFS configuration from the *JBoss example* with this command:
 

```
jar -cvf ..\webcop.mywar *
```
  - c. Copy **webcop.mywar** to `<Tomcat install_dir>\webapps` and rename it to **webcop.war**.
9. There is an issue with XML parsers between Tomcat and WebCOP. To avoid this, you need to replace some JAR files. Follow these steps:
  - a. Change to `<Tomcat installation directory>\common\endorsed`
  - b. Rename **xercesImpl.jar** and **xmlParserAPIs.jar** to backup file names.
  - c. Copy the **xercesImpl.jar** and **xmlParserAPIs.jar** files from the WebCOP deploy directory to this directory as shown below:
 

```
Copy <Tomcat install_dir>\webapps\webcop\WEB-INF\lib\xercesImpl.jar *
Copy <Tomcat install_dir>\webapps\webcop\WEB-INF\lib\xmlParserAPIs.jar *
```

### Run the Tomcat WFS example

1. Open a command window.

2. Enter this command to start the MySQL database:

```
C:> net start NESI_SQL
```

3. Enter this command to start the Web Feature Server instance:

```
<JBoss install_dir>\bin\run -c=WebFeatureServer
```

4. Wait for it to completely start up before proceeding.

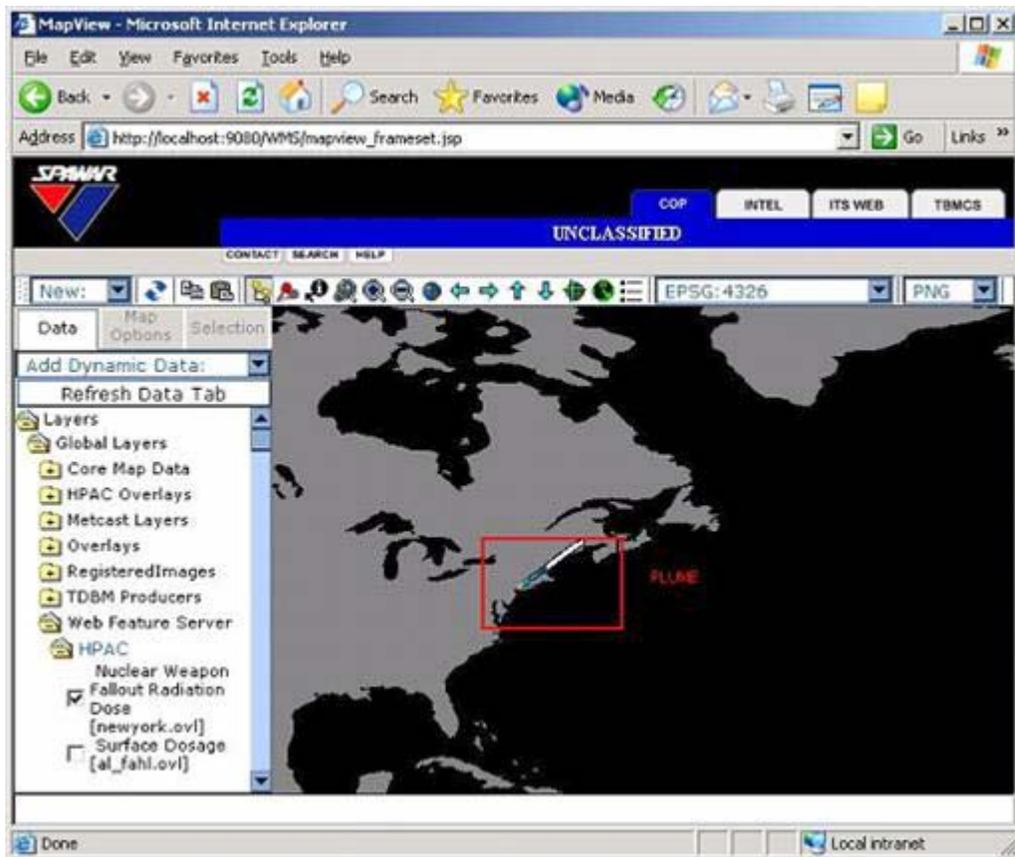
5. Enter this command to start the Tomcat web server:

```
<Tomcat install_dir>\bin\Catalina.bat run
```

6. Open a web browser and access the WebCOP with this URL:

```
http://localhost:8080/webcop.
```

This figure shows the output:



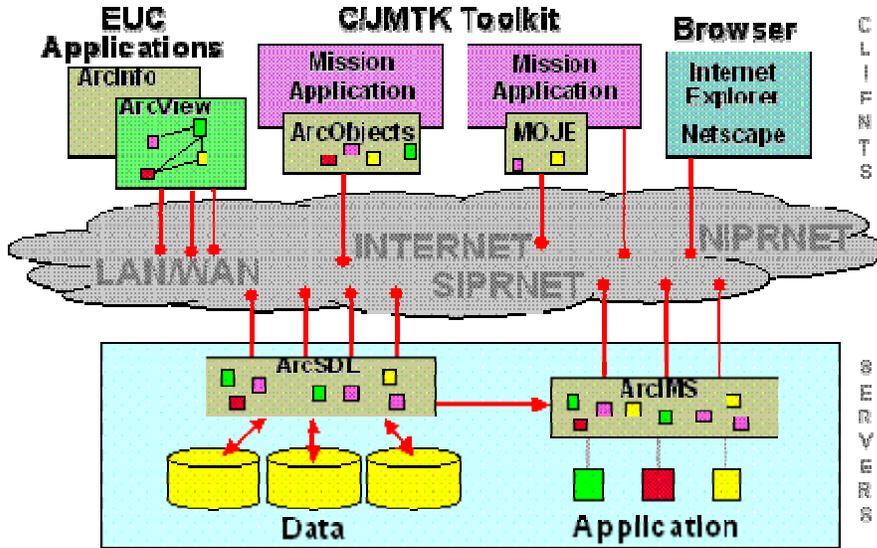
### Commercial Joint Mapping Tool Kit (C/JMTK)

C/JMTK is a *NIMA*-funded, congressionally mandated initiative for GIS applications. The initiative began in June 2002. It is a commercial replacement for the government-developed *JMTK* and is based on the *ESRI* suite of products. It is expected to undergo DISA certification in Q4 2003.

Its goals are:

- Be the standard geospatial tool for **COE/GES**
- Support the next generation of DoD C2 capabilities and systems
- Provide a standard architectural framework and common software components and services to the entire GIS community
- Foster software reuse
- Reduce development and integration costs

**Architecture**



**References**

For sample code, developer's guidance, and help desk support, visit the C/JMTK web site at <http://www.cjmtk.com/>. The government point of contact is Sue Riley, who can be reached at [rileys@nima.mil](mailto:rileys@nima.mil).

**C/JMTK licensing**

There are currently four license options:

<i>Toolkit</i>	Replaces the current <b>JMTK</b> . Licensed through NIMA, this option provides unlimited license use and 10 years of life cycle support (maintenance, training, technical support, and upgrades) to the <b>COE/GES</b> community. This community is defined <i>as</i> the current and future C2I programs, including DODIID and <b>GCSS</b> . This license is not retroactive nor transferable to ESRI products.
<i>Extended User Community (EUC)</i>	Provides commercial application licenses for software from participating team members ( <b>ESRI</b> , ERDAS, AGI) to those who want to be compatible with COE. Licenses and maintenance costs are the responsibility of the Extended User through a Basic Purchase Agreement.
<i>Foreign Military Sales (</i>	Covers the toolkit and applications. Funding and FMS approval

<b>FMS)</b>	authority are the responsibility of the sponsoring organization.
<i>ESRI Products (as for EUC)</i>	Covers the license between the vendor and the user regardless of whether the software is also in the toolkit. The cost is absorbed by the user ( <b>BPA</b> pricing for EUC).

### **C/JMTK vs. JMTK**

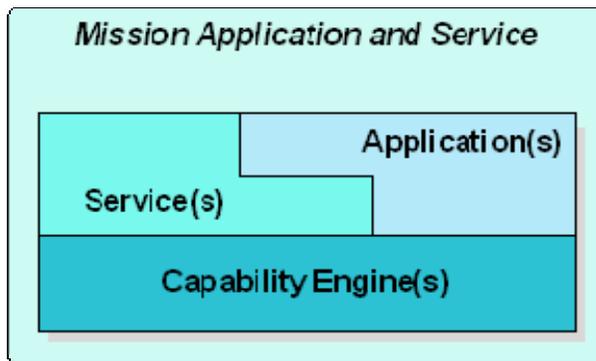
	<b>JMTK</b>	<b>C/JMTK</b>
<i>2D Map Display</i>	✓	✓
<i>3D Map Display</i>		✓
<i>Image Processing</i>		✓
<i>DBMS Integration</i>	partial	✓
<i>Web-enabled</i>		✓
<i>Enterprise Solution</i>		✓
<i>Single Scalable Architecture</i>		✓

### **C/JMTK architecture**

The C/JMTK is an open, layered, *services-oriented architecture*.

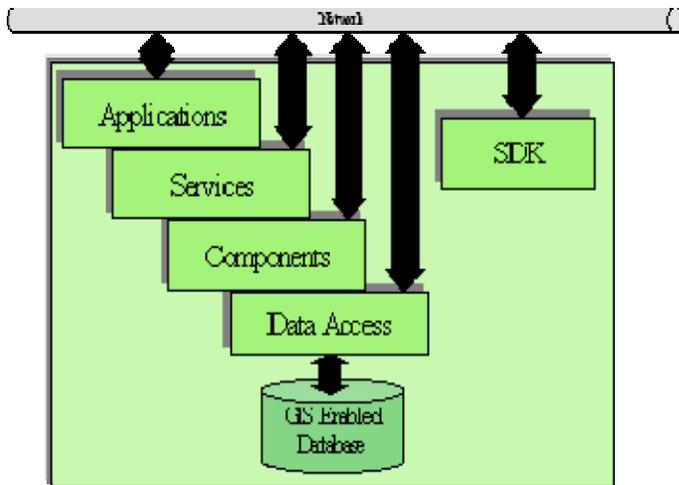
It has three layers:

- Application(s)
- Service(s)
- Engine(s)

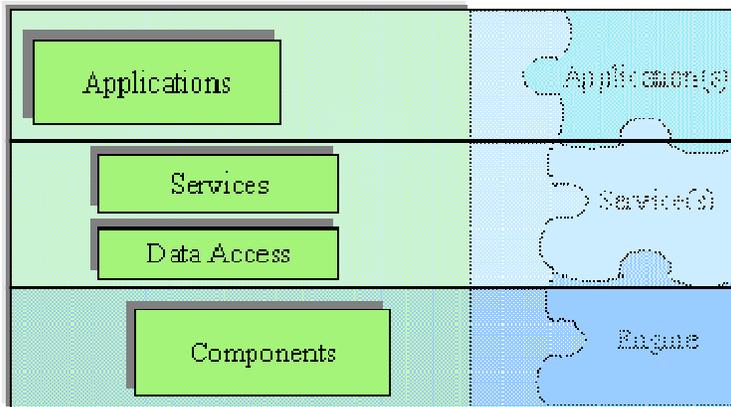


The architecture enables *Net-Centric* system development and software *interoperability* by separating the capabilities from the applications. It promotes software unit reuse at two levels:

- Functional primitives (Engine)
- Mission services (Service)



**Layering and associated services**



**C/JMTK features and capabilities**

**Image processing**

Image processing uses **ESRI's** raster engine from ERDAS. It supports:

- Import/export of multiple formats
- Pyramid layers
- Band/RGB management
- Histogram manipulation
- Contrast stretching
- Layer transparency
- Brightness
- Contrast
- Swipe tool
- Mosaicing

**Sensor modeling**

Sensor modeling uses ESRI's AGI sensor model component. It supports:

- Subsatellite point
- Satellite ground-track
- Satellite intervisibility
- Sensor footprints
- Sensor prediction
- Radar prediction

### **3D capabilities**

C/JMTK offers the following 3D capabilities:

- Create three-dimensional visualizations
- Query three-dimensional data
- Generate fly-through simulations
- Build surface models
- Interactive perspective viewing
- Line-of-site analysis
- Viewshed analysis

### **Large data set solutions**

Using ESRI's ArcGlobe (under development) component to handle large data sets, C/JMTK creates a "Whole Earth" solution. It supports an intelligent scale-dependent paging mechanism that runs on Windows platforms.

### **C/JMTK toolkit components**

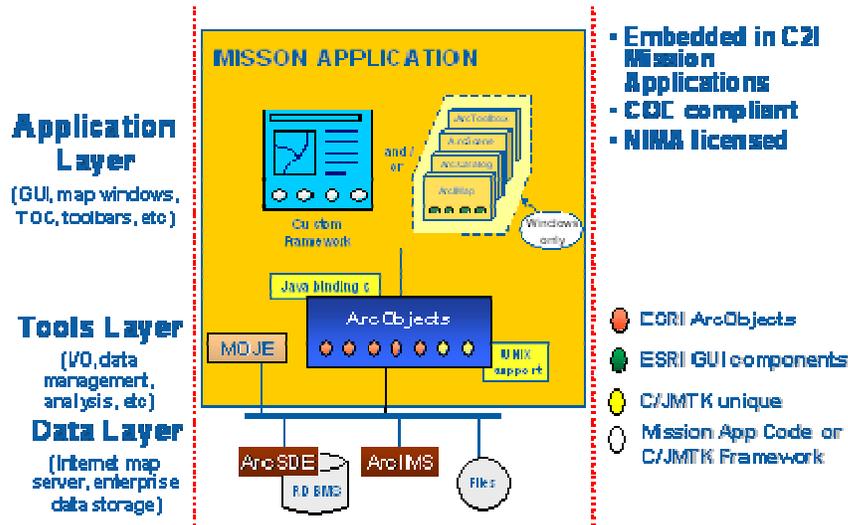
#### **Components**

C/JMTK v1.0 is based on the ESRI ArcGIS v9.0 set. The main components of C/JMTK and their associated implementations appear in this table and are described in more detail below:

<b>C/JMTK component</b>	<b>ESRI component</b>	<b>Platform</b>
<i>Thick Client</i>	ArcObjects extended by ArcSDE, Spatial Analyst, and 3D Analyst	Windows 2000: Native COM implementation Solaris: Uses Mainsoft MainWin software to port ArcObjects
<i>Thin Client</i>	MapObjects Java Standard Edition (or browsers such as IE/Netscape)	Windows 2000 and Solaris
<i>Application Server</i>	<i>ArcIMS</i>	Windows 2000 and Solaris
Data Server	<i>ArcSDE</i>	Windows 2000 and Solaris

### Toolkit-Architecture mapping

This figure shows the mapping of the toolkit to the architecture:



### Thick client components

The thick client uses the ESRI product *ArcObjects*. This is a fully compliant C/JMTK solution and is an integral part of the ESRI desktop client. It is based on Microsoft's Component Object Model (COM).

### JAVA bindings

C/JMTK uses Intrinsyc's J-Integra to map to Java. It is a pure Java solution and supports both custom interfaces as well as idispatch and connections points.

### Thin client components

The thin client uses the ESRI product Map Objects Java Edition (*MOJE*) to create a client customization framework. There are two toolkits:

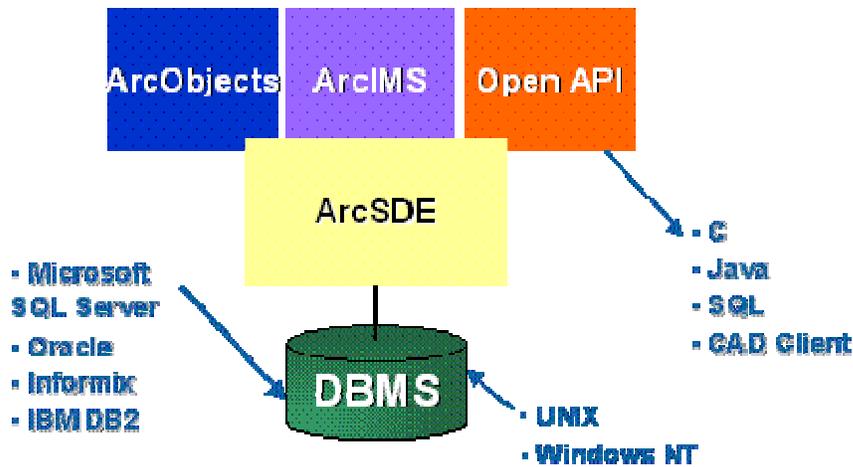
- Display/query client toolkit
- ArcIMS client toolkit

### Server components

There are four server components;

- *ArcSDE* service, the Spatial Database Engine
- *ArcIMS* service, the Internet Map Server
- *ArcObjects*, the component **API**
- Open API

### Server architecture



### ArcSDE component

ArcSDE is the Geodatabase component. It centralizes the management of geographic information.

Features include:

- Fast, multi-user access
- Optimization for network enterprise implementations
- Support for large, continuous databases
- Versioning
- *TCP/IP*, with no need for NFS mounts
- Interaction with a backend database such as SQLServer or ORACLE
- Multi-resolution pyramids that organize large imagery and NIMA image products

The Geodatabase data model:

- Uses features as objects
- Geometry
- Attributes
- Behavior (rules, methods, relationships)
- User-definable

### ArcObjects component

ArcObjects provides data readers for data access using native formats such as *VPF*, *RPF*, *NITF*. It is not optimized for network systems.

### ArcIMS component

Features include:

- Internet mapping
- Distributed servers
- True services

- Developer options
- Client-side tools
- Scalable
- Industrial-strength
- Internet-tested
- Advanced GIS browsers
- Integrates data from multiple sources

**Example: ESRI ArcIMS and web client in Tomcat**

*Disclaimer:* This example uses open-source products, since *NESI* itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

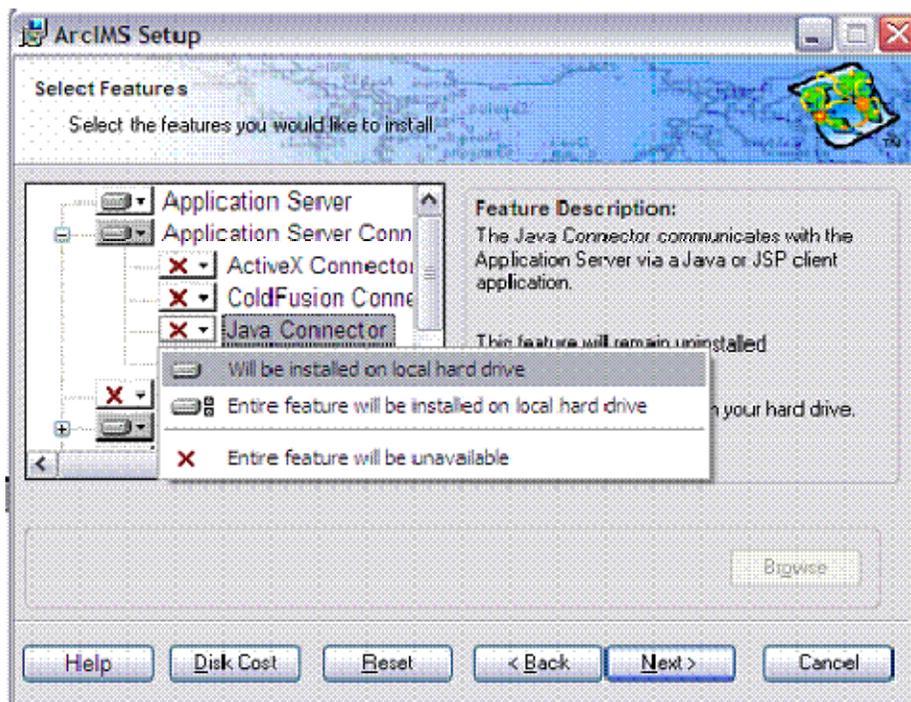
This example demonstrates basic WFS capability with the ESRI WFS under *ArcIMS* on Tomcat. It uses the same environment as the *WebCOP* example.

**Set up the example**

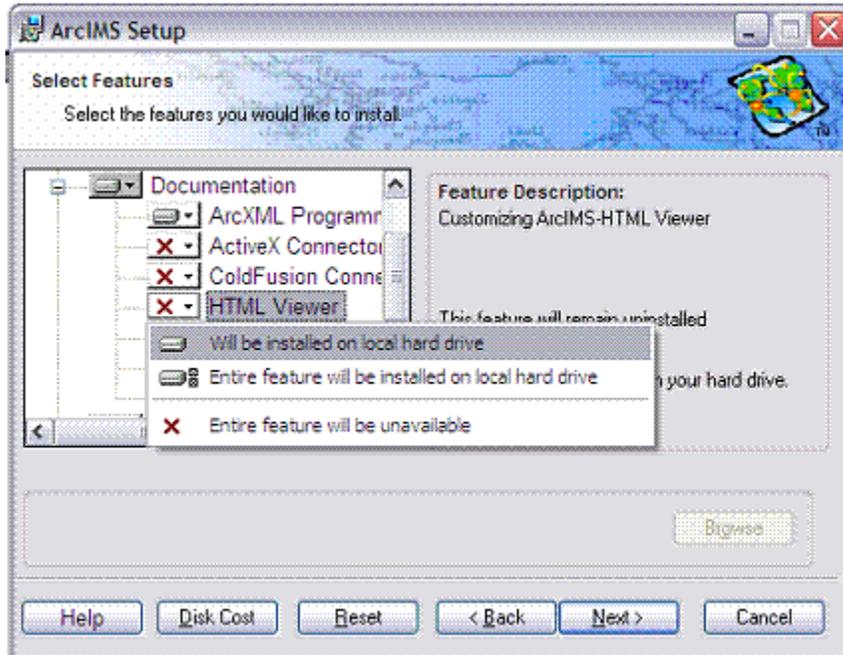
Make sure the Java Development Kit is installed on your computer.

**To install and configure ArcIMS:**

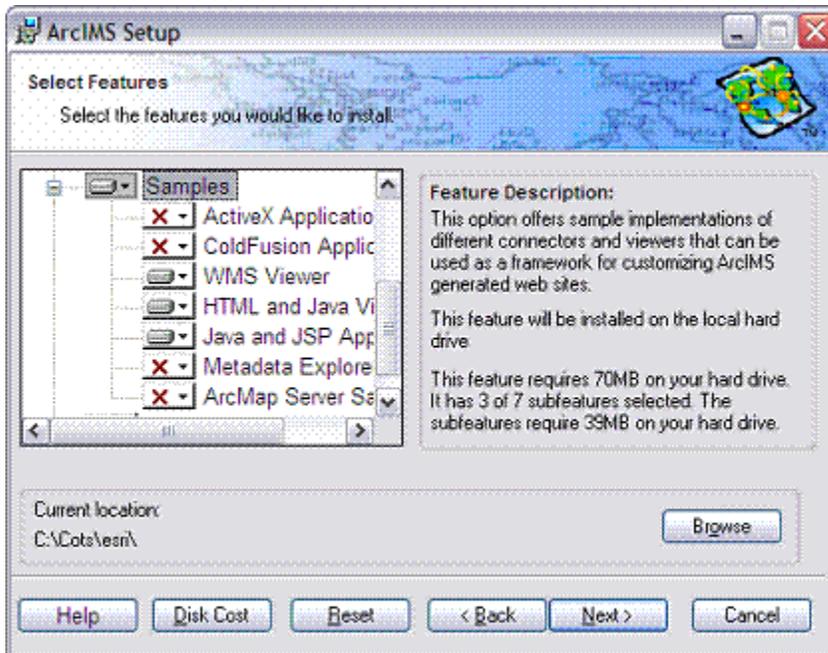
1. Go to <http://www.esri.com/products.html> and obtain ArcIMS for Windows. Start the installation.
2. In the Select Features window, select **Application Server Connectors > Java Connector**.



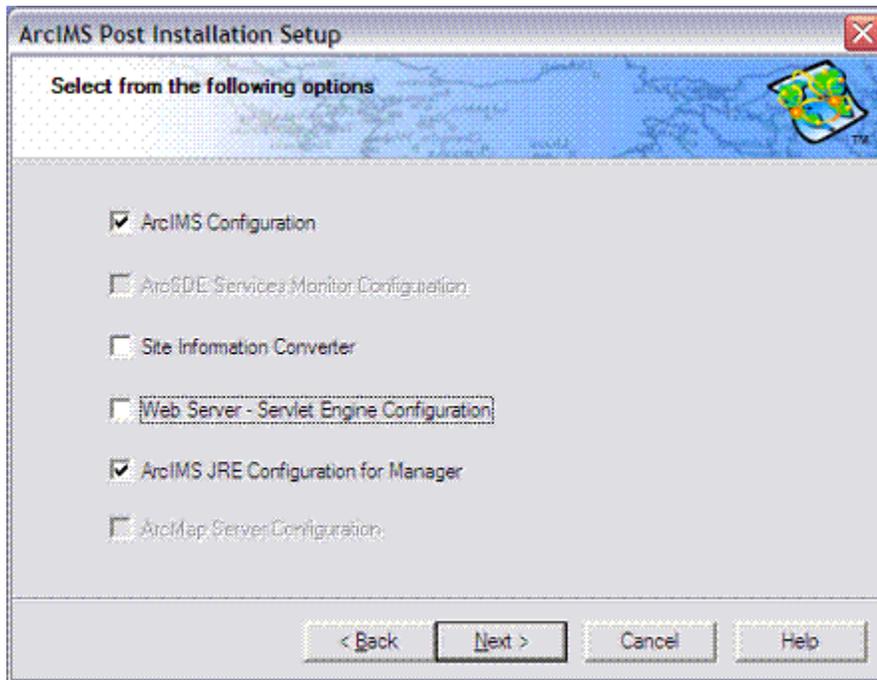
3. Select **Documentation > HTML Viewer**.



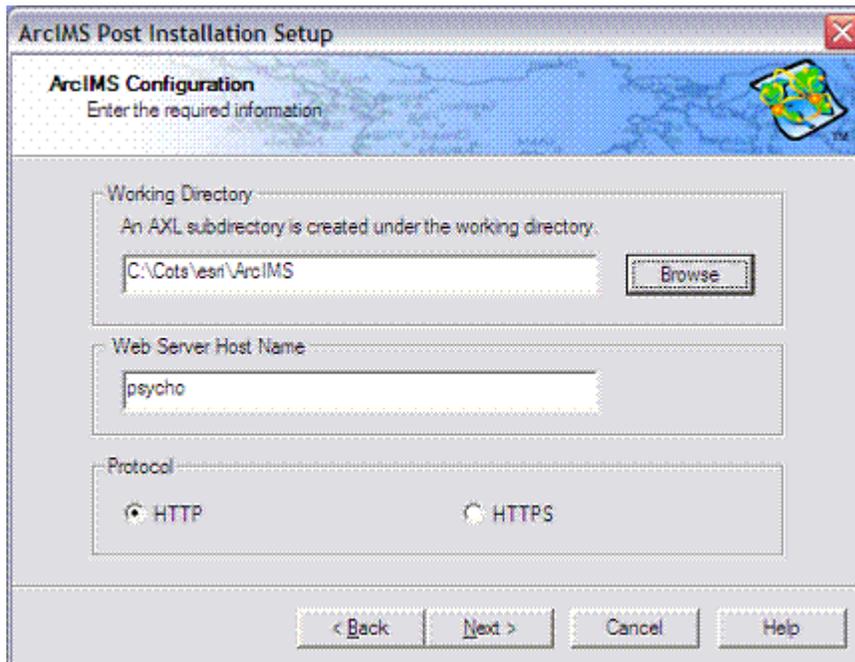
4. Turn off the **Metadata** feature.
5. Select **Samples > WMS viewer** and **Samples > Java and JSP Applications**.



6. Select **Tutorial Data** and click **Next**. After the installation finishes, it enters a post-installation phase.
7. Select the **Custom** option, then select **ArcIMS Configuration and ArcIMS JRE Configuration for Manager**.

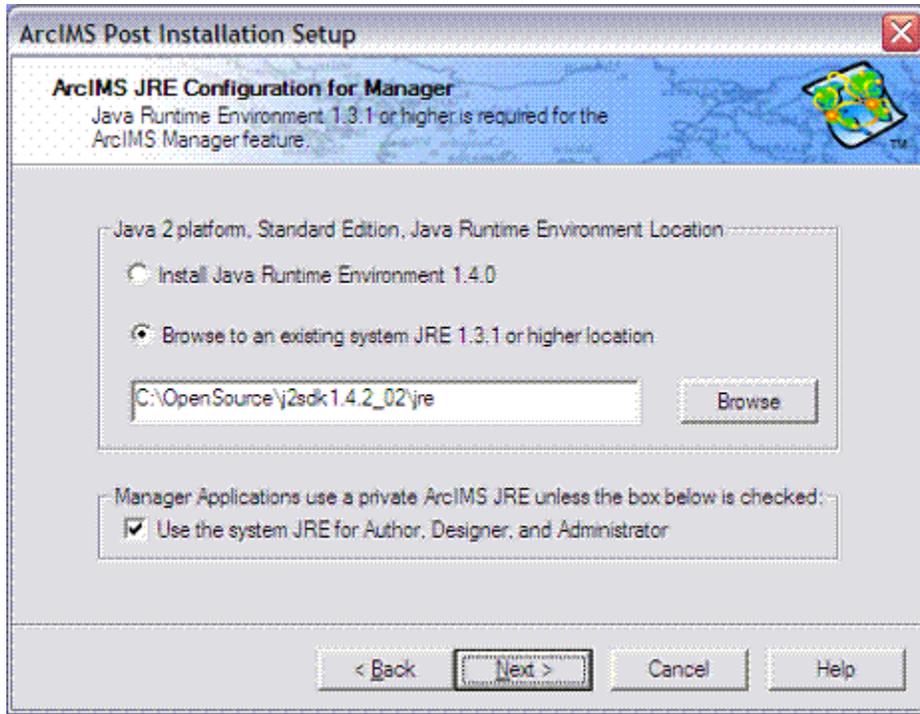


8. Set the output and working directories, then click **Next**.
9. Enter the directory where you want to store generated images and web pages. This is typically the ArcIMS installation directory. Click **Next**.
10. Set the web server host name to the name of your machine, select the **http** option, then click **Next**.



11. Set the ports and click **Next**.

12. Set the user name and password for the administrator console. Click **Next**.
13. Set the JRE path, then select **Use the system JRE for Author, Designer, and Administrator**. Click **Next**.



ArcIMS is now installed. Three services will be added to your machine and started automatically. You may want to change these services to start up manually.

#### To install and configure the connectors:

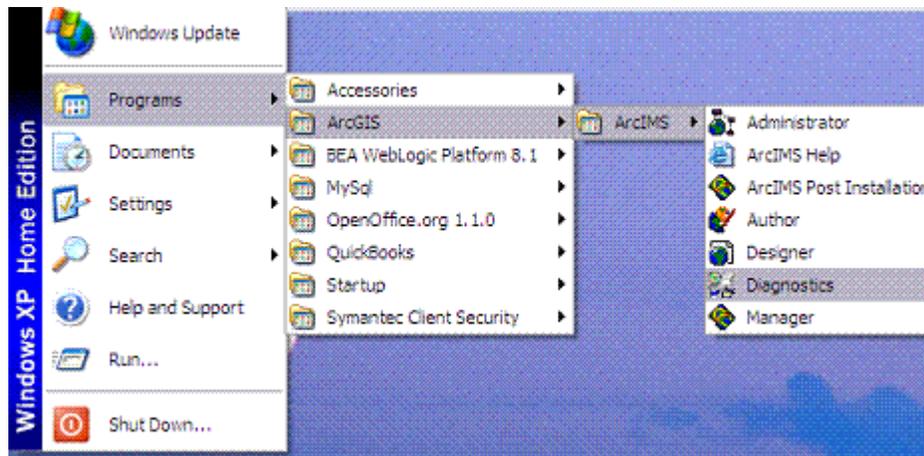
1. Create a subdirectory under `<Tomcat install_dir>\webapps` called `servlet`.
2. Copy the file `<ESRI>\ArcIMS\ArcIMS\Connectors\Servlet\arcimsservletconnector.war` to `<Tomcat install_dir>\webapps\servlet`.
3. Expand the **WAR** file in `<Tomcat install_dir>\webapps\servlet`. Open a command window and enter:
 

```
C:> jar -xvf arcimsservletconnector.war
```
4. Change to the **WEB-INF\classes** subdirectory.
5. Modify **Esrimap\_prop** and **WMSEsrimap\_prop** as follows:
  - **Esrimap\_prop**: Set the `appServerMachine` property to the name of your server from the ArcIMS installation or your IP address.
  - **WMSEsrimap\_prop**: Set the `appServerMachine` property to the name of your server from the ArcIMS installation or your IP address.
6. Set the `enable` property to `True`.

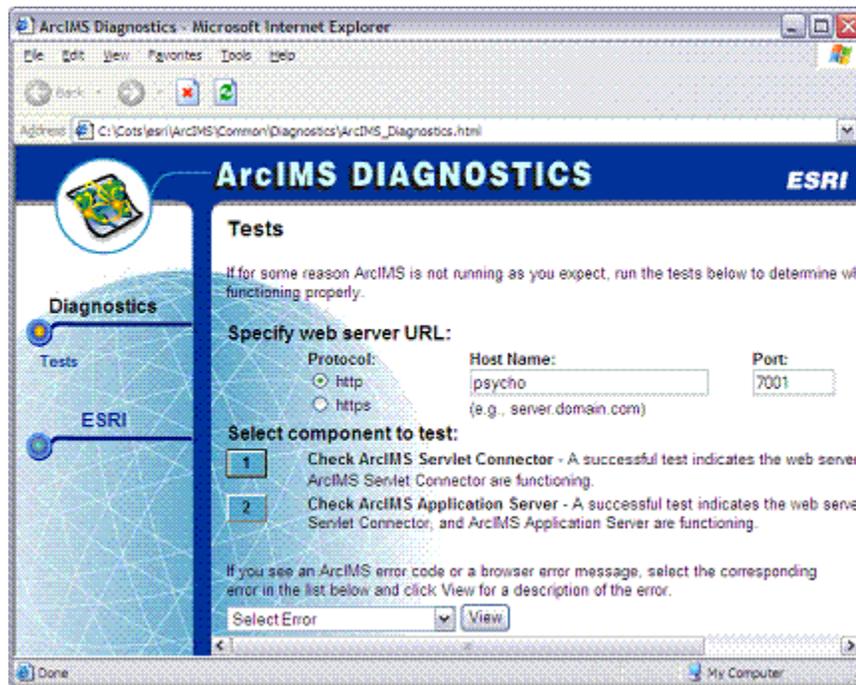
The connectors are now ready to test.

#### To test the ESRI connectors with the ESRI Diagnostics tool:

1. Start Tomcat.
2. Open the Windows **Start** menu, then select **Programs** (in Windows XP, **All Programs**) > **ArcGIS** > **ArcIMS** > **Diagnostics**.



3. In the ArcIMS Diagnostics window, set the **Protocol** to http. Enter the host name and port.
4. Run both the number 1 and the number 2 test.



This figure shows the test output:



### To configure the administrator tool:

Install the ArcIMS Administrator.

1. Create a subdirectory under `<Tomcat install_dir>\webapps` called **esriadmin**.
2. Copy the file **esriadmin.war** from `<ESRI install_dir>\ArcIMS\Administrator\` to `<Tomcat install_dir>\webapps\esriadmin`.
3. Expand the **WAR** file in `<Tomcat install_dir>\webapps\esriadmin`.
4. Open a command window and enter:

```
C:> jar -xvf esriadmin.war
```

### To create an ESRI output area:

1. Create a subdirectory under `<Tomcat install_dir>\webapps` called **esrioutput**.
2. Create a subdirectory under `<Tomcat install_dir>\webapps\esrioutput` called **WEB-INF**.
3. Create a file called **web.xml** under `<Tomcat install_dir>\webapps\esrioutput\WEB-INF`.
4. Add the following code to **web.xml**:

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

### **Example: ESRI WFS in Tomcat**

*Disclaimer:* This example uses open-source products, since **NESI** itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

This example builds on the *ESRI ArcIMS and web client in Tomcat example*. In this example both the ESRI **WFS** and the ESRI client are hosted as Tomcat applications.

This example uses ESRI WFS v7.0.0.

### **Set up WFS in Tomcat**

Follow the steps in this section to set up the example.

1. Obtain the ESRI WFS evaluation software from <http://www.esri.com/software/opensis/interopdownload.html>.
2. Click the **OGC WFS Connector for ArcIMS** link and download the distribution. Save it in a subdirectory under <ESRI install\_dir>. This example uses the subdirectory **WFS\_0\_0\_7**.
3. Unzip the distribution.
4. Create a new subdirectory under <Tomcat install\_dir>\webapps called **esriwfs**.
5. Create the subdirectories **WEB-INF**, **lib**, and **classes** under <Tomcat install\_dir>\webapps\esriwfs.
6. Populate these Tomcat subdirectories with the WFS distribution files as follows:
 

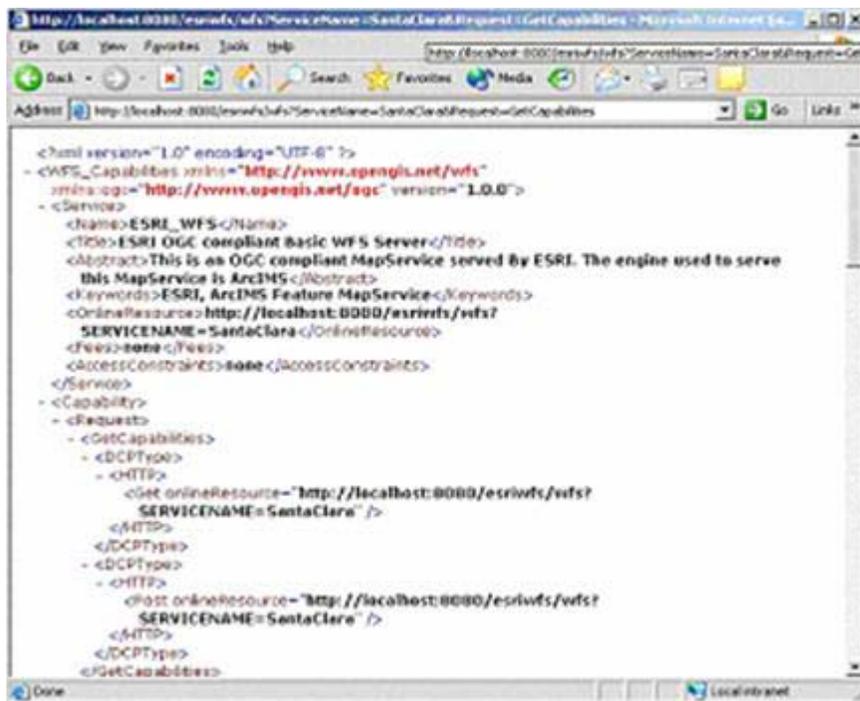
```
COPY C:\ESRI\ArcIMS\WFS_0_0_7\class\* c:\<tomcat install
directory>\webapps esriwfs\WEB-INF\classes\*
COPY C:\ESRI\ArcIMS\WFS_0_0_7\jars\* c:\<tomcat install
directory>\webapps esriwfs\WEB-INF\lib\*
COPY C:\ESRI\ArcIMS\WFS_0_0_7\config\web.xml
c:\\webapps\esriwfs\WEB-INF\*
COPY C:\ESRI\ArcIMS\WFS_0_0_7\config\ ogc_wfs.properties,
WFS_response_capabilities_0014.xsl,
WFS_response_capabilities_100.xsl c:\\webapps\esriwfs\*
```
7. Open **web.xml** in <Tomcat install\_dir>\webapps\esriwfs\WEB-INF\.
8. Add the following code between the <web-app> and </web-app> tags and save the file:
 

```
<servlet>
  <servlet-name>WFSServlet</servlet-name>
  com.esri.ogc.wfs.WFSServlet
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>WFSServlet</servlet-name>
  <url-pattern>/wfs</url-pattern>
</servlet-mapping>
```
9. Open **ogc\_wfs.properties**.
10. Set the host parameter to your machine's host name and set the servicename parameter to SantaClara.

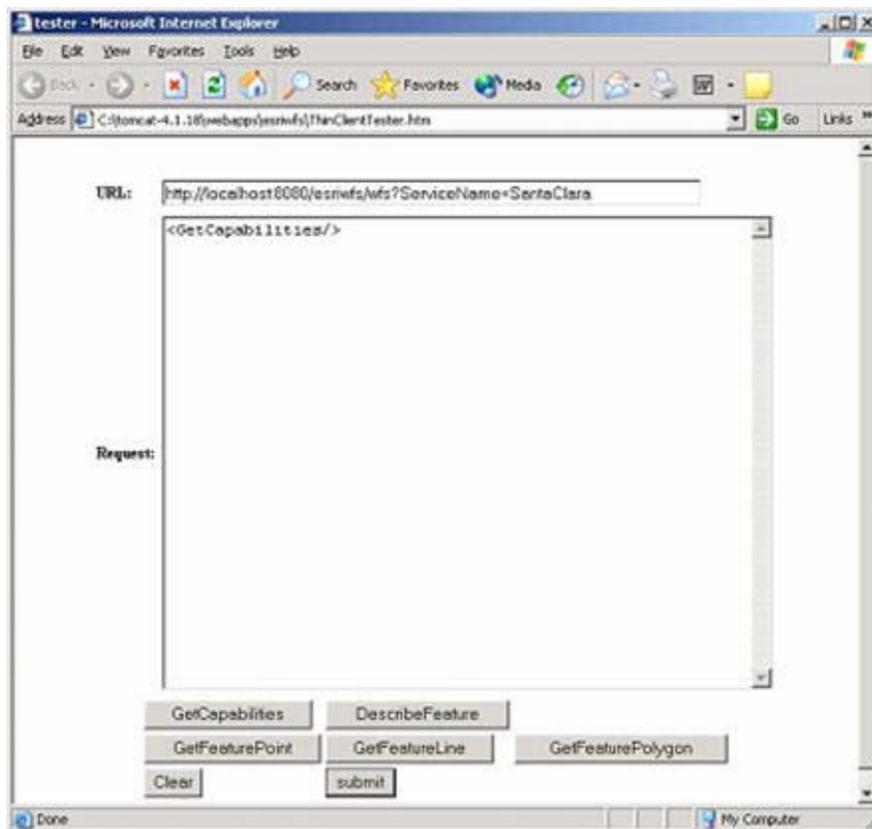
The Web Feature Server is now ready for testing.

### **Test WFS in Tomcat**

1. Start Tomcat.
2. Start the santaclara service.
3. To test the basic functionality of the WFS, point your browser at the following URL: <http://localhost:8080/esriwfs/wfs?ServiceName=SantaClara&Request=GetCapabilities>. This tests the WFS GetCapabilities API call. You should get an XML response as shown in the figure below:



4. Test other WFS APIs by using the **ESRI ThinClientTester.htm** tool located in `<Tomcat install_dir>\webapps\esriwfs`. Sample output appears in the figure below:



### **Run the ESRI WFS client to the ESRI the WFS in Tomcat**

This example builds on the *previous example* for both the ESRI WFS. It uses:

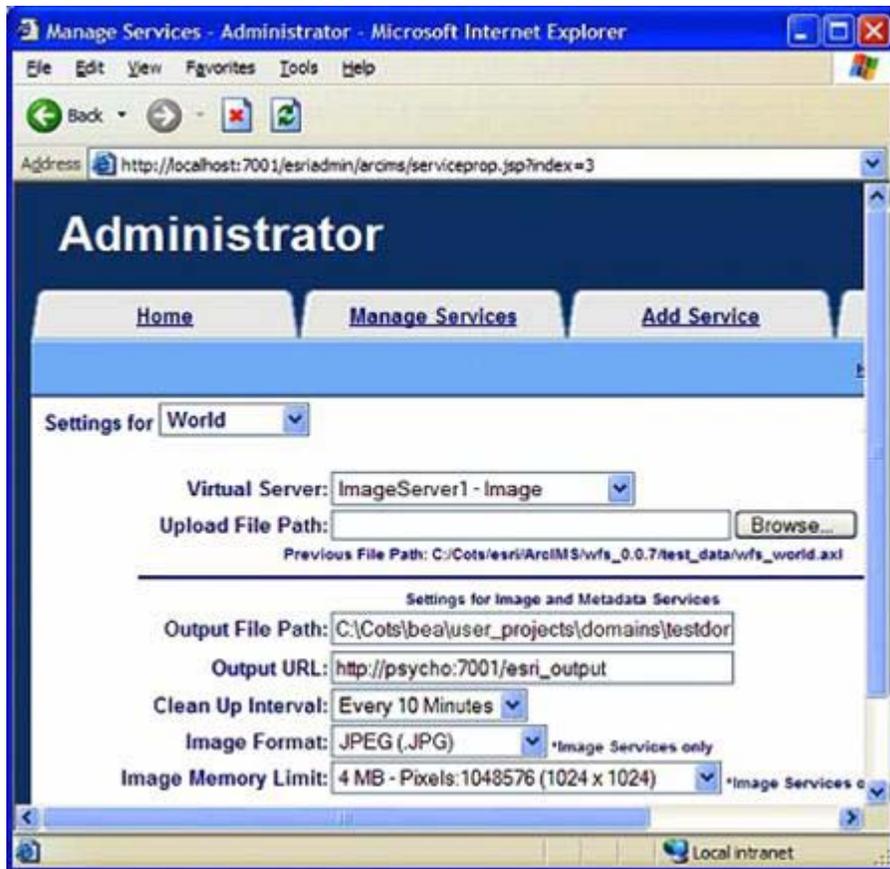
- wfs\_world sample data that comes with the ESRI WFS release
- ArcExplorer\_4.0.1, Java edition
- the interoperability extension

#### **To set up the ESRI WFS:**

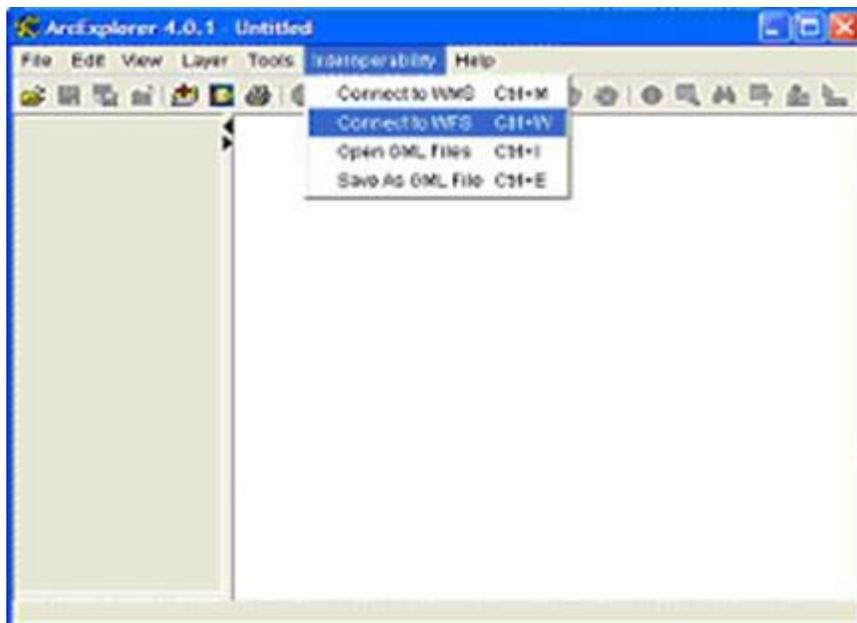
1. Configure the directory path of the <SHAPEWORKSPACE tag in the file <ESRI install\_dir>\ArcIMS\WFS\_0\_0\_7\test\_data\wfs\_world.axl to point to the directory where this file resides. For this example, it would be:

```
<SHAPEWORKSPACE name="shp_ws-0"  
  directory="C:\ESRI\ArcIMS\WFS_0_0_7\test_data" />
```

2. Go to <http://www.esri.com/software/opengis/interopdownload.html>. Obtain the ESRI WFS Client called arcExplorer and the interoperability extension that goes with it. Put them in your ESRI install directory.
3. Install arcExplorer and the interoperability extension in the ESRI install directory.  
**Note:** The ESRI WFS client is a standalone JAVA SWING thick client and the interoperability extension is a plug-in to arcExplorer for communicating with a WFS.
4. Start Tomcat.
5. From the ESRI administrator tool, create a new service and point the Upload File Path parameter to the directory containing the **wfs\_world.axl** file.



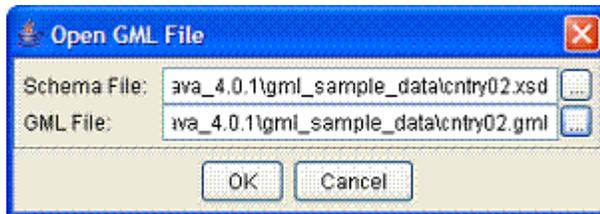
6. Open the Windows **Start** menu and select **ArcGIS > ArcExplorer**.
7. Open the **Interoperability** menu and select **Connect to WFS**.



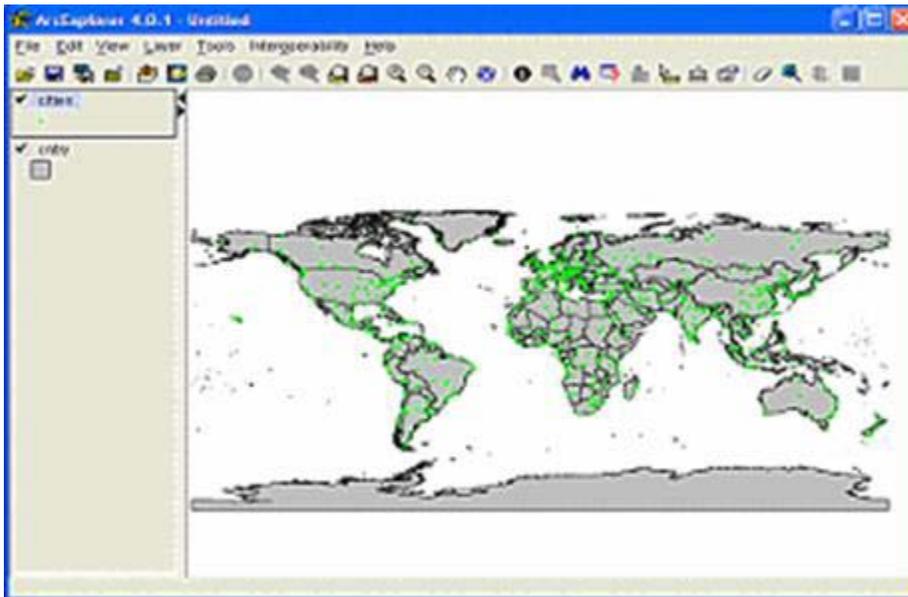
- Set the connection to the ESRI Web Feature Server by specifying the **URL** and **Service Name** of the WFS.



- Open the **Interoperability** menu and select **Open GML Files** to load in a background map. You can find the **entry02.gml** and **entry02.xsd** files in the directory where you installed the interoperability extension.



The output appears in the figure below:



**Example: ESRI WFS to NOSWC WFS client**

*Disclaimer:* This example uses open-source products, since **NESI** itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

This example builds on the previous examples of the **ESRI WFS** server and the **NOSWC** configuration. You will configure the NOSWC to point to the ESRI **WFS** by modifying **baselayers.xml** and **web.xml**, which pointed to the NOSWC WFS in *Example: WFS to NOSWC in JBOSS*.

**Set up the example**

1. Shut down the Tomcat web server.
2. Configure **web.xml** in `<Tomcat install_dir>\webapps\WebCOP\WEB-INF` to point to the ESRI WFS as shown here:

```
<param-name>WFS_PATH</param-name>
<param-
value>http://198.253.7.109:7001/esriwfs/wfs?ServiceName=World
</param-value>
```

To connect to this WFS, you must specify the `ServiceName=World` parameter.

3. Configure the **baselayers.xml** file in `<Tomcat install_dir>\webapps\WebCOP\WEB-INF` to point to the ESRI WFS as shown here:

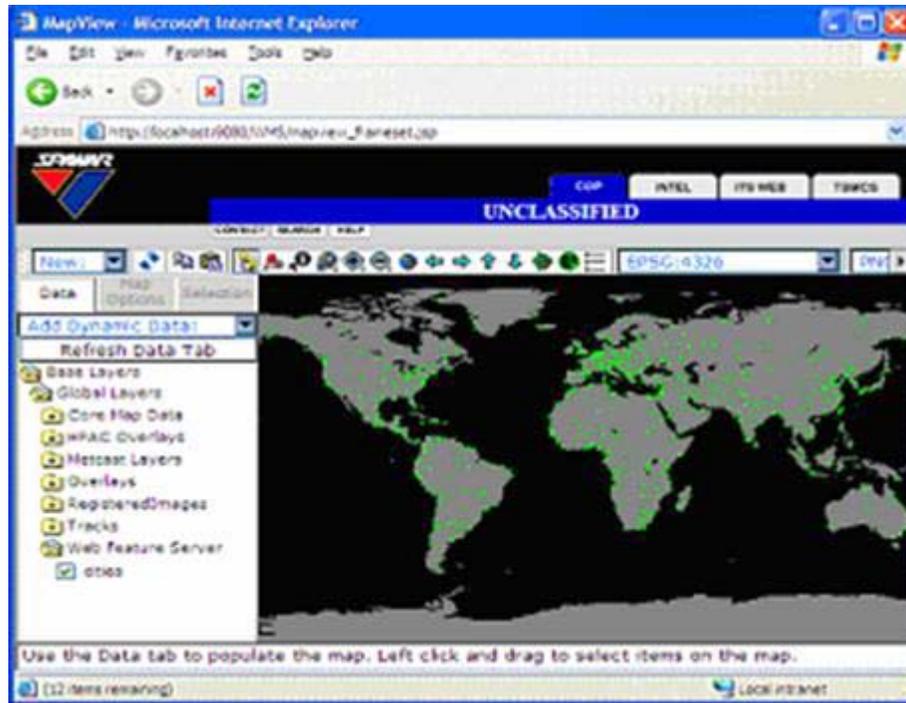
```
<Port>7001</Port>
<Path>/esriwfs/wfs?ServiceName=World</Path>
```

4. Start Tomcat and point the browser to the NOSWC URL:

`http://localhost:8080/WMS.`

**Note:** The port number is the number you used when you installed NOSWC.

5. In the left menu, open the Web Features Server folder and select **Cities**. The cities appear as dots on the NOSWC as shown in the figure below:

**Command and Control Personal Computer (C2PC)**

C2PC is a Windows-based thick client application for viewing and manipulating command-and-control (**C2**) data. ATLAS is the GIS layer of the application. You can obtain ATLAS from the C2PC program office, Ground C2 Program Manager for the Marine Corps at MARCORSSYSCOM. This section identifies infrastructure interfaces, **APIs**, and specifications for applications sharing the enterprise network

**Recommendation**

To ensure decoupling from the visualization layer, do not develop to the ATLAS APIs. Develop to either the *OGC* open-standards APIs (*GO-1* and Geobjects) or to the JMTK COE APIs. C2PC bindings allow developers to use either strategy.

**Interface layers**

Interface layer	Description
<i>C2PC ICSF Interface Layer (CIIL)</i>	This is a combined Navy and FIOP sponsored effort that is under development. It provides a <i>JMTK</i> interface layer that allows applications developed for the COE to run on C2PC. Segments must be repackaged for the non-kernel C2PC environment.
<i>C2PC XIS Interface Layer (XIL)</i>	This is a combined Army and FIOP sponsored effort that is under development. No sample code is available. It provides an OGC -compliant interface layer that runs on both C2PC and COE 4.x.  The GIS layer of the segments must be recoded to run on C2PC, and segments must be repackaged for the non-kernel C2PC environment.

**Example: C2PC to NOSWC WFS**

*Disclaimer:* This example uses open-source products, since *NESI* itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

This example shows how to use *C2PC* with the *Web Feature Server* architecture. The setup in this section is specific to C2PC, but the JBoss and MySQL configurations apply. This example requires an *OGC* bridge. Installing the XIS Integration Layer (*XIL*) creates the bridge to C2PC. This example was done on an alpha release of XIL.

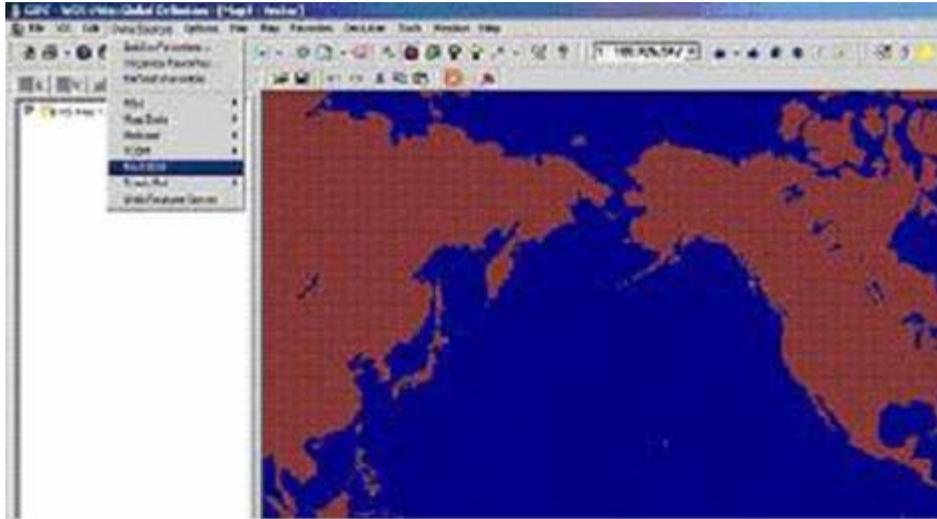
**To set up the example:**

1. Install C2PC.
2. Install XIL.
3. Create a directory under **C:\Program Files\USMC\xil\** called **xisc2pc**.
4. Extract **xisc2pc.jar** into the **xisc2pc** directory.
5. Change to the **xisc2pc\com\xis\wfsdsi\leifResources** directory.
6. Edit **wfs.xisref** as shown below to set the port number to that of the WFS server:

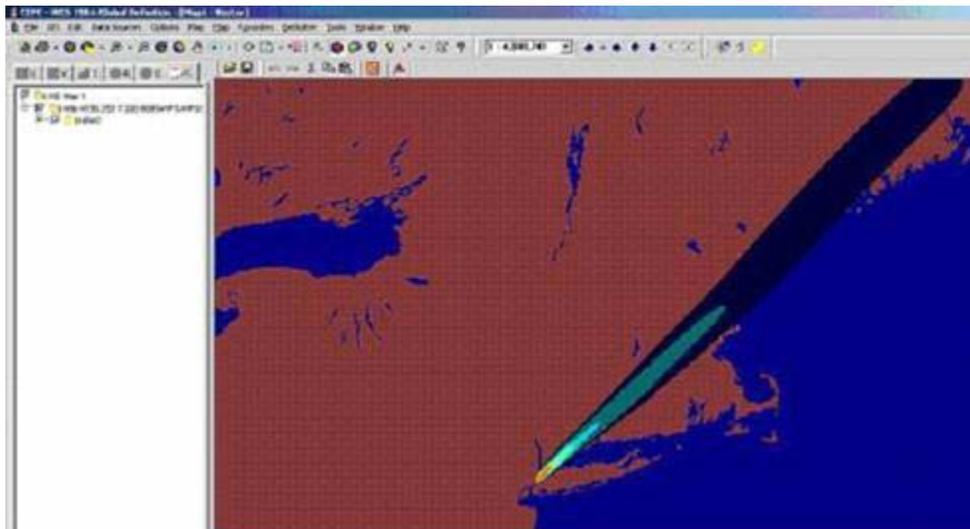
```
doscript=\
.. wfs = new Packages.com.xis.wfsdsi.WFSDSI \
  ( "localhost", 8080, "/WFS/WFSServlet");\
..if (view == null) {\
.. ../ view = new
```



7. If the DSI menu items do not appear, select **Options > Open Console**. The Console window displays any Java exceptions thrown by XIL. If there are no exceptions, double check that all of the above instructions were followed.



C2PC connects to the WFS and displays the plume on the C2PC map:



### **Example: C2PC to C/JMTK WFS**

*Disclaimer:* This example uses open-source products, since **NESI** itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

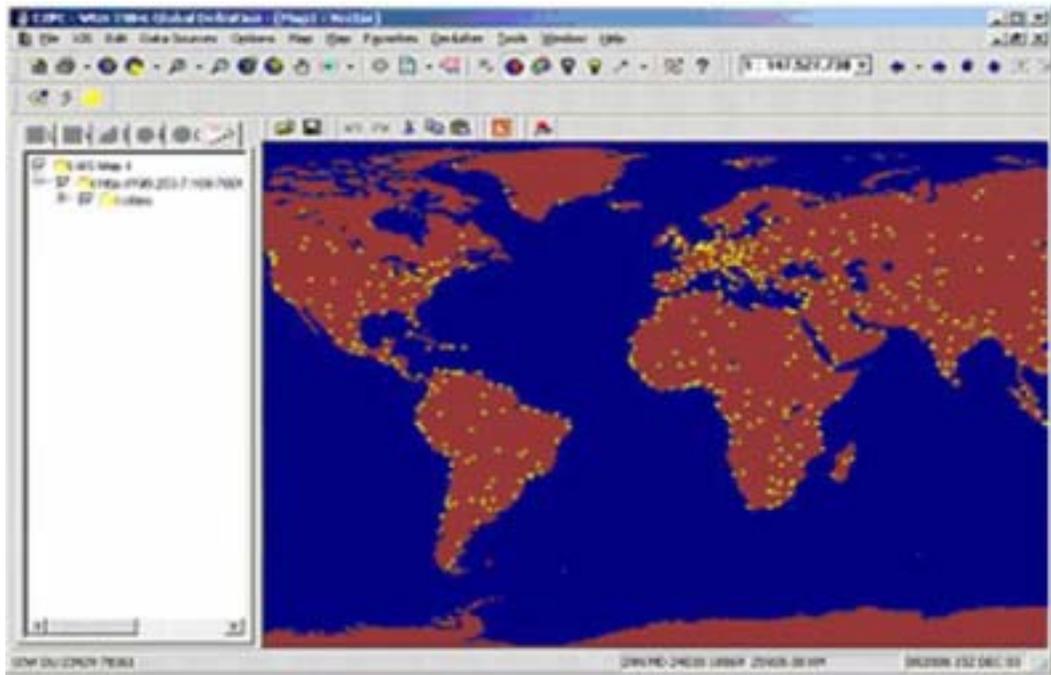
**C2PC** can connect to the ESRI Web Feature Server via the **XIL** layer just as it connected to the **NOSWC** WFS. The WFS client connectors come with the XIL release. You can obtain the code for this from the NOSWC site (for more information, send email to [info@polexis.com](mailto:info@polexis.com)).

**To set up this example:**

- Configure C2PC to accept WFS input. The ESRI WFS configurations used in the *C2PC to NOSWC WFS example* also apply to this example.

#### To run the example:

1. Start the Tomcat server.
2. Start the world service from the ESRI administrator console.
3. Open the Windows **Start** menu and select **Programs** (in Windows XP, **All Programs**) > **C2PC** > **C2PC Client** to launch the C2PC client application.
4. Follow these steps to start up XIL from within C2PC:
  - a. Open the **Tools** menu and select **XIS Injector**. This reconfigures the menu items in C2PC.
  - b. Open the **Data Sources** menu and select **Web Feature Server**.
  - c. Select **Options** > **Open Console** to display any errors during the connection to the WFS.
5. The cities are rendered on the world map (similar to the ESRI ArcExplorer display) as shown in the figure below:



#### Joint WebCOP (JWC)

The **JWC** is a new **GCCS** FoS initiative under development. Based on GCCS FoS ERA, it combines separate GIS efforts into a joint services-based platform that is open and extensible. The JWC combines the C/JMTK ArcIMS product suite, the NOSWC, and **DISA's** WebCOP.

The JWC project is a multi-team, open-source effort between **DISA**, Army, Navy, and Air Force. The first engineering drop was delivered in Q2CY04 and it is expected to be complete by Q4CY05. JWC will be incorporated into the GCCS-J 4.2 baseline in Q1 Fy06.

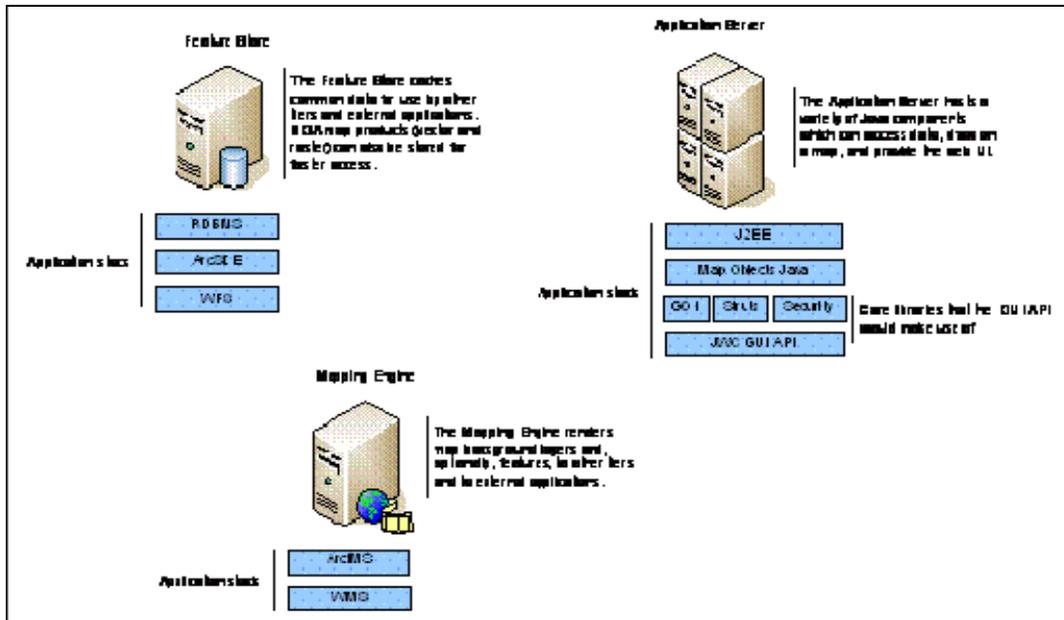
## Standards

The JWC adheres to the following standards:

Standard	Web site
<b>SOAP</b> 1.1	<a href="http://www.w3.org/TR/SOAP">http://www.w3.org/TR/SOAP</a>
<b>WSDL</b> 1.1	<a href="http://www.w3.org/TR/2001/NOTE-wsdl-20010315">http://www.w3.org/TR/2001/NOTE-wsdl-20010315</a>
GML 3.0	<a href="http://www.opengis.org/techno/documents/02-023r4.pdf">http://www.opengis.org/techno/documents/02-023r4.pdf</a>
<b>J2EE</b> 1.3.1	<a href="http://java.sun.com/j2ee/sdk_1.3">http://java.sun.com/j2ee/sdk_1.3</a>
<b>HTML</b> 4.0	<a href="http://www.w3.org/TR/1998/REC-html40-19980424">http://www.w3.org/TR/1998/REC-html40-19980424</a>
<b>CSS</b> 2.0	<a href="http://www.w3.org/TR/REC-CSS2">http://www.w3.org/TR/REC-CSS2</a>
<b>JavaScript</b> 1.5	<a href="http://www.ecma-international.org/publications/standards/ECMA-262.htm">http://www.ecma-international.org/publications/standards/ECMA-262.htm</a>

## Architecture

The **JWC** is a modular, loosely coupled, web-enabled, distributed, N-tier, service-oriented architecture written in Java, **JavaScript**, and HTML. It is platform-independent and designed to work in a heterogeneous environment.



## Tiers

The N-tiered architecture consists of:

- **Web-service**-based data tier

- **J2EE**/CJMTK middle tier
- Thin-client-visualization tier

### Services and components

The architecture comprises these major services and components:

- *Feature store service*
- *Mapping service*
- *Application service*
- *Mediator service*
- *Security*
- *Client*

### Feature store service

The feature store service is a web service that provides high-speed, scalable access to common data. Data communications occur via the **WFS** protocol, an open standard published by the **OGC**. This provides a layer of abstraction between data services and the mapping application, offering an easy migration path for existing data feeds. The WFS uses a relational database to cache data locally as geospatial features.

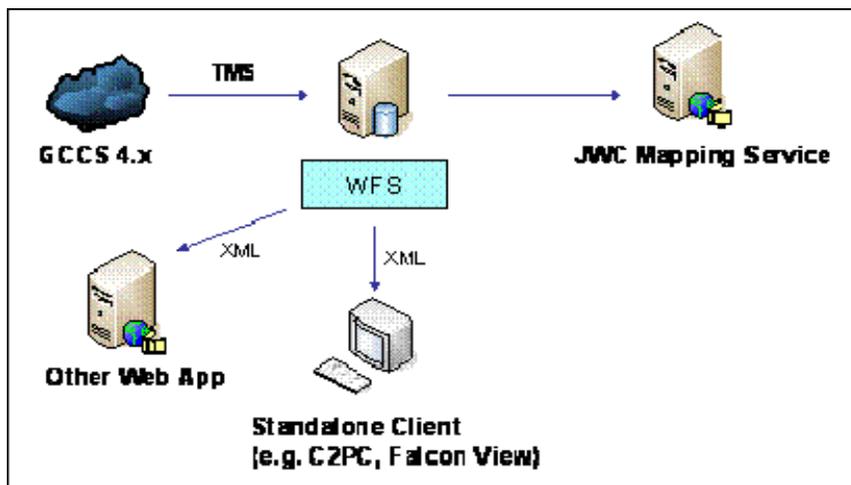
The feature store is completely modular:

- Not tied to any other **JWC** service
- Based on the principle of loose **coupling**
- Independent of a given **RDBMS** implementation

The planned data services for Phase 1 are:

- COP Track Service
- TBMCS Air Battle Information Services
- I3 Data Services

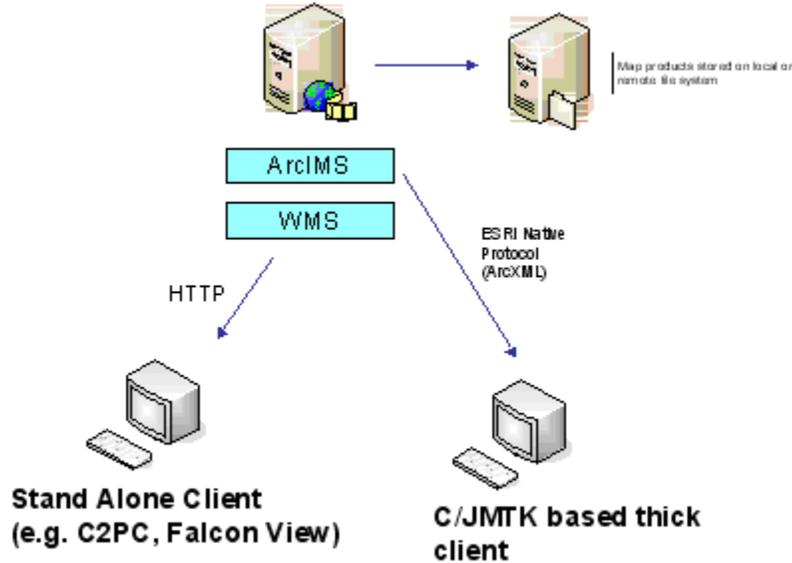
This figure illustrates the simplified **collaboration** model for the WFS in the JWC:



### Mapping service

The mapping service is based on ESRI's ArcIMS. It supports all NGA mapping products and commercial satellite formats. It is accessible via an open-standard map protocol. Any client can retrieve multi-layered imagery from this service using a simple HTTP call. The mapping service renders the full MIL-2525B symbology set, plus custom sets such as NTDS.

This figure illustrates the simplified collaboration model for the **WMS** in the **JWC**:



### Application service

The **JWC** application service is built on a WebLogic 8.1 **J2EE** application server. It contains **business logic** modules and user interface management modules, and can coexist with other J2EE apps such as WEEMC.

The application service uses a standards-based API GO-1, a map-independent API, to communicate with the *mapping service*. This offers applications the ability to use a variety of maps with the same code base, thus minimizing the porting costs between maps.

### Mediator service

The mediator service promotes the loose coupling between the **JWC** services and the ArcSDE database. It allows independent control and coordination of the interactions between the ArcSDE database and the other JWC components and data sources. It simplifies the JWC component by replacing the many-to-many interactions with one-to-many interactions. Mediating the ArcSDE database operation involves:

- Querying a data service
- Transforming the **XML**
- Applying any business rules to the data
- Storing the data in the ArcSDE database

### Security

The **JWC** security architecture is **PKI**-enabled and based on **GSALT** and **LDAP**. It collaborates with the **GCCS** security mechanisms for security management. It supports:

- PKI
- User- and role-based authentication
- The HTTP (HTTP/S) standard for all client connections
- Audit trails

### **Client**

One of the **JWC's** primary benefits is its convenient web-based interface. End users can connect to the **LAN** from any computer or even a handheld device and gain instant access to the COP. It is no longer necessary to install the **COE** on each machine, which reduces the cost of supporting users.

The **client** is built with HTML and **JavaScript** only, and does not require any plugins. It will run on Internet Explorer 5.5, Mozilla 1.3, and subsequent versions of either.

Since the client is web-based:

- It does not require a client-side installation
- All client profile information is managed on the server
- Any authorized user on the network can access the client

### **Infrastructure components**

The **JWC** uses the following industry-standard, **loosely coupled** components:

- Data plugins
- Web mapping engine
- Symbology and overlay rendering
- Geospatial database
- Application server
- TMS web service wrappers

### **Recommendations**

1. Developers developing to the JWC should develop to the **OGC** standards. Future releases of this will include sample code for the JWC.
2. Functional service providers should follow these high-level recommendations in their code:
  - Expose interfaces via **WSDL**. JWC can support either **REST** or **SOAP**.
  - Expose all data in **XML**.
  - Publish a schema file for your data.
  - Use DoD XML registry tags where appropriate.
  - Use **HTTPS** to communicate with the JWC.

### **Best practices**

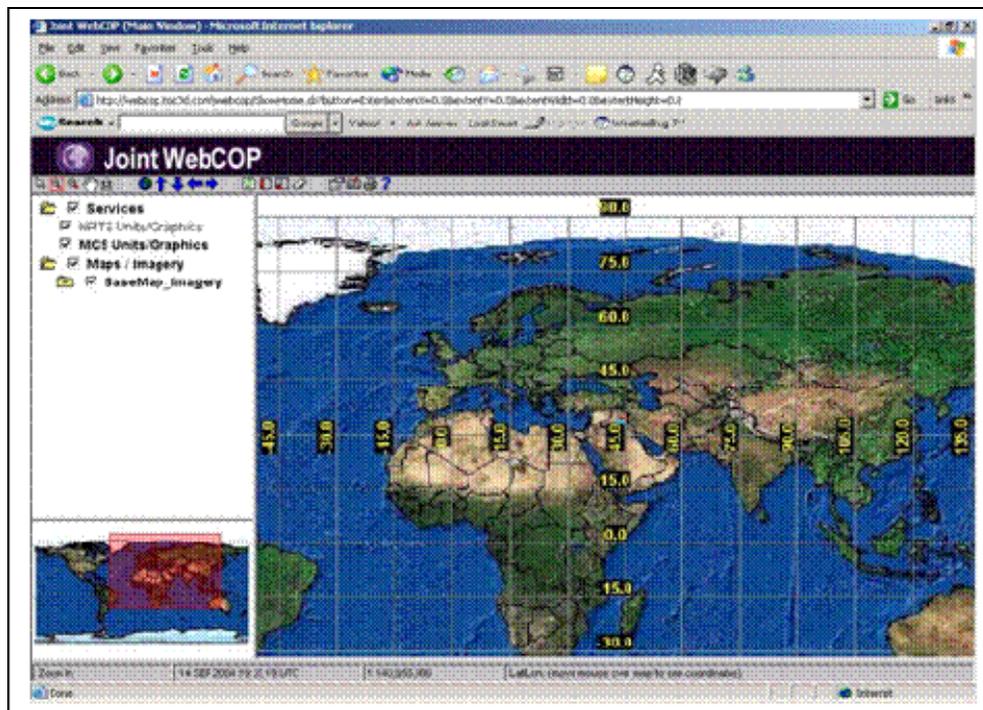
To join the **JWC** open-source development community:

1. Go to the **DISA** open-source site at <https://www.geden.org/>.
2. Register for an account and log in.
3. Request access to the Joint WebCOP project.
4. Download the document for the development process overview and submission guidelines from <https://jointwebcop.geden.org/servlets/ProjectDocumentList?expandFolder=687&folderID=687>.

For more information, go to the Joint WebCOP project on the DISA open-source site and click on one of the project owners at <https://jointwebcop.geden.org/>.

### References

1. To run the current build of the Joint WebCOP, go to <http://webcop.toc3d.com/jwebcop/index.jsp>.
2. To get to the symbology, click one of the colored dots on the map and drill down.
3. Set the Overview Map so that you can see the details and overviews in the multiple panes.



## Implementing GIS open architecture

Currently, developers connect to NOSWC via **DSIs**, which connect data sources to the NOSWC framework, or **GO-1 APIs**. Developers can use existing DSIs, use GO-1 APIs, or write custom DSI components and integrate them into the framework. The framework graphically displays the data on a map.

## Strategy

Develop and use an OGC-compliant abstraction layer that operates with existing GIS applications like COE 4.x, C2PC, NOSWC, WebCOP, *JWC* and C/JMTK. The *mission* applications program to this OGC-compliant abstraction layer for rendering. The abstraction layer manages all rendering features.

The strategies outlined in this section offer the lowest cost strategies based on currently available implementations and release schedules.

## GIS development communities

To support an open-standards, component-based approach, *NESI* has created two development communities for GIS applications:

- Rendering control community
- Data exposing community

### *Rendering control community*

This community produces software that visually represents features on a GIS canvas.

Some examples of user communities in this area are 4.x Symplot Plugins, GCCS-M mission apps that plot to charts, and Draw modules and plotters.

### *Data exposing community*

This community produces software that supplies data to various communities, such as the rendering control community. Developers in this community, such as COE 4.x data producers, expose data for rendering.

There are two data exposing subcommunities, based on the type of data exposed to GIS applications:

- Gridded data producers
- Point or feature data producers

## GIS data categories

To help developers choose between *WFS* and *WCS*, *NESI* identified three broad *categories of data* based on complexity.

## Target architectures

*Thin client architecture*

*Thick client architecture*

## Example: Using GO-1/Geobjects APIs in NOSWC

*Disclaimer:* This example uses open-source products, since *NESI* itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

The NOSWC uses connectors called Data Source Interfaces (DSIs). The DSIs use the *Geobject* abstraction layer to render data into an image before passing the image to the browser. A *DSI* consists of two Java classes: a DSI and a DSI translator.

Both **GO-1** and Geobject make it easy to insert other rendering capabilities (ESRI's ArcGIS) into the NOSWC without impacting the DSI business logic. You also use this programming model to program the **WFS** DSIs.

This example shows how a developer in the *rendering community* would code to a Geobjects **API** using the NOSWC. The first section describes the overall process, and the following sections explain how to code individual sections.

## Process

### To build a connection to the NOSWC:

1. Write the DSI (*TestDsi.java*).
2. Write the translator class (*TestDsiTranslator.java*).
3. Place the compiled classes into a **JAR file**.
4. Copy the **jar** file into the **lib** directory of the NOSWC installation.  
For example, if you are using a Tomcat web server, the **lib** directory is located in **<drive letter>:\<Tomcat install\_dir>\webapps\WebCOP\WEB-INF\lib**.
5. *Register* this new DSI in the **baselayers.xml** file so the DSI appears in the left navigation tree of the NOSWC GUI interface.  
For example, if you are using a Tomcat web server, this file is located in **<drive letter>:\<Tomcat install\_dir>\webapps\WebCOP\WEB-INF**.
6. Restart the server.
7. *Review* your custom DSI as it appears in the NOSWC environment, where it can be manipulated.

### Write TestDsi.java

```
package dsi;

import java.awt.Color;

public class TestDsi
{
    private String type;
    private double latRadians;
    private double lonRadians;
    private String weatherData;
    private Color color;

    public TestDsi(){}

    // type - name
    // lat, lon - latitude, longitude in degrees
    public TestDsi(String type, double lat, double lon)
    {
        this.type = type;
        setColor(new Color(0,255,255));
        latRadians = Math.toRadians(lat);
        lonRadians = Math.toRadians(lon);
    }
}
```

```
public void setType(String tp)
{
    type = tp;
}

public void setLat(double lat)
{
    latRadians = lat;
}

public void setLon(double lon)
{
    lonRadians = lon;
}

public void setWeatherData(String wx)
{
    weatherData = wx;
}

public void setColor(Color clr)
{
    color = clr;
}

public String getType()
{
    return type;
}

public double getLat()
{
    return latRadians;
}

    public double getLon()
    {
        return lonRadians;
    }

public String getWeather()
{
    // the developer might make a live call to a data source here like
    // a webservice or a database this method is one of several
    // that are called when a user right-clicks on
    // this dsi on the map and then selects the properties item
    // from the menu
    return weatherData;
}

public Color getColor()
{
    return color;
}

public String toString()
{
```

```

    return type;
}
}

```

### **Write TestDsiTranslator.java**

```

package dsi;

import com.xis.leif.im.Domain;
import com.xis.leif.im.Translator;
import com.xis.leif.im.BaseDataItem;
import com.xis.leif.im.AttributeGetRequest;
import com.xis.leif.im.AttributeDescriptor;
import com.xis.leif.im.AttributeDescriptorFactory;
import com.xis.leif.im.FieldMetaData;
import com.xis.domains.leif.LeifDomain;
import com.xis.domains.map.MapDomain;
import com.xis.domains.geo.GeoDomain;
import com.xis.domains.display.DisplayDomain;
import com.xis.domains.temporal.TemporalDomain;
import com.xis.im.types.StringTypeMetaData;
import org.geobject.coord.LatLonAlt;
import org.geobject.GeobjectDefault;
import java.awt.Color;

public class TestDsiTranslator extends Translator
{
    private static final Domain[] baseDomains =
        new Domain[] {
            DisplayDomain.getDomain(), MapDomain.getDomain(),
            TemporalDomain.getDomain(), LeifDomain.getDomain(),
            GeoDomain.getDomain(),
        };

    private static FieldMetaData[] fieldMetaDataArray;
    private static AttributeDescriptor[] localAttributeDescriptors;
    private static AttributeDescriptor type;
    private static AttributeDescriptor weather;

    public TestDsiTranslator()
    {
        if (localAttributeDescriptors == null)
        {
            AttributeDescriptorFactory factory =
                AttributeDescriptorFactory.getAttributeDescriptorFactory();

            type
                = factory.createAttributeDescriptor
                    ( "type",
                      TestDsiTranslator.class,
                      new StringTypeMetaData("Type")
                    );

            weather
                = factory.createAttributeDescriptor
                    ( "weather",
                      TestDsiTranslator.class,

```

```

        new StringTypeMetaData("Weather")
    );

    localAttributeDescriptors = new AttributeDescriptor[] {type, weather};
}

if( fieldMetaDataArray == null )
{
    FieldMetaData latLonAlt = new FieldMetaData(GeoDomain.latLonAlt );
    FieldMetaData typeMetaData = new FieldMetaData(type, "Type");
    FieldMetaData weatherMetaData = new FieldMetaData(weather,
"Weather");
    FieldMetaData penColor = new FieldMetaData(DisplayDomain.penColor);
    fieldMetaDataArray
    = new FieldMetaData[]
    { latLonAlt,
      typeMetaData,
      weatherMetaData ,
      penColor
    };
}
}

public Domain[] getBaseDomains()
{
    return baseDomains;
}

public FieldMetaData[] getFieldMetaDataArray()
{
    return fieldMetaDataArray;
}

public AttributeDescriptor[] getAttributeDescriptors()
{
    return localAttributeDescriptors;
}

public LatLonAlt getLatLonAlt(AttributeGetRequest agr)
{
    LatLonAlt position
    = (LatLonAlt)GeobjectDefault.createCoordinate(LatLonAlt.class);
    double lat = ((TestDsi)agr.getRawDataItem()).getLat();
    double lon = ((TestDsi)agr.getRawDataItem()).getLon();
    position.setLatLon(lat,lon);
    return position;
}

public String getType(AttributeGetRequest agr)
{
    return ((TestDsi)agr.getRawDataItem()).getType();
}

public String getWeather(AttributeGetRequest agr)
{
    return ((TestDsi)agr.getRawDataItem()).getWeather();
}
}

```

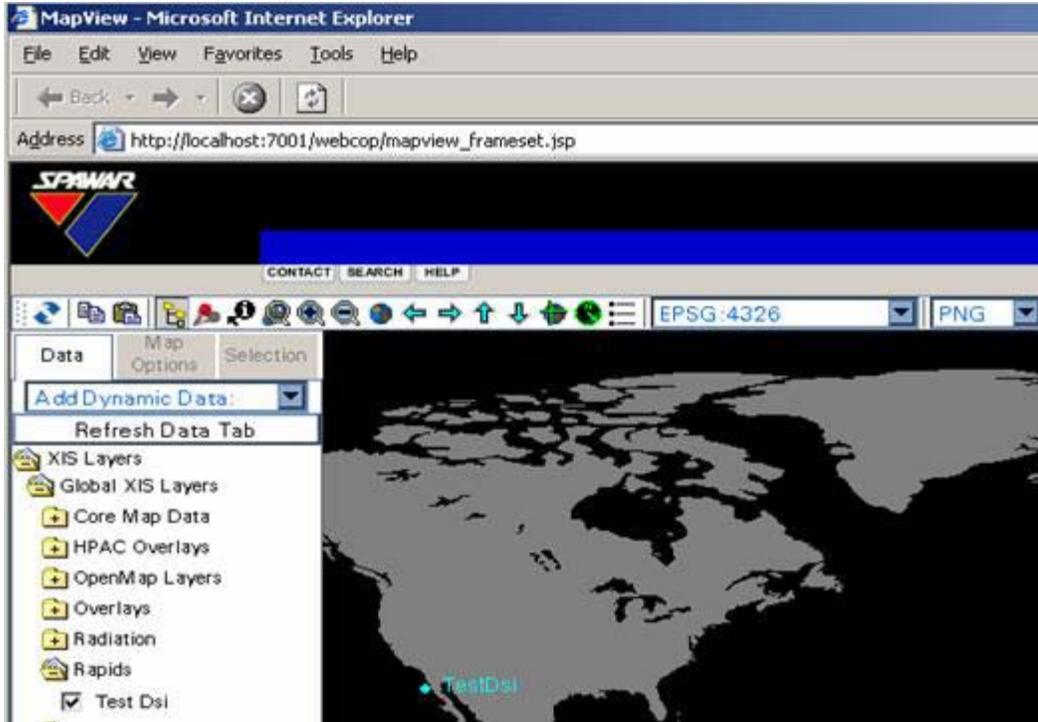
```
public Color getPenColor(AttributeGetRequest agr)
{
    return ((TestDsi)agr.getRawDataItem()).getColor();
}
}
```

### ***Register the DSI in baselayers.xml***

```
<Layer queryable="0">
  <Title>NESI</Title>
  <Abstract>NESI</Abstract>
  <Layer queryable="0">
    <Name>testdsi</Name>
    <Title>Test Dsi</Title>
    <Abstract>Test Dsi</Abstract>
    <SRS>
      EPSG:4326 AUTO:42400 AUTO:42402 AUTO:42403
      AUTO:42404 AUTO:42405 AUTO:42406 AUTO:42407 AUTO:42408
    </SRS>
    <LatLonBoundingBox minx="-180.0" miny="-90.0" maxx="180.0"
maxy="90.0"/>
    <Style>
      <Name>default</Name>
      <Title>Default</Title>
    </Style>
    <ScaleHint min="0.0" max="0.0"/>
    <DataItemJavaScript>
      newPackages.dsi.TestDsi("TestDsi", 30.71, -117.12);
    </DataItemJavaScript>
  </Layer>
</Layer>
```

### ***Review the test output***

The WebCOP output below shows how the test DSI should appear. Notice the entry in the left navigation tree and the dot on the map.



### ***Adding icons to the map***

To place a custom icon (shown in this *example* as a red X) on the map, add the following code to the sample files.

#### **To place a custom icon:**

1. Add the following line of code to **baselayers.xml** as the first line in the `<DataItemJavaScript>` tag. The path to the icon must be a absolute path. Escape the slashes.

```
java.lang.System.setProperty("testicon", "<path-to-your-
icon>//testicon.gif");
```

This snippet of code loads the directory path and name of your icon into the system properties, so the DSI can locate the icon during runtime.

2. Update **TestDsi.java** with this code:

```
// add these lines under existing imports
import javax.swing.ImageIcon;
import com.xis.icon.IconShape;
import com.xis.icon.SimpleIconShape;

// define new variable
private SimpleIconShape icon;

// place this line inside the constructor
icon = new SimpleIconShape(new
ImageIcon(System.getProperty("testicon")));

// add this new method
public IconShape getIconShape()
{
```

```

    return icon;
}

```

- Update **TestDsiTranslator.java** with the new method and the accompanying import statement shown below:

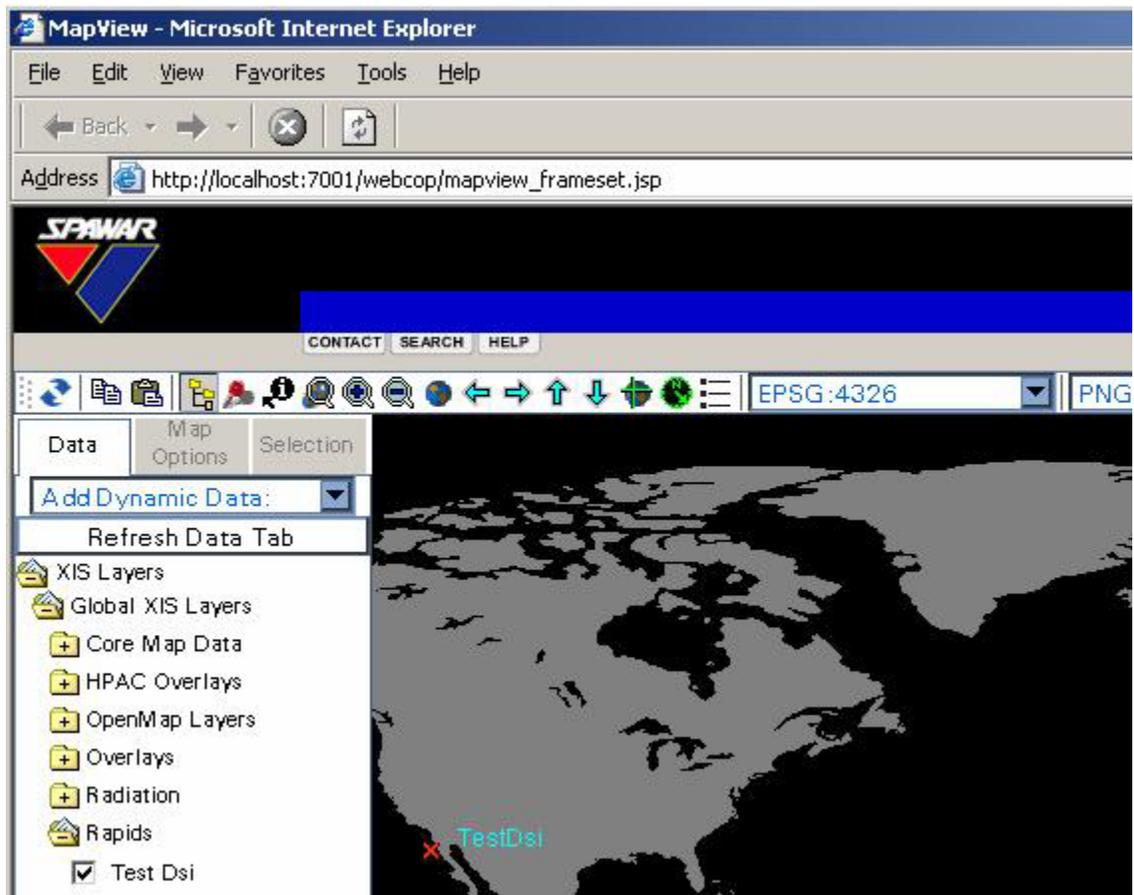
```

// add these to existing imports
import com.xis.icon.IconShape;

// add this new method
public IconShape getIconShape(AttributeGetRequest agr)
{
    return ((TestDsi)agr.getRawDataItem()).getIconShape();
}

```

The figure below shows the new icon on the map for the test DSI.



### ***Adding collections of objects to the map***

#### ***Members DSI***

You can dynamically add multiple **DSI** objects to the map using the Members DSI. This feature lets any DSI act as a **container** for other DSI objects. This technique is useful for animation and rendering data types from a data stream. To get the new objects onto the map, add them as members to an existing DSI. If a DSI contains members, the webCOP framework automatically renders them.

You can use the Members feature in many situations. Two common scenarios are:

- To hold new DSI objects that are created by a data stream
- To animate existing member DSI objects by changing the latitude and longitude attributes

### To enable the Members feature:

1. Add this code to **TestDsi.java**:

```
// add these lines under existing imports
import com.xis.leif.im.BaseInfoModel;
import com.xis.leif.im.LeifDataItem;

// define new variable
private ArrayList members;

// place this line inside the constructor
members = new ArrayList();

// add this new method
public boolean canHaveMembers()
{
    return true;
}

// add this new method
public void addMember(Object o)
{
    members.add(o);
    BaseInfoModel bim = BaseInfoModel.getBaseInfoModel();
    LeifDataItem ldi = bim.getLeifDataItem(this, false);
    try
    {
        // for some reason WebCOP throws a null pointer here if the
checkbox
        // in the browser is not selected for this DSI
        ldi.fireMemberAdded(o, false);
    }
    catch (Exception ignore){}
}

// add this new method
public Object[] getMembers()
{
    return members.toArray();
}
```

2. Add this code to **TestDsiTranslator.java**:

```
// add this new method
public boolean canHaveMembers(AttributeGetRequest agr)
{
    return
((NESIChemBioDsi)agr.getRawDataItem()).canHaveMembers();
}

// add this new method
public Object[] getMembers(AttributeGetRequest agr)
{
    return ((NESIChemBioDsi)agr.getRawDataItem()).getMembers();
}
```

```

    }

    // add this new method
    public int getInitialDrillDownLevel(AttributeGetRequest agr)
    {
        return 1;
    }

```

### **Activating the collection**

To activate the collection, you must instantiate a triggering mechanism. In this example, a new thread in **testDSI.java** listens for incoming data and adds a new DSI or modifies an existing one. The example shows a portion of the `run()` method of the DSI that adds members to itself. Latitude and Longitude are generated randomly.

```

try
{
    // read attributes from data source here...

    TestDsi dsi = new TestDsi("<myID>", 100d*Math.random(), -
100d*Math.random());

    dsi.setColor(new Color(255,255,0));
    addMember(dsi);
}
catch (Exception ex)
{
    System.out.println(ex.toString());
}

```

### **Adding overlays to the WebCOP**

There are various ways to add overlays to the WebCOP. The following example uses HPAC overlays to add plume models to the map. It creates a **DSI** that uses the Members feature. A thread triggers the framework for updates. The following example shows part of the DSI's `run()` method.

```

try
{
    String txt = "this is my incoming data stream";

    HpacOverlay hpac = HpacSetup.createOverlayFromStream(
        new ByteArrayInputStream(txt.getBytes()), "");

    HpacLayer[] layers = hpac.getLayers();

    double latRads = Math.toDegrees(
        layers[layers.length-
1].getGeoBounds().getTopLeftLatLonAlt().getLat());
    double lonRads = Math.toDegrees(
        layers[layers.length-
1].getGeoBounds().getBottomRightLatLonAlt().getLon());

    TestDsi dsi = new TestDsi("<myID>", latRads, lonRads);
    dsi.addMember(hpac);
    addMember(dsi);
}
catch (Exception ex)

```

```
{
  System.out.println(ex.toString());
}
```

### ***Dynamically updating DSI attributes***

#### **To dynamically change the attributes of a DSI:**

1. When you create the Member DSI, store a data structure of IDs within it.
2. Include an **ID** with the incoming data. Then, any subsequent data can obtain a handle to the appropriate DSI by using its ID.
3. When attributes change, signal the framework with the attribute-changed event.

## **Migrating to GIS open architecture**

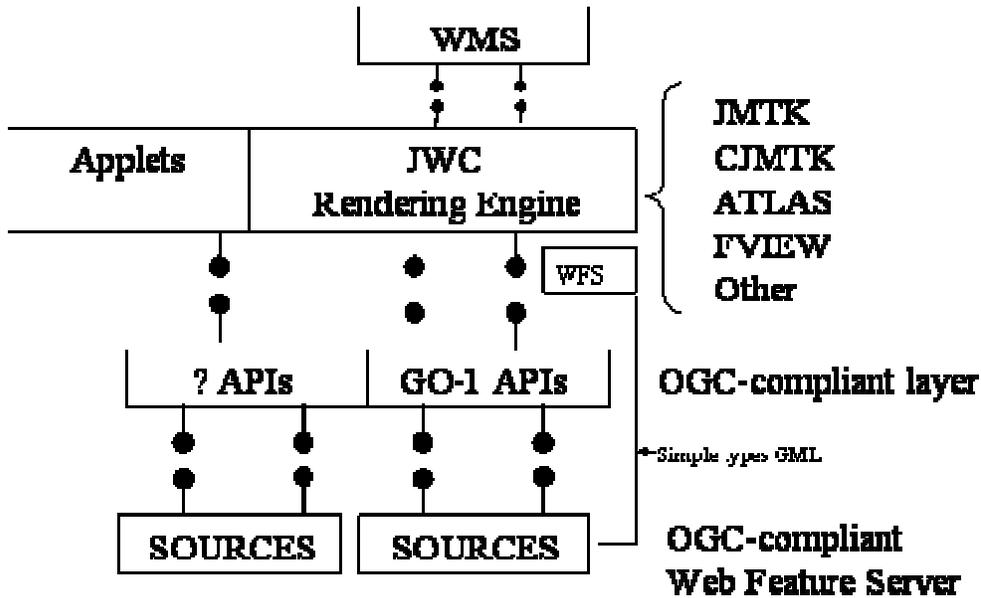
This section provides recommendations on migrating *thin*- and *thick*-client applications to an *open-standards* approach.

### **Migrating Navy thin-client applications to OGC**

#### ***Recommendations***

1. All data sources or data producers posting data to a map display such as **GCCS-M** must go through an OGC-compliant layer of open-standard interfaces for both thick and thin client maps.
2. For applications that display data on either the WebCOP visualization layer or the C/JMTK web visualization layer in a non-OGC -compliant architecture:
  - Program to the GO-1 API specification or the *WFS* model, as appropriate
  - Insulate applications from the mandated GIS conversions
  - Make sure the converted code operates on existing WebCOPs (or next mandate)
  - Restructure applications to a multi-tiered architecture
  - Migrate applications to an OGC data producer model and decouple from the chart
  - Do not render directly into the legacy APIs; program to the OGC standards
  - Change service providers to OGC-compliant Web Feature Servers or other service types in the OGC Services interoperability stack, based on content type
  - Simple data sources should be **WFSs**
  - Complex data sources should be **WCSs**

**Architecture**



**Migrating Navy thick-client applications to OGC**

The Navy is moving towards an open-standards model for thick-client GIS applications. This will be a layered *architecture* from OGC -based APIs to existing JMTK APIs. There are two initiatives towards this end:

- OGC *GO-1* API (recommended since it conforms to open standards)
- C/JMTK (not recommended; no current plans to make the implementation work on the COE)

JMTK developers should use the interim initiatives *CIIL* and *XIL* to render on COE and *C2PC* while the open-standards architectures are being completed.

**Recommendations**

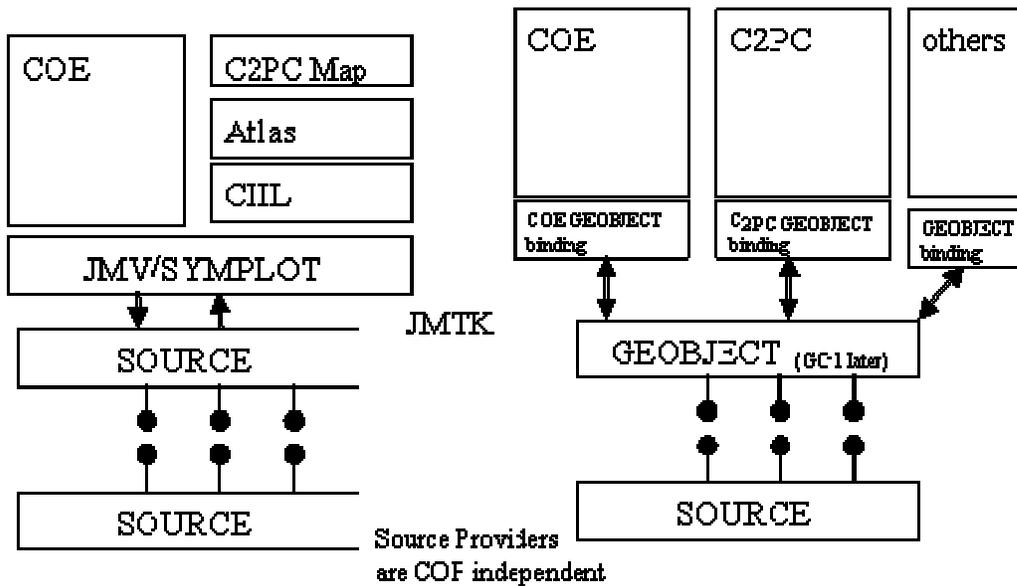
1. All data sources or data producers posting data to a map display such as *GCCS-M* must go through an OGC-compliant layer of open-standard interfaces for both thick and thin client maps.
2. For applications that display data on either the COE 4.x JMTK API visualization layer, the C2PC ATLAS API visualization layer, or the C/JMTK visualization layer:
  - Eventually, program to the GO-1 API specification
  - Insulate applications from the mandated GIS conversions
  - Make sure the converted code operates on COE 4.x, C2PC or C/JMTK (or next mandate)
  - Plan an insulation strategy till GO-1 is available (the strategy involves either a two-step or a three-step process, depending on whether there is an implemented binding of the OGC-compliant layer for the target platform)

- Restructure to a multi-tiered architecture, regardless of whether a binding of the OGC-compliant layer is available for your target platform
  - Migrate applications to 4.x data producer model and decouple from the chart, regardless of the availability of a binding
  - Decouple the rendering layer, using a design pattern like Facad or Bridge
  - Do not render directly into the legacy APIs
3. *JMV/Symplot API developers:* Convert foreground objects to Geobject APIs.
  4. *JMV/JMTK API developers:* Convert background objects to Geobject APIs.
  5. *TMS API developers:* For track producers, use an insulation strategy such as the *Façade* or *Proxy* pattern to decouple the TMS APIs from the rest of the application, positioning the application for insertion of an open-standard API currently under development.
  6. *C2PC ATLAS developers:* Use an insulation strategy such as the Façade or Proxy pattern to decouple the ATLAS APIs from the rest of the application, positioning the application for upcoming migration efforts.

In the long term, ATLAS is migrating to C/JMTK and will use the C/JMTK map. In the short term there are two initiatives:

- CIIL, an interface layer added to ATLAS that allows developers to use JMV/JMTK and SYMPLOT API calls to C2PC
- GO-1, an OGC abstraction layer added to ATLAS that allows developers to use OGC GO-1/GEOBJECTS API calls

**Architecture**



# Mobile devices

## Overview

Mobile devices encompass portable technologies such as PDAs, wireless devices, cell phones, tablets, Blackberries, and so on. Future devices are likely to become smaller, more powerful, and more portable.

DoD mobile device guidelines do not allow wireless connections in classified spaces. Handheld devices have the same operational restrictions as other equipment.

## Best practices

### To get the DoD mobile device guidelines:

1. Go to <http://www.c3i.osd.mil/>.
2. Search on wireless security policy.
3. On the search results page, select: **Pentagon Area Common Information Technology (IT) Wireless Security Policy**.

There are two environments in the mobile device arena:

- *PalmOS*
- *Wireless devices* (such as cell phones)

DoD wireless strategies are undergoing development. Check back for additional information in future releases.

## Wireless cell phone environments

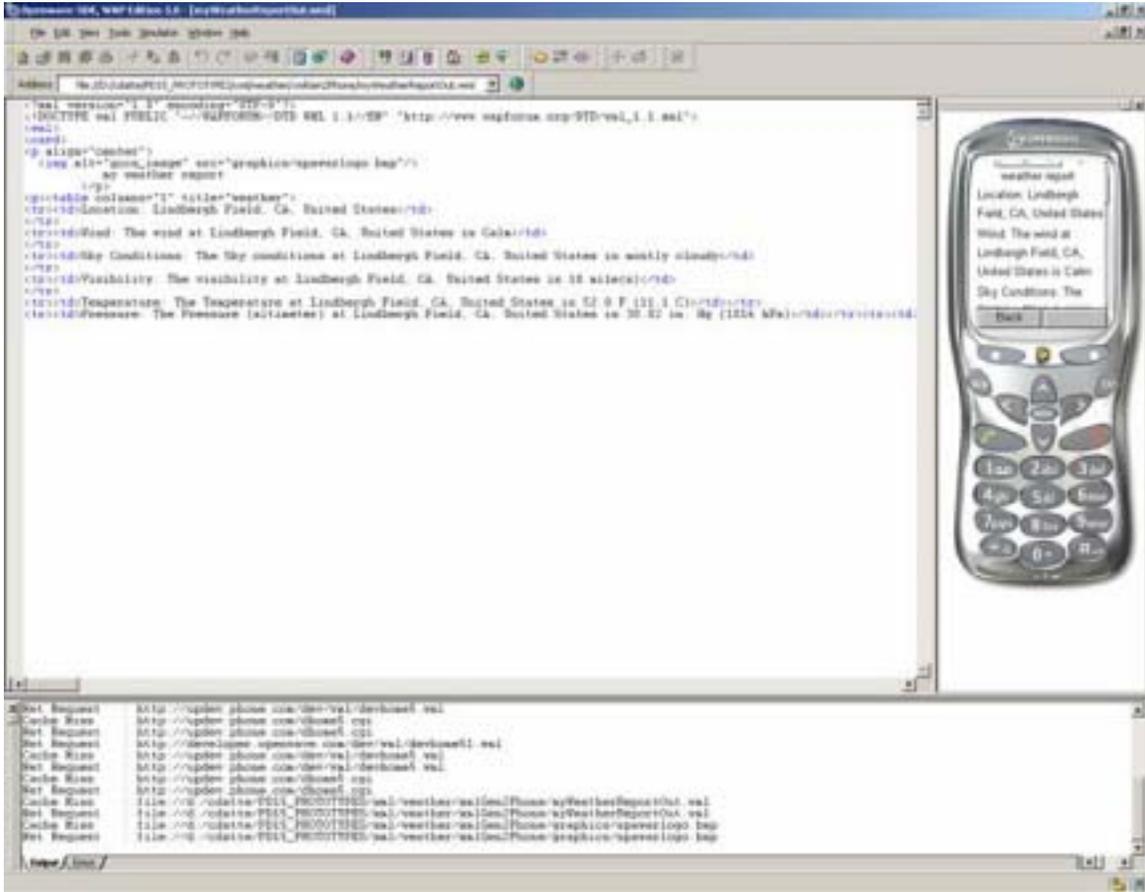
### Target platforms

The wireless cell phone environment has three main target platforms:

Platform	Description	Download from
<i>Openwave</i> (shown below)	Has a freeware development environment	<a href="http://www.openwave.com">http://www.openwave.com</a>
<i>Nokia</i>	Has a freeware development environment	<a href="http://www.forum.nokia.com">http://www.forum.nokia.com</a>
<i>Motorola</i>	Has some freeware development tools	<a href="http://kb.motorola.metrowerks.com/motorola/pcsHome.do">http://kb.motorola.metrowerks.com/motorola/pcsHome.do</a>

Emulators are included in the development environments.

This figure shows an example of the Openwave *IDE*:



## Testing

### To test a wireless cell phone:

1. Test it in the IDE to verify correct operation.
2. Test it through a wireless gateway when it is ready to go for live testing.

## PalmOS 4

### Overview

There are three main development environments for developing palm applications with Palm OS 4 and below.

Environment	Description	Download from
PRC-TOOLS GCC	PRC-TOOLS GCC environment is a freeware development environment	<a href="http://www.palmos.com/dev/tools/gcc/">http://www.palmos.com/dev/tools/gcc/</a>

	<p>for PalmOS 4 development. It contains these components:</p> <ul style="list-style-type: none"> <li>• Cygwin</li> <li>• PRC-Tools</li> <li>• PiIRC</li> </ul>	
<i>Codewarrior IDE</i>	<p>Codewarrior IDE is a product of Metrowerks.</p> <p>The Codewarrior IDE suite contains two products, one for Java and one for C/C++. Use the Java version for J2ME devices and the C/C++ version for PalmOS. The suite also contains a GUI builder IDE called the Constructor.</p>	<p><a href="http://www.metrowerks.com/">http://www.metrowerks.com/</a> or <a href="http://www.palmos.com/developers">http://www.palmos.com/developers</a></p>
<i>Falch.net IDE</i>	<p>Falch.net IDE is a product of Falch.net.</p>	<p><a href="http://www.falch.net">http://www.falch.net</a></p>

## Recommendations

1. Use the constructor tool to build Palm GUIs. Do not build your own.
2. Use the Palm emulator to test your Palm applications.

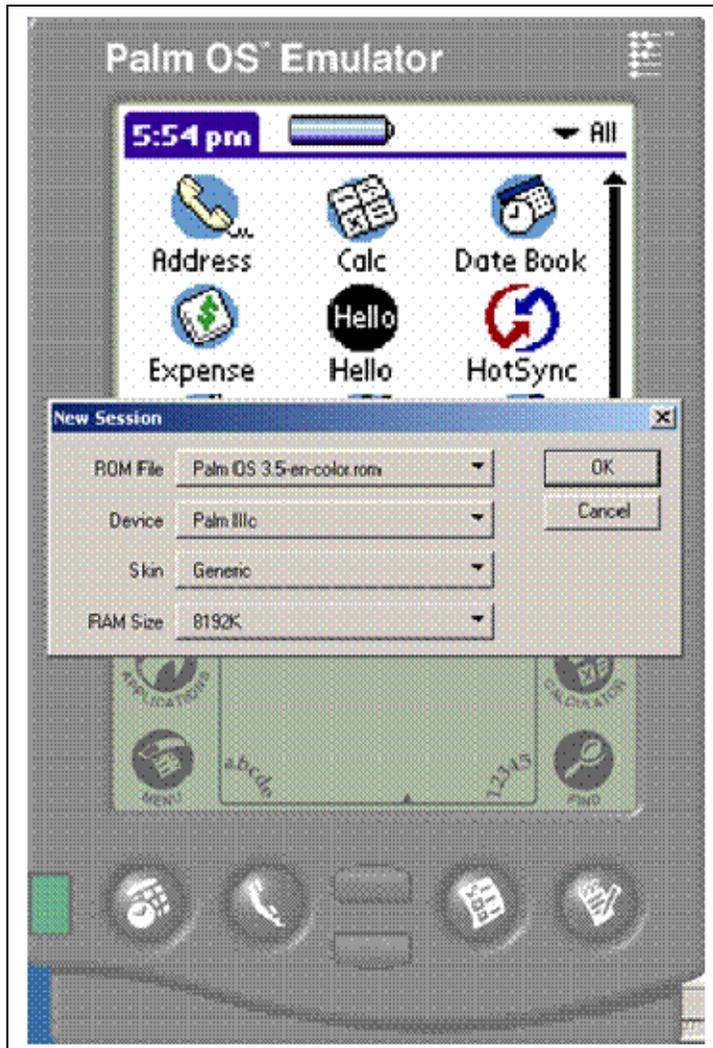
See *Necessary components* for instructions on how to obtain these components.

## Necessary components

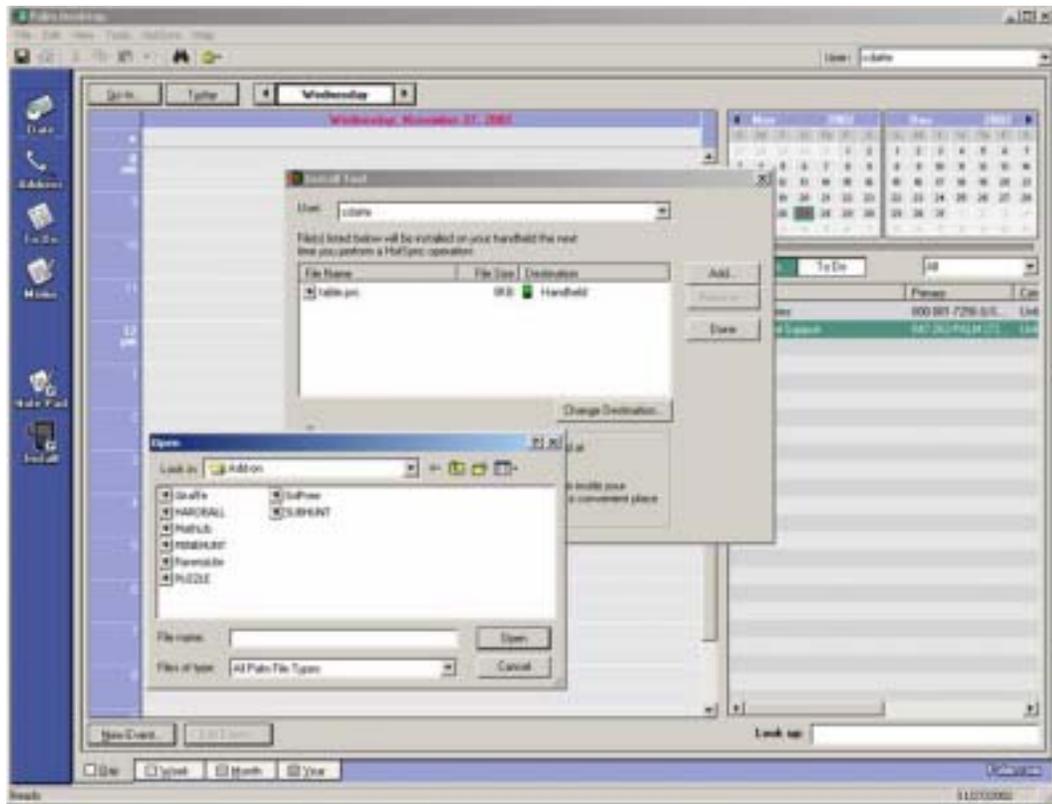
### To set up the full development environment for PalmOS:

- Download the PalmOS **SDK** from <http://www.palmos.com/dev/tools/sdk/index.html>. This contains resources for building Palm applications. The most important resources are:
  - The full documentation suite
  - Include files for the different development IDEs
  - Libraries for linking the applications
  - The constructor tool, which is an IDE-like tool for building the GUIs
- Obtain the Palm emulator called POSE, shown below, from <http://www.palmos.com/dev/tools/simulator/>.

To emulate any specific device, you need the ROM image that is compatible with that device. These are loaded into the emulator when you start POSE. You should use gremlins in the emulator when testing Palm applications.



- Download skins and ROMs for the emulator from <http://www.palmos.com/dev/tools/emulator/>.
- Download the conduit development kit from <http://www.palmos.com/dev/tools/cdk/win/>.
- Download the Desktop SDK for developing Palm desktop plug-ins from <http://www.palmos.com/dev/tools/emulator/>. Palm applications interface with the desktop via the Palm Desktop Application. You can add application functionality to the desktop through the Palm Desktop SDK, shown here:



## References

The best way to get started is to get a book with examples that build out the basic skeleton, then clone that example and use it to build your application. The *Palm OS Programming Bible* by Lonnon R. Foster is a useful starting point.

For documentation, further examples, and SDKs, go to the open-source site, <http://www.palmos.com/developers>.

---

# Guidance

## Guidance details

This section contains a complete set of the numbered guidance statements that are referenced elsewhere in this guide.

# G1001

<b>Statement</b>	Define public interfaces using a formal standard.		
<b>Rationale</b>	<p>It's important that a common language is used to define the interfaces so producers and consumers can work independently and together.</p> <p>There are many standards for defining interfaces (<i>UML</i>, <i>WSDL</i>, and <i>CORBA</i>). The standard used must be documented and widely accepted by the industry.</p>		
<b>Derived From</b>			
<b>Justifies</b>			
<b>Referenced By</b>	<i>Publish and insulate public interfaces</i>		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Do UML documents exist that describe the shared interfaces?</i>
		<b>Procedure</b>	Ask for the design documents to be provided during the review process.
		<b>Examples</b>	None
	<b>2.</b>	<b>Test</b>	<i>Are there WSDL files that document the interface to web services?</i>
		<b>Procedure</b>	Look for the existence of .WSDL files.
		<b>Examples</b>	None
	<b>3.</b>	<b>Test</b>	<i>Are there <b>IDL</b> files that document the interfaces to CORBA services?</i>
		<b>Procedure</b>	Look for the existence of .idl files.
		<b>Examples</b>	None

# G1002

<b>Statement</b>	Separate public interfaces from implementation.	
<b>Rationale</b>	<p>This guidance encourages clean separation between <i>interface</i> and implementation details for all types of application development. This allows components and systems to be <i>loosely coupled</i>. The flexibility allows groups of developers to work independently and in parallel to the contract defined by the interface.</p> <p>Another benefit of hiding implementation details is that it allows the implementation to change without affecting users of the interface. This means the interface can support dynamic and pluggable implementation.</p>	
<b>Derived From</b>		
<b>Justifies</b>	[G1217], [G1218], [G1219], [G1220], [G1221]	
<b>Referenced By</b>	<i>Publish and insulate public interfaces</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b> <i>C++: Check to make sure interfaces are defined as pure virtual functions.</i></p> <p><b>Procedure</b> Make sure C++ classes are defined in header files. Classes that represent external interfaces should contain only pure virtual functions. Make sure the class does not declare non-constant data members. Also, make sure it does not define default implementation. An interface should provide no default behavior.</p>
	2.	<p><b>Test</b> <i>C: Check to make sure functions are declared in a header file using prototypes.</i></p> <p><b>Procedure</b> Make sure each library function has a prototype declaration in the header file.</p> <p><b>Examples</b> None</p>

# G1003

<b>Statement</b>	Separate the contents of application libraries that are to be shared from libraries that are to be used internally.	
<b>Rationale</b>	<p>The public libraries that are intended to be shared with outside consumers need to remain fairly static in order to facilitate independent development by the <i>consumer</i> and the <i>producer</i> of the libraries' functionality. Changes in libraries should be mutually agreed upon by both the producer and the consumer.</p> <p>All library content should not have external dependencies that are not related to supporting the interface.</p> <p>There must be clear separation between domain-specific and shared libraries. Libraries that will be used in joint or multiple projects should not have domain-specific code.</p>	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	<i>Publish and insulate public interfaces</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Do the publicly shared libraries have any private or undocumented functionality?</i></p> <p><b>Procedure</b> Check each library against the publicly defined header and make sure that all objects or methods are public.</p> <p><b>Examples</b> None</p>
	<b>2.</b>	<p><b>Test</b> <i>Does the library contain extraneous interfaces or code that is not required?</i></p> <p><b>Procedure</b> Use coverage tool/Junit to make sure there is no extraneous code.</p> <p><b>Examples</b> None</p>
	<b>3.</b>	<p><b>Test</b> <i>Do the publicly shared libraries have any private or undocumented functionality?</i></p> <p><b>Procedure</b> Check to make sure that one library use of another library does not cross domain-specific boundaries. For instance, a common library of <i>XML</i> utilities should not have dependencies on another library that supports a specific <i>domain</i> such as UHF satellites. However, the reverse is okay.</p> <p><b>Examples</b> None</p>

# G1004

**Statement** Public interfaces shall be backward-compatible, within the constraints of a published deprecation policy.

**Rationale** The public interface is basically a contract between the *producer* of the functionality defined in an interface and the *consumer* of the functionality. These guidance statements are intended to ensure that this contract remains intact and that the consumer of the functionality is not broken during the update cycle of the interface.

**Derived From**

**Justifies** [G1018], [G1019], [G1020], [G1206], [G1207], [G1208]

**Referenced By** *Publish and insulate public interfaces*

**Acquisition Phase** Development

<b>Evaluation Criteria</b>	1.	<b>Test</b>	<i>Does the public interface (interfaces that are used externally, outside the project's <b>domain</b>) contain versioning information?</i>
		<b>Procedure</b>	Check to make sure the interface/class has versioning information.
		<b>Examples</b>	None
	2.	<b>Test</b>	<i>Does the document structure contain a document that indicates the shelf life of deprecated interfaces?</i>
		<b>Procedure</b>	Note: This is a mandatory document  Check for project documents that have information on the life of deprecated interfaces.
		<b>Examples</b>	None

# G1005

## Statement

Separate infrastructure capabilities from *mission* functions.

## Rationale

Applications should not try to reinvent the wheel by creating custom *enterprise services* such as messaging, directory services, logging, etc. Application development should use standardized *APIs* to access common enterprise services. For instance, in Java, use *JMS* to access a messaging system.

## Derived From

## Justifies

## Referenced By

*Publish and insulate public interfaces*

## Acquisition Phase

Development

## Evaluation Criteria

- |    |                  |   |
|----|------------------|---|
| 1. | <b>Test</b>      | <i>Does the application re-create common and available enterprise services?</i>                   |
|    | <b>Procedure</b> | Check the application code for code that recreates functionality of an enterprise service.        |
|    | <b>Examples</b>  | None  |
| 2. | <b>Test</b>      | <i>Does the application code access enterprise services in a vendor-specific way?</i>             |
|    | <b>Procedure</b> | Check for code that accesses a vendor-specific API instead of utilizing an industry-standard API. |
|    | <b>Examples</b>  | None  |

# G1007

<b>Statement</b>	Applications shall use open, standardized, vendor-neutral <b>APIs</b> .		
<b>Rationale</b>	Using standardized, open APIs will enable the code to be more portable. It will also prevent vendor lock-in. "Standardized" means industry consensus. "Open" means available to everyone.		
<b>Derived From</b>			
<b>Justifies</b>	[G1071]		
<b>Referenced By</b>	<i>Publish and insulate public interfaces</i>		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	1.	<b>Test</b>	<i>Does the application create customized/proprietary solutions where standardized API exists?</i>
		<b>Procedure</b>	Check the application for code that has proprietary solutions where standardized API exists. For instance, does the application write its own messaging system, bypassing utilizing the <b>Java Messaging System</b> API.
		<b>Examples</b>	None
	2.	<b>Test</b>	<i>Does the application utilize vendor-specific API?</i>
		<b>Procedure</b>	Check the application to make sure it is not using a vendor-specific API. For instance, see if the application accesses the database using a proprietary interface from Oracle instead of the standard <b>JDBC</b> calls.
		<b>Examples</b>	None

# G1008

<b>Statement</b>	Isolate platform-specific interfaces and vendor dependencies.	
<b>Rationale</b>	Insulating platform-specific code using standard abstractions or custom classes will keep all non-portable code in one place and prevent proliferation of non-portable code throughout the application.	
<b>Derived From</b>		
<b>Justifies</b>	[G1073]	
<b>Referenced By</b>	<i>Publish and insulate public interfaces</i> , [G1118]	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Does the application contain any platform-specific code that has not been abstracted?</i></p> <p><b>Procedure</b> Check code that is non-portable. For instance, does the code use back slashes (Windows) or forward slashes (UNIX) in literal strings to create a path.</p> <pre>IE: String path = "\\tmp";</pre> <p><b>Examples</b> None</p>
	<b>2.</b>	<p><b>Test</b> <i>Is platform-specific code isolated into a single class or file?</i></p> <p><b>Procedure</b> Search the files for platform-specific code.</p> <p><b>Examples</b> None</p>

## G1010

<b>Statement</b>	Use open-standards logging frameworks.
<b>Rationale</b>	Standardizing on one logging <i>API</i> means the code will be more portable between developers, and developers no longer need to learn multiple logging frameworks.
<b>Derived From</b>	
<b>Justifies</b>	[G1209], [G1210]
<b>Referenced By</b>	<i>Publish and insulate public interfaces</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel guidance.

# G1011

<b>Statement</b>	All components must be independently deployable.							
<b>Rationale</b>	Independently deployable components do not have any dependencies on other components. This is often unattainable because components are often aggregations of lower-level components. Exceptions to this rule can occur if the relationships between components:							
	<ul style="list-style-type: none"> <li>• Are well-defined and well thought out</li> <li>• Are carefully managed</li> <li>• Are externally configurable</li> </ul>							
<b>Derived From</b>								
<b>Justifies</b>								
<b>Referenced By</b>	<i>Implement a component-based architecture</i>							
<b>Acquisition Phase</b>	Development							
<b>Evaluation Criteria</b>	1.	<table> <tr> <td><b>Test</b></td> <td><i>Is the component dependent on other components?</i></td> </tr> <tr> <td><b>Procedure</b></td> <td>{Place the procedure to follow to evaluate the test question here. The procedure can be multiple steps}</td> </tr> <tr> <td><b>Examples</b></td> <td>None</td> </tr> </table>	<b>Test</b>	<i>Is the component dependent on other components?</i>	<b>Procedure</b>	{Place the procedure to follow to evaluate the test question here. The procedure can be multiple steps}	<b>Examples</b>	None
<b>Test</b>	<i>Is the component dependent on other components?</i>							
<b>Procedure</b>	{Place the procedure to follow to evaluate the test question here. The procedure can be multiple steps}							
<b>Examples</b>	None							

# G1012

<b>Statement</b>	Components should expose functionality through a set of services.	
<b>Rationale</b>	By exposing discrete units of functionality as <i>services</i> , business and data integrity remain intact. A service receives a request, processes it, and returns the result to the requester as a single operation.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	<i>Implement a component-based architecture</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b> <i>Are there <b>WAR</b> files that contain the component?</i>
		<b>Procedure</b> Check for the occurrence of <code>.war</code> files.
		<b>Examples</b> None.
	<b>2.</b>	<b>Test</b> <i>Are there <b>WSDL</b> files that define the services?</i>
		<b>Procedure</b> Check for the occurrence of <code>.wsdl</code> files.
		<b>Examples</b> None.

# G1014

<b>Statement</b>	Access the database only through <i>open-standards</i> interfaces to promote database independence.
<b>Rationale</b>	Standard <i>API</i> (s) such as <i>JDBC</i> or <i>ODBC</i> promote database independence. However, even if you use a standard API, you can still write non-portable code if you use non- <i>ANSI</i> -compliant <i>SQL</i> . Using non- <i>ANSI</i> -compliant <i>SQL</i> causes vendor lock-in and makes <i>interoperability</i> difficult.
<b>Derived From</b>	
<b>Justifies</b>	[ <i>G1211</i> ], [ <i>G1212</i> ]
<b>Referenced By</b>	
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel guidance to evaluate this guidance.

# G1018

**Statement** Add version numbers/ identifiers to all public interfaces that will be shared between projects or groups.

**Rationale** Assigning versions is necessary when determining compatibility between the *interface* and its *consumer*. Versioning public interfaces allows all parties to track the evolution of the interface for backward compatibility. This can help consumers plan for integration and migration.

**Derived From** [G1004]

**Justifies**

**Referenced By**

**Acquisition Phase** Development

**Evaluation Criteria**

- Test** *Ensure that version information can be identified. Does the code contain versioning information? It is important to have the version information in the shared public interface code because it identifies the actual interface that consumers of the interface will be coding to. Another benefit is that it allows tools to automatically generate the documentation so it does not need to be in two places.*

**Procedure** For Java, check for @version javadoc tag.

For other languages, and Java, check to see if the code is annotated using *XML* tags or language-specific tags that support versioning.

**Examples** None

# G1019

<b>Statement</b>	Deprecate old versions of publicly shared interfaces and do not remove them until a specified time period has passed, as defined by the project document for deprecating obsolete interfaces.	
<b>Rationale</b>	By deprecating instead of removing interfaces, development teams can plan for software migration and continue to run the software with existing deprecated interfaces.	
<b>Derived From</b>	[G1004]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Are old versions of public interfaces marked as deprecated?</i></p> <p><b>Procedure</b> Check the SCM logs of public interface files to ensure that old interface functionality has not been removed.</p> <p><b>Examples</b> None</p>

## G1020

<b>Statement</b>	A project must provide additional documents that describe plans and procedures that can be used to evaluate the project's compliance.
<b>Rationale</b>	To ensure a <i>NESI</i> evaluation can be performed, these documents must be provided.
<b>Derived From</b>	[G1004]
<b>Justifies</b>	[G1213], [G1214], [G1215], [G1216]
<b>Referenced By</b>	
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel guidance to evaluate this guidance

# G1021

**Statement**

Create fully insulated classes.

**Rationale**

Data members should not be public.

Do not expose implementation details of a class. For instance, information such as the use of a link list or `hashtable` in a class should not be exposed (i.e. made public).

Making implementation details public creates interdependencies between the class and its users, subjecting the users to changes in implementation. Therefore, access should only occur via public interface methods. This makes the implementation more robust, because all data can be validated when assigned new values or the changes can be logged.

**Derived From**
**Justifies**
**Referenced By**
**Acquisition Phase**

Development

**Evaluation Criteria**

- |           |                  |  |
|-----------|------------------|--|
| <b>1.</b> | <b>Test</b>      | <i>Do instance variables have public access or are they more accessible than necessary?</i>                        |
|           | <b>Procedure</b> | Check that the instance variable in classes does not have public access unless it is static and final.             |
|           | <b>Examples</b>  | None   |
| <b>2.</b> | <b>Test</b>      | <i>Does the class provide direct access to internal data via pass by reference?</i>                                |
|           | <b>Procedure</b> | Check to make sure that the methods that access the internal state do not return a reference to the internal data. |
|           | <b>Examples</b>  | None   |

# G1022

**Statement**

Insulate public interfaces from compile-time dependencies.

**Rationale**

There are three distinct advantages to separating interface from implementation:

- Multiple interested parties (*COIs*) can develop the interface and publish it to the user community ahead of any specific implementation. This allows groups to work independently and in parallel.
- It prevents multiple copies of the defining interface. Duplicating the code for the interface in each implementation (library, jar, and assembly) makes it difficult to maintain, especially as the interface evolves.
- It insulates developers from the constant changes in implementation.

**Derived From**

**Justifies**

**Referenced By**

*Publish and insulate public interfaces*

**Acquisition Phase**

Development

**Evaluation Criteria**

- |    |                  |   |
|----|------------------|---|
| 1. | <b>Test</b>      | <i>Is the packaging or deployment of the public interface self-contained and isolated to only the public interface(s)?</i>  |
|    | <b>Procedure</b> | Check to make sure that the jar, library, assembly, and <i>WSDL</i> only contain the agreed-upon public interface (interfaces being shared externally).   |
|    | <b>Examples</b>  | None  |
| 2. | <b>Test</b>      | <i>Does the container (jars, libraries, assemblies, WSDL) contain files other than the interface?</i>   |
|    | <b>Procedure</b> | Check to make sure the library does not include or rely upon any other files such as resource files, properties files, configuration files, other libraries, xml files, and so on that would force the repackaging of the public interface. |
|    | <b>Examples</b>  | None  |
| 3. | <b>Test</b>      | <i>Are there any outside influences that could affect the packaging of the public interface?</i>  |
|    | <b>Procedure</b> | Check the public interface for dependence on resource files, properties files, configuration files, XML files, and other libraries or packages.   |

**Examples**    None

# G1027

<b>Statement</b>	All source code developed with DoD funding must be internally documented.		
<b>Rationale</b>	<p>Well-documented source code is easier to maintain and enhance over time. It is hard enough to get documentation about software and to keep it up to date. If the documentation is not internal to the source code, the chances that the software is current and up-to-date decreases. In recent years, the trend has been to generate external documentation about the software by processing the source code and comments (e.g., JavaDoc).</p> <p>In addition to documenting the functionality of the source code, it is important to capture the configuration control information (e.g., CVS).</p>		
<b>Derived From</b>			
<b>Justifies</b>			
<b>Referenced By</b>	Standard interface documentation		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Do all the source code files have a header that includes a statement protecting government rights to the source code and the right to change the source code?</i>
		<b>Procedure</b>	Scan each file and make sure the header includes a statement that protects the government's right to use, modify, and share the information with other government departments and agencies.
		<b>Examples</b>	None
	<b>2.</b>	<b>Test</b>	<i>Do all the source code files have a header that includes configuration information?</i>
		<b>Procedure</b>	Scan each file and make sure the header also includes configuration management information such as author, date created, and a history of modifications and versions.
		<b>Examples</b>	None
	<b>3.</b>	<b>Test</b>	<i>Do all the source code files have internal documentation for attributes, methods that can be processed by a computer?</i>
		<b>Procedure</b>	Scan the source files and make sure they are internally documented with tags such as JavaDoc or XML tags.
		<b>Examples</b>	None

# G1030

## Statement

Use a standard GUI *component* library.

## Rationale

A predefined component library helps control cost and configuration. Licensing issues can be resolved before development begins, and component costs are minimized by avoiding library overlap.

Now that component architecture is standard, it is possible to put together applications using a variety of components from multiple vendors. These components are bundled in third-party toolkits that vastly extend the range of options available in standard Windows or Java GUI toolkits. These toolkits are in common use and possess a wide variety of pre-built components. Almost all support common *look-and-feel* (e.g., Windows or Java).

## Derived From

## Justifies

## Referenced By

Thick clients

## Acquisition Phase

Development

## Evaluation Criteria

- |    |                  |  |
|----|------------------|--|
| 1. | <b>Test</b>      | <i>Does the user interface code use any other toolkits besides a Standard GUI Toolkit?</i>   |
|    | <b>Procedure</b> | Check to make sure the thick-client code is developed using the Swing/AWT library in Java, and the standard, included Windows Toolkit In .NET. |
|    | <b>Examples</b>  | None   |

# G1031

**Statement** Architect applications to cleanly separate the presentation, business, and data layers.

**Rationale** This guidance applies to all application types, from thick-client standalone applications to distributed *web applications*. Clean separation between presentation, business, and data layers will allow the application to be easier to maintain and more reusable.

**Derived From**

**Justifies**

**Referenced By**

**Acquisition Phase** Development

**Evaluation Criteria**

**1. Test**

*Presentation layer:*

- Check that the presentation layer does not access the data layer directly.
- Check the presentation layer for the presence of *business logic*.

*Business layer:*

- Check to make sure the business does not contain any GUI code.
- Make sure access to the data layer is insulated to data access interface.

*Data layer:*

- Check to make sure the data layer does not contain GUI code.
- Check to make sure the data layer does not contain business logic.

**Procedure**

Presentation layer:

- Check the presentation layer for *JDBC*, *SQL*, or *ODBC* code.
- Make sure code such as specialized data processing algorithms, or code that manages workflow is not in the presentation tier.

Business layer:

- Check the business layer to make sure it does not import GUI libraries or GUI components.
- Make sure database code such as SQL and JDBC are isolated using Data Access Pattern; data tier code should not proliferate

throughout the middle tier.

- Make sure Value Object Pattern is used for data transfer between the middle and data layer.

Data layer:

- Make sure the data layer is not responsible for generating GUI code.
- Make sure the data layer does not perform any business logic. Look for use of *stored procedures*.

**Examples**

None

# G1032

<b>Statement</b>	Validate all input fields.	
<b>Rationale</b>	Errors should be detected as close to point-of-data-entry as possible. This greatly enhances the end-user experience and reduces frustration. This can be done by reducing the number of freeform text fields and using selection mechanisms such as radio buttons, option boxes, pull down lists, maps, calendars, clocks, slider bars, and other numeric validation entries.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	Presentation Tier	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b>      <i>Do the GUI screens use non-freeform text entry fields?</i></p> <p><b>Procedure</b>      Scan the GUI code looking for the use of non-freeform text data entry mechanisms.</p> <p><b>Examples</b>      None.</p>

# G1035

**Statement** Code must not deviate from *W3C standards or use vendor-specific add-on features.*

**Rationale** Code cannot be browser-independent if vendor-specific add on features are used. Vendor-specific add-on features reduce the portability and *interoperability* of the code. Vendor-specific *API*(s) can cause vendor lock-in and in many cases can also cause version lock-in. Following the W3C standard avoids these problems.

**Derived From**

**Justifies**

**Referenced By** GUI design

**Acquisition Phase** Development

<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Does the code adhere strictly to the W3C standards?</i>
		<b>Procedure</b>	Check to make sure there is no vendor-specific code.
		<b>Examples</b>	None

# G1037

<b>Statement</b>	File type must match file content.	
<b>Rationale</b>	<p>Makes the code easier to read, maintain, and port to other environments if information in a file is consistent with its file type and separated according to functionality.</p> <p><i>JavaScript</i> files (.js) and cascading style sheets (.css) can be cached by browsers for better performance.</p> <p>This guidance also prevents mixing contents, such as including JavaScript in an html file.</p>	
<b>Derived From</b>		
<b>Justifies</b>	[G1053]	
<b>Referenced By</b>	GUI design	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Is there any other information stored in an .html file other than HTML?</i></p> <p><b>Procedure</b> Check the contents of the .html file for the inclusion of JavaScript or <i>style sheet</i> information.</p> <p><b>Examples</b> None</p>
	<b>2.</b>	<p><b>Test</b> <i>Is there any other information stored in a .js file other than JavaScript?</i></p> <p><b>Procedure</b> Check the contents of the .html file for the inclusion of HTML or style sheet information.</p> <p><b>Examples</b> None</p>
	<b>3.</b>	<p><b>Test</b> <i>Is there any other information stored in a .css file other than style sheet information?</i></p> <p><b>Procedure</b> Check the contents of the .html file for the inclusion of HTML or JavaScript.</p> <p><b>Examples</b> None</p>

## G1043

<b>Statement</b>	Decouple the graphical style from the content format.	
<b>Rationale</b>	Makes it easy to change the style for the entire site.	
<b>Derived From</b>		
<b>Justifies</b>	[G1044]	
<b>Referenced By</b>	GUI design	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b>      <i>Do all web document <b>HTML</b>, <b>JSP</b>, <b>ASP</b>, and <b>CSS</b> follow the Disability Act guidelines?</i></p> <p><b>Procedure</b>      Check to make sure all web documents follow the guidelines.</p> <p><b>Examples</b>      None</p>

# G1044

<b>Statement</b>	Web documents shall comply with Disability Act guidelines.	
<b>Rationale</b>	These guidelines benefit all communities of interest. For more information, see <a href="http://www.section508.gov">http://www.section508.gov</a> or <a href="http://www.w3.org/TR/WAI-WEBCONTENT/">http://www.w3.org/TR/WAI-WEBCONTENT/</a>	
<b>Derived From</b>	[G1043]	
<b>Justifies</b>		
<b>Referenced By</b>	GUI design	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Do all web document <b>HTML, JSP, ASP, and CSS</b> follow the Disability Act guidelines?</i>
	<b>Procedure</b>	Check to make sure all web documents follow the guidelines.  Use available validation tools to validate Section 508 accessibility and WAI accessibility. Go to <a href="http://www.contentquality.com/Default.asp">http://www.contentquality.com/Default.asp</a> to validate the page.
	<b>Examples</b>	None

# G1045

<b>Statement</b>	Define <i>XML</i> format information separately in <i>XSL</i> .	
<b>Rationale</b>	XML documents should be free of any presentation information and should only contain data. Separating presentation data from content allows multiple presentations for the same content data.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	XML rendering	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b>      <i>Check for presentation information in XML documents?</i></p> <p><b>Procedure</b>      Does the XML document contain only data? If the XML document is not an <i>XSLT</i> document, does it contain presentation information?</p> <p><b>Examples</b>      None</p>

# G1049

<b>Statement</b>	Do not use <i>ActiveX</i> controls.		
<b>Rationale</b>	Browser incompatibility poses serious security risk, because it does not run inside a sandbox. ActiveX controls are like <i>applets</i> , except they are not restricted by a sandbox and can access client machine resources such as the hard disk directly. This makes them very dangerous.		
<b>Derived From</b>			
<b>Justifies</b>			
<b>Referenced By</b>	Active Server Pages (ASP)		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Does the ASP use any ActiveX controls?</i>
		<b>Procedure</b>	Check for Active X controls inside web pages.
		<b>Examples</b>	None

## G1050

<b>Statement</b>	In <b>ASP</b> , isolate the presentation tier from the middle tier using <b>COM</b> objects.	
<b>Rationale</b>	This is the best way to isolate the presentation tier from the middle tier in ASP.	
<b>Derived From</b>	[G1058]	
<b>Justifies</b>		
<b>Referenced By</b>	Active Server Pages (ASP)	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Is all the middle tier code isolated from the presentation tier in ASP via COM?</i></p> <p><b>Procedure</b> Verify that ASP files do not contain middle-tier code. Instead, this code should be in COM objects referenced from the ASP.</p> <p><b>Examples</b> None</p>

# G1052

**Statement** Use the code-behind feature in ASP.NET to separate presentation code from the business logic.

**Rationale** Separating presentation code from business logic allows the developers and content designers to work independently. It also makes the code more maintainable because changes in the design elements or business elements do not affect each other.

**Derived From**

**Justifies**

**Referenced By** Active Server Pages for .NET (ASP.NET)

**Acquisition Phase** Development

- Evaluation Criteria**
1. **Test** *Is there code in ASP pages?*
  - Procedure** Check to make sure that ASP files have the code-behind attribute in the first line instead of embedded C# code in the ASP.
  - Examples** None

# G1053

<b>Statement</b>	Do not embed HTML code in any code-behind code used by aspx pages.	
<b>Rationale</b>	Intermixing VB or C# or C++ with presentation code (HTML) makes the code unnecessarily difficult to maintain by both the developer and designer. This is similar in concept to Java's not embedding HTML code in <i>servlets</i> .	
<b>Derived From</b>	[G1037], [G1058]	
<b>Justifies</b>		
<b>Referenced By</b>	Active Server Pages for .NET (ASP.NET)	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b>      <i>Check for HTML code in code-behind code.</i></p> <p><b>Procedure</b>      Check the code-behind file (<b>.aspx.vb</b> for example) for any HTML tags.</p> <p><b>Examples</b>      None</p>

# G1055

**Statement** Use a fully qualified, registered *namespace* with identity information for all custom controls.

**Rationale** *.NET* allows users to create a custom control from a web page. This allows the custom web page to be reusable just like a GUI control. This feature is great; however, users must fully qualify their controls to prevent namespace collisions.

**Derived From**

**Justifies**

**Referenced By** Active Server Pages for .NET (ASP.NET)

**Acquisition Phase** Development

- Evaluation Criteria**
- 1. **Test** *Does the ASP register its identity?*
  - Procedure** Check the `.aspx` file and make sure there is a statement to register the custom control. Look for something similar to
  - Examples** None

# G1056

<b>Statement</b>	Specify a versioning policy for <i>.NET</i> assemblies.						
<b>Rationale</b>	Versioning assemblies and configuring dependent assemblies allow the <i>Common Language Runtime (CLR)</i> to load the proper assemblies at runtime for your application. This insulates the application from system configuration changes.						
<b>Derived From</b>							
<b>Justifies</b>							
<b>Referenced By</b>	Active Server Pages for .NET (ASP.NET)						
<b>Acquisition Phase</b>	Development						
<b>Evaluation Criteria</b>	<ol style="list-style-type: none"> <li> <table> <tr> <td><b>Test</b></td> <td><i>Does the application assembly have versioning information?</i></td> </tr> <tr> <td><b>Procedure</b></td> <td>           Check the application assembly manifest for versioning information.             Use the .NET configuration tool to check for versioning policy and versioning information.         </td> </tr> <tr> <td><b>Examples</b></td> <td>None</td> </tr> </table> </li> </ol>	<b>Test</b>	<i>Does the application assembly have versioning information?</i>	<b>Procedure</b>	Check the application assembly manifest for versioning information.  Use the .NET configuration tool to check for versioning policy and versioning information.	<b>Examples</b>	None
<b>Test</b>	<i>Does the application assembly have versioning information?</i>						
<b>Procedure</b>	Check the application assembly manifest for versioning information.  Use the .NET configuration tool to check for versioning policy and versioning information.						
<b>Examples</b>	None						

# G1058

**Statement** Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.

**Rationale** Separating data-layer code from presentation-layer code provides the ability to base multiple views on the same model. This is especially important in the enterprise model because often, the user interface varies with the device (browser, mobile phone, thick client, etc.).

Isolating different layers allows changes to occur in each layer without impacting other layers. For instance, if the data layer (model) decides to switch databases, the changes are isolated to the data layer and do not affect the view layer or controller layer.

Lastly, because MVC architecture enforces separation between presentation, processing, and data layer, this allows functionality to be loosely coupled and therefore more suited for reuse.

**Derived From**

**Justifies** [G1050], [G1053]

**Referenced By**

**Acquisition Phase** Development

**Evaluation Criteria** 1. **Test** *Does the application use a Model 2 (MVC) pattern?*

**Procedure** Check to see if all requests are being mapped to a single controller servlet.

Check that all page rendering are being done by a **JSP** and not a *servlet*.

2. **Test** *Does the application enforce clear separation between data layer (model), presentation layer (view), and middle/business layer (controller)?*

**Procedure** Check to make sure the application presentation is not accessing the database directly.

Check to make sure the application data layer (model) is not implementing business logic (store procedures).

Check to make sure the middle/business layer (controller) does not contain presentation code. For example, make sure servlets do not generate HTML.

Make sure access to the database is isolated to Data Access Object instead of proliferated throughout the middle layer.

**Examples** None

# G1060

**Statement**

Encapsulate Java code that is used in *JSP*(s) in tag libraries.

**Rationale**

Separating code from presentation allows developers and designers to work independently. It makes the code reusable and more maintainable because it is defined in a tag library.

**Derived From****Justifies****Referenced By**

Java Server Pages (JSP)

**Evaluation  
Criteria**

- Test** *Do the JSP pages use tag libraries?*  
**Procedure** Look through the JSP pages for embedded Java source code.  
**Examples** None

# G1071

**Statement** Connections to the enterprise (e.g., *LDAP*, *JNDI*, *JMS*, databases) should use vendor-neutral interfaces.

**Rationale** Increases *portability* and maintainability. Many of the newer connection mechanisms are vendor-neutral. Use these instead of isolation design patterns or vendor-specific connection mechanisms.

**Derived From** [G1007]

**Justifies**

**Referenced By** [G1239], *Java Naming & Directory Interface (JNDI)*

**Acquisition Phase** Development

- Evaluation Criteria**
- 1. **Test** *Is the connection mechanism vendor-neutral?*
  - Procedure** Examine the source code for vendor-specific imports or includes. Make sure only standard APIs are used.
  - Examples** None

# G1073

<b>Statement</b>	Isolate vendor extensions to enterprise-services standard interfaces.	
<b>Rationale</b>	Vendor extensions are convenient, but help create "vendor lock" and reduce vendor neutrality and migration. It is best to avoid these extensions altogether. If that is not possible, then isolate them in an <i>adapter</i> or a wrapper-like construct.	
<b>Derived From</b>	[G1008]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Are vendor extensions to enterprise services used?</i></p> <p><b>Procedure</b> Make sure that no vendor-specific code is included or imported except as part of an adapter or wrapper.</p> <p><b>Examples</b> None</p>

# G1078

<b>Statement</b>	Document the use of vendor-specific <i>J2EE</i> deployment descriptors.
<b>Rationale</b>	Deployment descriptors that are not defined by the J2EE specification are not portable between <i>application servers</i> . For example, BEA WebLogic has a vendor-specific deployment descriptor called <b>weblogic-ejb-jar.xml</b> and JBoss has a vendor specific deployment descriptor called <b>jboss-jar.xml</b> .
<b>Derived From</b>	
<b>Justifies</b>	
<b>Referenced By</b>	<i>J2EE environment</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	<p><b>1. Test</b> <i>Are all the XML files that are not part of the J2EE specification identified in a delivered document?</i></p> <p><b>Procedure</b> Search all XML documents in the META-INF and WEB-INF directories and identify any XML files that are not defined by J2EE. These files should be found in a README or other delivered file that describes their purpose.</p> <ul style="list-style-type: none"> <li>a. Web application WEB-INF/web.xml</li> <li>b. EJB JAR META-INF/ejb-jar.xml</li> <li>c. J2EE Connector META-INF/ra.xml</li> <li>d. Client application META-INF/application-client.xml</li> <li>e. Enterprise application META-INF/application.xml</li> </ul>
<b>Examples</b>	None

## G1079

<b>Statement</b>	<i>J2EE</i> applications should isolate tailorable data values into the deployment descriptor.
<b>Rationale</b>	Do not hard-code tailorable data into source files. The standard location for tailorable data for J2EE applications is in deployment descriptors. Developers should not reinvent the wheel of creating a non-standard mechanism for retrieving configurable data. Tailorable data is made accessible through application contexts that are provided by the application <i>container</i> (J2EE <i>application server</i> ).
<b>Derived From</b>	
<b>Justifies</b>	[G1200], [G1201]
<b>Referenced By</b>	<i>J2EE environment, Java Naming &amp; Directory Interface (JNDI)</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See the evaluation criteria for the guidance statements that this guidance statement justifies.

# G1080

**Statement** Web service environments should adhere to the **WS-I** standards for Basic Profile.

**Rationale** Most of the **COTS** web service products have already met this requirement. This is intended to cause a rejection of the non-standard web server.

The WS-I standards for Basic Profile can be found at *WSI Org Basic Profile* and at the Microsoft site, *Microsoft Basic Profile*.

**Derived From**

**Justifies**

**Referenced By** WS-I compliance guidance

**Acquisition Phase** Development

**Evaluation Criteria**

1. **Test** *Is the web service product WS-I compliant?*

**Procedure** Identify the web-service product being used, and verify through a literature search that it is WS-I compliant.

**Examples** None

## G1082

<b>Statement</b>	Use the document literal style for all data transferred using <i>SOAP</i> where the document is a W3 Organization's Document Object Model ( <i>DOM</i> ).	
<b>Rationale</b>	The document literal style requires that the input and output parameters to a web service be defined as W3 Organization Documents that follow the Document Object Model (DOM). The DOM acts as a contract between the <i>producer</i> and the <i>consumer</i> of the web service that is formal, well-defined, and rigorous. By validating the DOM against an <i>XML</i> Schema Definition ( <i>XSD</i> ), any discrepancies in the interface can be resolved.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	WS-I compliance guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b>      <i>Does the <b>WSDL</b> define input, output, or returned parameters as W3 Organization Documents that follow the Document Object Model (DOM)?</i></p> <p><b>Procedure</b>      Review all WSDL files used to describe a web service, and make sure they only pass documents. Document types should be <code>xsd:anyType</code>.</p> <p><b>Examples</b>      None</p>

## G1083

<b>Statement</b>	Do not pass <i>DOM</i> documents <i>as</i> strings.	
<b>Rationale</b>	Because of the relative simplicity of converting an <i>XML</i> document to a string, it is easy to pass an entire document as a string rather than as an XML document. This can cause problems if the document contains tags that are similar to the tags used in the <i>SOAP</i> . Passing it as an XML document ensures that the document is treated as a single entity.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	WS-I compliance guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Does the <b>WSDL</b> define input, output, or returned parameters as strings?</i></p> <p><b>Procedure</b> Review all the WSDL files used to describe a web service and make sure that they only pass documents, not strings. Document types should be <code>xsd:anyType</code>.</p> <p><b>Examples</b> None</p>

# G1084

<b>Statement</b>	Documents transferred using <b>SOAP</b> should be validated by an <b>XSD</b> defined by the <b>Community of Interest (COI)</b> .	
<b>Rationale</b>	<p>Numerous COIs are defining data that is specific to their needs. Many are capturing the data in schemas that can be used in a <b>DOM</b>.</p> <p>For example, the Joint Air and Missile Defense (JAMD) COI is working in accordance with the DoD Network Centric Data Strategy.</p> <p>The interfaces should leverage interface documents based on these efforts rather than defining their own.</p>	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	WSDL guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Does the <b>WSDL</b> reference external XSDs defined by independent COIs?</i>
	<b>Procedure</b>	Review all the WSDL files and determine if the parameters use or reference external XSDs.
	<b>Examples</b>	None

# G1085

<b>Statement</b>	Use DoD-registered namespaces to avoid name collisions and conflicts.	
<b>Rationale</b>	<p>Many organizations and groups provide services at the same time. Often, two groups provide a service using the same name; for example, weather. There are formal groups and organizations that can prevent these name collisions.</p> <p>For interim <i>namespace</i> management, use the COE prefix and segment name for all components, even if your application is not a COE segment. See <i>namespace management procedures</i> for the <i>NESI</i> Service Name procedure.</p>	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	WSDL guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b> <i>Is the COE prefix and segment name used for all components?</i></p> <p><b>Procedure</b> Parse the <i>XML</i> and look for the appropriate COE prefix and namespace.</p> <p><b>Examples</b> None</p>

# G1086

<b>Statement</b>	All published <b>WSDL</b> files should use a method of defining the Document Literal style for parameters that is interoperable across web-service vendors.
<b>Rationale</b>	There are subtle differences between the ways web-service vendors handle the document literal style. The method in which they define the Document Literal style within the WSDL can introduce incompatibilities that cause problems during ports between vendors.
<b>Derived From</b>	
<b>Justifies</b>	
<b>Referenced By</b>	WSDL guidance
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	<p><b>1. Test</b> <i>Are all the types used to pass documents into and out of the web service <code>xsd:anyType</code>?</i></p> <p><b>Procedure</b> Examine the WSDL file input parameters and return the parameters' element type to make sure they are defined as <code>xsd:anyType</code>.</p> <p><b>Examples</b> The Axis WSDL code snippet below is an example of how to resolve interoperability issues. It modifies the WSDL file schema definition section and changes the argument element type to <code>xsd:anyType</code>.</p> <pre> &lt;!-- WSDL snippet from Axis for Document Literal Style. ◇ &lt;wsdl:types&gt;   &lt;schema &lt;!-- . . . Some code removed for brevity ◇   &lt;element     name="in0"     type=" apachesoap:Document"/&gt;   &lt;element     name="getCelestialInfoReturn"     type=" apachesoap:Document"/&gt;   &lt;/schema&gt; &lt;/wsdl:types&gt; &lt;!-- WSDL snippet from Axis for Document Literal Style. ◇ &lt;wsdl:types&gt;   &lt;schema &lt;!-- . . . some code removed for brevity ◇   &lt;element     name="in0"     type="xsd:anyType"/&gt;   &lt;element </pre>

```

        name="getCelestialInfoReturn"
        type="xsd:anyType"/>
    </schema>

```

## 2. Test *Are XML documents passed as strings?*

**Procedure** Examine the code or the *SOAP* message to ensure the the document is not passed as a string.

**Examples** Passing the result of a report as a string (INCORRECT):

```

<soapenv:Body>
  <getCelestialInfoReturn
    xmlns="urn:CelestialInfoDocDoc"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:type="xsd:String">
    &lt;CelestialInfoRpt xmlns=""&gt;
      &lt;description&gt;
        DOC-DOC: Results returned from :
        Softology01 (192.168.2.4)
      &lt;/description&gt;
      &lt;moonrise&gt;2004-07-12 1:59 AM
PDT&lt;/moonrise&gt;
      &lt;moonset&gt;2004-07-12 4:22 PM
PDT&lt;/moonset&gt;
      &lt;sunrise&gt;2004-07-12 5:50 AM
PDT&lt;/sunrise&gt;
      &lt;sunset&gt;2004-07-12 7:58 PM
PDT&lt;/sunset&gt;
      &lt;/CelestialInfoRpt&gt;
    </getCelestialInfoReturn>
  </soapenv:Body>

```

Passing the result of a report as XML (CORRECT):

```

<soapenv:Body>
  <getCelestialInfoReturn
    xmlns="urn:CelestialInfoDocDoc"
    xmlns:ns1="http://xml.apache.org/xml-soap"
    xsi:type="ns1:Document">
    <CelestialInfoRpt xmlns="">
      <description>
        DOC-DOC: Results returned from :
        Softology01 (192.168.2.4)
      </description>
      <moonrise>2004-07-12 1:59 AM PDT</moonrise>
      <moonset>2004-07-12 4:22 PM PDT</moonset>
      <sunrise>2004-07-12 5:50 AM PDT</sunrise>
      <sunset>2004-07-12 7:58 PM PDT</sunset>
    </CelestialInfoRpt>
  </getCelestialInfoReturn>
</soapenv:Body>

```

# G1087

<b>Statement</b>	Validate all <b>WSDL</b> (Web Services Definition Language) files that describe <i>web services</i> .						
<b>Rationale</b>	Manually editing a WSDL file is error-prone, work-intensive, and hard to maintain. However, if the user wants to do it, there is no way to detect a manually edited file from one that was auto generated. The important thing is not how the WSDL file is generated but rather that the WSDL file is valid. It must be validated with a WSDL validator.  Note: Not all WSDL files that are generated and valid are necessarily interoperable.						
<b>Derived From</b>							
<b>Justifies</b>							
<b>Referenced By</b>	Insulation and structure guidance						
<b>Acquisition Phase</b>	Development						
<b>Evaluation Criteria</b>	<ol style="list-style-type: none"> <li> <table> <tr> <td><b>Test</b></td> <td><i>Can the WSDL file be validated?</i></td> </tr> <tr> <td><b>Procedure</b></td> <td>Obtain a WSDL validator from a source such as <a href="http://pocketsoap.com/wsdl/">http://pocketsoap.com/wsdl/</a>.</td> </tr> <tr> <td><b>Examples</b></td> <td>None</td> </tr> </table> </li> </ol>	<b>Test</b>	<i>Can the WSDL file be validated?</i>	<b>Procedure</b>	Obtain a WSDL validator from a source such as <a href="http://pocketsoap.com/wsdl/">http://pocketsoap.com/wsdl/</a> .	<b>Examples</b>	None
<b>Test</b>	<i>Can the WSDL file be validated?</i>						
<b>Procedure</b>	Obtain a WSDL validator from a source such as <a href="http://pocketsoap.com/wsdl/">http://pocketsoap.com/wsdl/</a> .						
<b>Examples</b>	None						

# G1088

<b>Statement</b>	Use isolation design patterns such as <i>façade</i> , <i>proxy</i> , or <i>adapter</i> to isolate the application from the connection and manipulation of <b>SOAP</b> messages.
<b>Rationale</b>	Insulating web-services (network)-specific code using standard abstractions such as a proxy object or an adapter will insulate the application from changes in web-service code and make the code easier to maintain, because it is centrally located.
<b>Derived From</b>	
<b>Justifies</b>	
<b>Referenced By</b>	Insulation and structure guidance
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	<p><b>1. Test</b> <i>Are web service calls inside of the application code?</i></p> <p><b>Procedure</b> Check for proliferation of web service calls inside an application.</p> <p><b>Examples</b> None</p> <p><b>2. Test</b> <i>Are web service calls isolated in a single adapter or proxy object?</i></p> <p><b>Procedure</b> Check to see if all web service calls are isolated to a single adapter or proxy object.</p> <p><b>Examples</b> None</p> <p><b>3. Test</b> <i>Are SOAP-client calls inside the application code?</i></p> <p><b>Procedure</b> Check to see if SOAP-client code is proliferated inside the application code?</p> <p><b>Examples</b> None</p>

# G1090

<b>Statement</b>	Do not hard-code a web service's <i>endpoint</i> .	
<b>Rationale</b>	<p>This causes unnecessary dependencies between the client code and the web service that it uses.</p> <p>Sometimes hard-coding may be unavoidable. For example, many tools provided by web service vendors hard-code the web service's URL in the generated client-side helper classes.</p>	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<p><i>Are there any hard-coded URLs in the client-side code?</i></p>
	<b>Procedure</b>	Parse the client code looking for hard-coded URLs.
	<b>Examples</b>	<p>The Java code samples below illustrate how this might be done. The first sample shows parameters that are hard-coded; the second sample shows how parameters and web-service endpoints are insulated.</p> <p>1. Hard-coded parameters:</p> <pre>// Sample code that has hard-coded parameters // before applying insulation public static void main ( String[] args ) throws Exception { //The SOAP endpoint String sSoapEndpoint = "http://live.capescience.com:80" + "/ccx/AirportWeather"; AirportWeatherClient myProxy = null; try { myProxy = AirportWeatherClientFactory.create ( sSoapEndpoint); System.out.println ("Location: " + myProxy.getLocation(args[0]) ); //rest of code removed for brevity } // End try Catch ( Exception exception ) { System.out.println("Error: " + exception);</pre>

```
    } // End catch
}; //end of main program
```

## 2. Insulated parameters and web-service endpoints

a. Property file - this code shows the property file itself:

```
/* Property file: property.dat
*/
targetUrl=http://198.253.106.75/
```

b. Proxy sample code

```
// Sample code that has
parameters and
// web service connection
through helper
// methods after applying
insulation
public interface
airportWeatherProxy
{ public abstract String
getLocation();
    // other public API's removed
for brevity
} // End airportWeatherProxy
```

c. Client sample code:

```
import java.io.*;
import java.rmi.*;
import java.util.*;
import AirportWeatherClient; //
auto-generated SOAP //

client from IDE */
public class WeatherProxy
    implements airportWeatherProxy
{
    //
    //code removed for brevity
    //
    public WeatherProxy
        ( String propFileStr )
    { try
        { getEndPoint(propFileStr);
        } // End try
        catch(Exception e)
        { // Handle exception here
        } // End catch
        connect2SOAP();
    } // End constructor
    /* public api's */
    public String getLocation()
    { return location;
    } // End getLocation
    . . . // Other public API's
removed for brevity
    private void getEndPoint
        ( String propsFile )
```

```

        throws Exception
        { if ( propsFile == null ||
propsFile.length() == 0 )
            { throw new Exception
                ( "SOAP EndPoint
parameter not defined");
            } // End if
            props = new Properties();
            try
            { InputStream is = new
FileInputStream(propsFile);
                props.load(is);
                is.close();
            } // End try
            catch ( Exception exception
                )
                { throw new Exception
                    ( "can't read props file
" + propsFile);
                } // End catch
                Enumeration enum =
props.propertyNames();
                while (
enum.hasMoreElements() )
                    { String endPointString =
null;
                        String propName =
enum.nextElement().toString();
                        if ( propName.equals (
endPointString ) )
                            { soapEndpoint =
props.getProperty( propName );
                                break;
                            } // end if
                        } // End while
                    } //end getEndPoint
private void connect2SOAP()
{ try
    { myProxy
        =
AirportWeatherClientFactory.crea
te
            ( soapEndpoint );
        . . . //code removed for
brevity
    } // End try
    catch ( Exception exception )
    { System.out.println
        ( "Error connecting to
SOAP server: "
            + exception
        );
    } // End catch
} // End connect2SOAP
private Properties props =
null;
private String propsFile =

```

```
    null;
    private AirportWeatherClient
myProxy = null;
    private String soapEndpoint =
null;
    private String location =
null;
} //end WeatherProxy
public class Weather
{ private static WeatherProxy
myWeatherProxy = null;
    public static void main
        ( String[] args
          ) throws Exception
        { try
          { myWeatherProxy = new
WeatherProxy ( args[0] );
          } // End try
          Catch ( Exception exception
                )
            { throw new Exception
              ( "can't connect to SOAP
server");
            } // End catch
            System.out.println
              ( "Location: "
                +
myWeatherProxy.getLocation()
              );
            . . . //code deleted for
brevity
          } //end main
        } //end Weather
```

# G1091

<b>Statement</b>	Do not hard code web-service-vendor specifics.
<b>Rationale</b>	Some web-service vendors add dependencies to their products and services, which can reduce <i>portability</i> and increase the cost of porting to other web-service vendors.
<b>Derived From</b>	
<b>Justifies</b>	[G1236], [G1237]
<b>Referenced By</b>	Insulation and structure guidance
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel guidance to evaluate this guidance.

# G1093

<b>Statement</b>	Web services must handle <i>SOAP</i> exceptions and SOAP faults.	
<b>Rationale</b>	SOAP exceptions are raised when there are connective problems or violations in the SOAP protocol between the client and the server.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	Error guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Does the web application client have exception handlers for SOAPExceptions?</i></p> <p><b>Procedure</b> Check to see that the web application client has an exception block specifically for SOAPException.</p> <p><b>Examples</b> None</p>
	<b>2.</b>	<p><b>Test</b> <i>Does the web application client test the SOAP response for a fault?</i></p> <p><b>Procedure</b> Verify the web application client handles a true value returned from the <code>response.generatedFault</code> method.</p> <p><b>Examples</b> None</p>

# G1094

<b>Statement</b>	Application code exposed as a web service should catch all exceptions.	
<b>Rationale</b>	Any exception can reveal system internals and thus compromise security. Also, internal exceptions are not user friendly.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	Error guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Does each exposed web method catch all possible exceptions and re-throw a declared application exception?</i></p> <p><b>Procedure</b> Verify that each exposed web method has an exception block that catches all possible exceptions and then re-throws them as a declared application exceptions.</p> <p><b>Examples</b> None</p>
	<b>2.</b>	<p><b>Test</b> <i>Does each exposed web method catch all possible runtime exceptions and re-throw a declared application runtime exception?</i></p> <p><b>Procedure</b> Verify that each exposed web method has an exception block that catches all possible exceptions and then re-throws them as a declared application exceptions.</p> <p><b>Examples</b> None</p>

# G1095

<b>Statement</b>	Use <b>W3C</b> fault codes for all <b>SOAP</b> faults.	
<b>Rationale</b>	Having predefined and accepted fault codes allows consumers to handle SOAP faults appropriately without prior knowledge of custom fault codes.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	Error guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Does the web application throw fault codes from the accepted list of fault codes?</i></p> <p><b>Procedure</b> Verify that each fault code thrown by the web application is from the accepted list of SOAP fault codes defined by the W3C.</p> <p><b>Examples</b> None</p>

# G1101

<b>Statement</b>	Use web services to bridge <i>J2EE</i> and <i>.NET</i> .	
<b>Rationale</b>	<p>The easiest and best way to bridge J2EE and .NET is to define a web service.</p> <p>There are other ways to bridge J2EE and .NET using <i>COTS</i> products. If used, these should follow the <i>ANSI</i> Abstract Syntax Notation One (ASN.1) standard <a href="http://asn1.elibel.tm.fr/en/standards/index.htm#asn1">http://asn1.elibel.tm.fr/en/standards/index.htm#asn1</a>.</p> <p>ASN.1 is a formal notation for describing data transmitted by telecommunications protocols. It applies regardless of language implementation, physical representation of this data, application, and degree of complexity. (<a href="http://asn1.elibel.tm.fr/en/introduction/index.htm">http://asn1.elibel.tm.fr/en/introduction/index.htm</a>).</p>	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	.NET Framework	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b>      <i>Are Java and .NET files in the project?</i></p> <p><b>Procedure</b>      Look for files with the .java, .class, .obj, .cs, .cc, or .c extensions existing with the source code.</p> <p><b>Examples</b>      None</p>

# G1117

<b>Statement</b>	Isolate topic and queue names by not hard-coding them in client code.	
<b>Rationale</b>	Since topics and queues are vendor-specific, maintain portability by isolating the hard-coded topics and queues from the rest of the application. To do this, use helper classes or property files.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	Messaging	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Does the client code use hard-coded topics and queues in unisolated places in the application?</i></p> <p><b>Procedure</b> Verify that all occurrences of hard-coded topics and queues are in isolated locations within the source code.</p> <p><b>Examples</b> None</p>

# G1118

**Statement** Localize *CORBA*-vendor-specific source code into separate modules.

**Rationale** The general guidance is to minimize CORBA vendor-specific source code, while recognizing that vendor-specific features are necessary in certain circumstances. However, isolating vendor-specific code reduces maintenance effort.

Vendor capabilities tend to change more rapidly than CORBA-standard specifications. Experience shows that vendor updates frequently require modification to application source code, due to changing vendor interface conventions. These modifications impose vendor-version-specific constraints on the application, thereby complicating maintenance.

## Example

### Encapsulating CORBA ORB operations

The following examples show how to encapsulate binding operations for a C++ *ORB*, and naming service operations for a Java ORB.

#### C++ ORB binder template

The code below shows a sample template for binding to the C++ ORB. IONA's ORBIX was used in this example.

```
/* =====
ServerBinder.h (Template)
this is a generic binder to ORBIX
===== */
#ifndef _BINDER_H_
#define _BINDER_H_
#ifndef IOSTREAM_H
#define IOSTREAM_H
#include <iostream.h>
#endif
#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif
template <class SERVERNAME, class VARPTR>
class Binder
{ private:
    char* serverName;
public:
    Binder(char* svName):serverName(svName){};
    ~Binder(){};
    int bind( VARPTR* p)
    { int attempts = 0, success = 0;
      int maxtries = 5, retval = 0;
      while ( ( attempts < maxtries )
              && (!success)
            )
    }
```

```

    { ++attempts;
      cout << "Binding to server, attempt "
            << attempts
            << endl;
      try
      { (*p) = SERVERNAME::_bind();
        cout << "Bound to server"
              << endl;
        success = retval = 1;
      } // End try
      catch ( CORBA::SystemException &systemException )
      { cout << "SystemException, ServerBinder::bind"
            << endl
            << systemException;
        success = 1;
        retval = 0;
      } // End catch SystemException
      catch (...)
      { cout << "unknown Exception,
ServerBinder::bind"
          << endl;
        success = 1;
        retval = 0;
      } // End catch all
    } //end while
    return retval;
  } //end bind
} //end Binder
#endif

```

### Ada ORB binder template for C++

The code below shows a C++ template for binding to an Ada ORB. ORBExpress was used in this example.

```

/* =====
ada_binder.h (Template)
this is a generic binder to ORBExpress
===== */
#ifndef _ADA_BINDER_H_
#define _ADA_BINDER_H_
#ifndef IOSTREAM_H
#define IOSTREAM_H
#include <iostream.h>
#endif
#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif
template <class SERVERNAME, class VARPTR >
class Ada_Binder
{ private:
  char* adaIorString;
public:
  Ada_Binder
    ( char* iorString)
  : adaIorString ( iorString )

```

```

};
~Ada_Binder(){};
int bindToAda( VARPTR* p)
{ int attempts = 0, success = 0;
  int maxtries = 5, retval = 0;
  while ( ( attempts < maxtries)
          && (!success)
        )
  { ++attempts;
    cout << "Binding to server, attempt "
          << attempts
          << endl;
    try
    { cout <<"adaIorString:"
      << endl
      << adaIorString
      << endl;
      (*p) = SERVERNAME::_bind(adaIorString);
//can't use string_to_object in this version
//it kills the ada IOR
//      CORBA::Object_ptr myptr
//      CORBA::Orbix.string_to_object
//      ( adaIorString );
//      (*p) = SERVERNAME::_narrow(myptr);
      cout << "Bound to server" << endl;
      success = retval = 1;
    } // End try
    catch (CORBA::SystemException& systemException)
    { cout << "SystemException, "
      << "AdaServerBinder::bind"
      << endl
      << systemException;
      success = 1;
      retval = 0;
    } // End SystemException
    catch (...)
    { cout << "Unknown Exception, "
      << "AdaServerBinder::bind"
      << endl;
      success = 1;
      retval = 0;
    } // End catch all
  } // end while
  return retval;
} // end bind
} // end ADA_Binder
#endif

```

### **Example: Naming service operations for a Java ORB**

#### **Java helper class**

This example is a helper class, `JavaNamingHelper.java`, that encapsulates CORBA naming service operations for all services to use. We used Java **JDK** 1.4 ORB to create this example.

```
import java.util.*;
```

```

import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.CORBA_2_3.ORB.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContext.*;
import org.omg.CosNaming.NamingContextPackage.*;
import CBRNSensors.JSLSCAD.*;
public class JavaNamingHelper
{ static NamingContext nameSvc = null;
  static org.omg.CORBA.Object objref = null;
  static JSLSCADSensor myCBRNSensor = null;
  static org.omg.CORBA.Object myobj = null;
  public JavaNamingHelper()
  {
  }
  private static void showNamingContext
    ( org.omg.CORBA.ORB myorb )
  {
  public static NamingContext getNamingSvc
    ( org.omg.CORBA.ORB lclorb,
      String nameSvcName
    )
  { NamingContext lclNameSvc = null;
    try
    { org.omg.CORBA.Object nameSvcObj
      = lclorb.resolve_initial_references
        ( "NameService" );
      // . . . other business logic removed
      //           for brevity
    } // End try
    catch(org.omg.CORBA.COMM_FAILURE cf)
    { . . . // error code goes here
    } // End catch
    catch ( org.omg.CORBA.ORBPackage.InvalidName
invalidName)
    { . . . // error code goes here
    } // End catch
    catch ( SystemException systemException )
    { . . . // error code goes here
    }
  } // End getNamingSvc
  public static org.omg.CORBA.Object getObjFromNameSvc
    ( org.omg.CORBA.ORB myorb,
      String targetSensorName
    )
  { . . . // business logic goes here
  } //end getObjFromNameSvc
  public static int setObj2NameSvc
    ( org.omg.CORBA.ORB myorb,
      BasesSensor mySensor,
      String targetSensorName
    )
  { . . . // business logic goes here
  } //end setObj2NameSvc
}; //end class JavaNamingHelper

```

**Java server implementation**

The code below is a sample Java server implementation that uses the naming service helper class.

```
import java.io.*;
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.CORBA_2_3.ORB.*;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContext.*;
import org.omg.CosNaming.NamingContextPackage.*;
class MyServer
{ public static Properties props;
  public static ORB myorb = null;
  public static NamingContext nameSvc = null;
  public static RootSensor mySensor = null;
  public static String propertyFilePath = null;
  public static final String MY_SENSOR_NAME = "MYSENSOR";
  static public void main(String[] args)
  { // handle arguments
    System.out.println(" CORBA Server starting...\n");
    try
    { // Initialize the ORB.
      myorb = ORB.init(args, props);
      //instantiate servant and create ref
      POA rootPOA
      =
      POAHelper.narrow(myorb.resolve_initial_references
        ( "RootPOA" );
      . . . // rest of initialization code goes here
    } // End try
    catch ( org.omg.CORBA.ORBPackage.InvalidName
      invalidName )
    { . . . //error code goes here
    } // End invalidName
    // other exception types to catch go here
    catch ( SystemException systemException)
    { System.err.println ( systemException );
    } // End systemException
    // naming service hookup
    JavaNamingHelper.setObj2NameSvc
      ( myorb,mySensor,
        MY_SENSOR_NAME
      );
    try
    { System.out.println(" Ready to service requests\n");
      myorb.run();
    } // End try
    catch(SystemException systemException)
    { System.err.println ( systemException );
    } // End catch systemException
  } // End static block
} // End MyServer
```

### ***Java client implementation***

The code below is a sample client implementation that uses the naming

service helper class.

```

import java.io.*;
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContext.*;
import org.omg.CosNaming.NamingContextPackage.*;
import CBRNSensors.*;
import CBRNSensors.JSLSCAD.*;
import CBRNSensors.JSLSCAD.Impl.*;
public class JSLSCADClient
{ public static Properties props;
  public static ORB myorb = null;
  public static String mySensorStr = null;
  private static org.omg.CORBA.Object objref = null;
  // helper class to handle orb connections etc.
  private static void connectToOrb
    ( String args[] )
  { try
    { myorb = ORB.init(args,props);
    } // End try
    catch(SystemException systemException)
    { System.err.println
      ( systemException.toString() );
      return;
    } // End catch systemException
    System.out.println("get naming service\n");
    objref
      = JavaNamingHelper.getObjFromNameSvc
        ( myorb,
          mySensorStr
        );
    sensorObj
      = JSLSCADSensorHelper.narrow(objref);
    try
    { POA rootPOA
      =
    POAHelper.narrow(myorb.resolve_initial_references
      ( "RootPOA" );
      rootPOA.the_POAManager().activate();
    } // End try
    catch(org.omg.CORBA.ORBPackage.InvalidName
invalidName)
    { //error code here
    } // End catch InvalidName
    . . . // other exceptions that may be required
      // for the operations
    catch(SystemException systemException)
    { System.err.println
      ( "System Exception during ops");
      System.err.println
        ( systemException );
    } // End systemException
  } // End connectToOrb

```

```

//helper method to handle orb specific issues
private static void disconnectFromOrb()
{ . . . // business logic goes here
} // End disconnectFromOrb
public static void main
( String args[] )
{ // Initialize the ORB.
System.out.println ( "Initializing the ORB\n" );
props = new Properties();
// load property values
// use helper methods
connectToOrb ( args );
try
{ . . . // client business logic goes here
} // End try
catch ( Exception exception )
{ . . . // Exception handling code goes here
} // End exception handler
disconnectFromOrb( );
} // end main
} // end client

```

**Derived From** [G1008]

**Justifies** [G1202]

**Referenced  
By**

**Acquisition  
Phase** Development

**Evaluation  
Criteria** The following evaluation criteria relate to non-IDL compiler auto-generated code. Further, the criteria relate to modules which are not annotated to contain vendor-specific code.

1.    **Test**            *Does the module contain vendor names anywhere in code text?*
  - Procedure**    Review the code looking for a service that can be used to obtain configuration.
  - Examples**     None
2.    **Test**            *Are any non-CORBA compliant CORBA:: objects declared or defined in the module?*
  - Procedure**    Review the code for a service that can be used to obtain configuration.
  - Examples**     None

# G1119

<b>Statement</b>	Isolate user-modifiable configuration parameters from the CORBA application source code.
<b>Rationale</b>	<p>Configuration parameters control the behavior of the CORBA <b>ORB</b> service environment and client/service processes during startup, execution, and termination. This parameterization allows execution-time control modification without having to rebuild, reinstall, or redeploy.</p> <p>Configuration defines the state of the client-and-service environment throughout the lifetime of the processes involved. This relates to considerations such as the allocation of threading and resources, <b>POA</b> policies, the instantiation of servants and their invocations, failure and security behavior, connection management, quality of service prioritization, and so forth. The point is that CORBA provides an extremely complex but flexible environment for distributed computing interaction. Consequently, the designer requires flexible guidance to handle this option-rich environment.</p> <p>Configuration processes and their related parameters fall into two categories. The first involves configuration matters, which are defined to be perpetually static by the system architecture. The second involves matters that are intended to be modifiable by users.</p> <p>The first category, immutable configuration settings, relates to fundamental underlying assumptions that are foundational for the implementation. These are matters for which no user modification is ever intended as it would lead to unspecified behavior. Consider the example of a service implementation that is programmed to be single threaded. In this case, multi-threading controls are irrelevant and multiple instantiation would lead to dangerous confusion. For immutable configuration parameters, localized and well-commented implementation in the application source code is appropriate.</p> <p>For user-modifiable configuration settings, there are two further by-design divisions. The first involves configuration settings that are intended to be accessible by distributed processes. The second involves host-specific settings which relate to resources locally available, for which remote access is not desired. These are discussed in the related sublevel guidance</p>
<b>Derived From</b>	
<b>Justifies</b>	[G1204], [G1205]
<b>Referenced By</b>	CORBA
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel guidance to evaluate this guidance.

# G1121

<b>Statement</b>	Do not modify CORBA <i>IDL</i> compiler auto-generated stubs and skeletons.
<b>Rationale</b>	<p>The purpose of the IDL auto-generated stub and skeleton files is to provide a source code facility/mechanism for the developer in a specific language to use the IDL-described object interface in that specific language. The internal content of these files changes with the application's IDL modification, with IDL compiler-environment configuration settings, and with vendor-product compiler and <i>ORB</i> upgrades. By design, these files are not intended to be modified by the application developer. Developer modification of any auto-generated stub or skeleton file will typically lead to very severe maintenance hazards and failed application rebuild results.</p> <p>The stub files describe the language source-code interface from the client side. Their use involves including the client stub header in the application's call invocation code.</p> <p>The skeleton files describe the language source code interface from the service implementation side. Their use involves including the skeleton header in the application's operator implementation code. Their use also requires developer modification of a renamed clone of the auto-generated skeleton body file. These techniques are described in every ORB vendor's programming reference manuals.</p>
<b>Derived From</b>	
<b>Justifies</b>	
<b>Referenced By</b>	CORBA
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	<p><b>1. Test</b> <i>Is any application code contained in the auto-generated code?</i></p> <p><b>Procedure</b> Inspect the auto-generated file creation/modification dates to verify that no tampering occurred after the IDL compilation step in the build process.</p> <p><b>Examples</b> The following examples are all based upon a single CORBA IDL interface.</p>

## MyIdlInterface.idl

```
interface MyIdlInterface
{
    readonly attribute string version;
    void stop();
    void start();
    string error();
}; // End MyIdlInterface
```

## ORBExpress compiler

The ORBExpress IDL compiler generates these files:

- **myIdlInterface.h** - Client-side stub header
- **myIdlInterface.cxx** - Client-side stub implementation
- **MyIdlInterface\_s.h** - Abstract servant header
- **MyIdlInterface\_s.cxx** - Abstract servant implementation
- **MyIdlInterface\_impl.h** - Server implementation header
- **MyIdlInterface\_impl.cxx** - Server implementation implementation

**Note:** The only files that should be edited are **MyIdlInterface\_impl.h** and **MyIdlInterface\_impl.cxx**. The IDL compiler checks for the existence of the implementation (i.e. **\_impl**) files and will not overwrite them.

### **MyIdlInterface\_impl.cxx**

```
// Generated for interface MyIdlInterface
// in myIdlInterface.idl
#include "MyIdlInterface_impl.h"
MyIdlInterface_impl::MyIdlInterface_impl
( PortableServer::POA* oe_poa,
  const char* oe_object_id
) : POA_MyIdlInterface
    ( oe_object_id,
      oe_poa
    )
{ . . . // TO DO: add implementation code here
} // emd constructor
MyIdlInterface_impl::MyIdlInterface_impl
( const MyIdlInterface_impl& obj )
: POA_MyIdlInterface(obj)
{ . . . // TO DO: add implementation code here
} // End constructor
MyIdlInterface_impl::~MyIdlInterface_impl()
{ . . . // TO DO: add implementation code here
} // End destructor
CORBA::Char* MyIdlInterface_impl::version
( CORBA::Environment& _env )
{ return CORBA::string_dup(_version);
} // End version
void MyIdlInterface_impl::stop
( CORBA::Environment& _env )
{ . . . // TO DO: add implementation code here
} // End stop
void MyIdlInterface_impl::start
( CORBA::Environment& _env )
{ . . . // TO DO: add implementation code here
} // End start
CORBA::Char* MyIdlInterface_impl::error
( CORBA::Environment& _env )
```

```

{ CORBA::Char* result;
  . . . // TO DO: add implementation code here
  return result;
} // End error

```

## Java JDK compiler

The Java JDK IDL compiler generates these files:

- **MyIdlInterface.java**
- **MyIdlInterfaceHelper.java**
- **MyIdlInterfaceHolder.java**
- **MyIdlInterfaceOperations.java**
- **MyIdlInterfacePOA.java**
- **\_MyIdlInterfaceStub.java**

**Note:** Do not edit any of these files. Place the server implementation code in a file that extends from `MyIdlInterfacePOA.java`. This isolates the ORB implementation and prevents subsequent IDL compilations from accidentally overwriting the files. The code for the auto-generated `MyIdlInterfacePOA.java` class and the implementation class appears below:

### ***MyIdlInterfacePOA.java***

```

/**
 * MyIdlInterfacePOA.java .
 * Generated by the IDL-to-Java compiler
 * (portable), version "3.1"
 * from myIdlInterface.idl
 */
public abstract class MyIdlInterfacePOA
    extends org.omg.PortableServer.Servant
    implements MyIdlInterfaceOperations,
               org.omg.CORBA.portable.InvokeHandler
{ . . . // rest of the auto-generated code
  removed for brevity
} //End MyIdlInterfacePOA

```

### ***MyIdlInterfaceImpl.java***

```

package myIdlImpl;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.CORBA_2_3.ORB.*;
import org.omg.PortableServer.*;
public class MyIdlInterfaceImpl
    extends MyIdlInterfacePOA
{
    private String strVersion;
    private String errString;
    public String version ()
    { . . . // implementation code goes here

```

```
        return strVersion;
    } // End version
    public void stop ()
    { . . . // implementation code goes here
    } // End stop
    public void start ()
    { . . . // implementation code goes here
    } // End start
    public String error ()
    { . . . // implementation code goes here
      return errString;
    } // End error
} // End MyIdlInterfaceImpl
```

# G1123

## Statement

Use the “Fat Operation Technique” in *IDL* operator invocation.

## Rationale

This reduces the CORBA messaging overhead. The performance cost of network CORBA messaging is determined by two factors: latency and marshaling rate. Call latency is the minimum cost of sending any message at all. The marshaling rate is determined by the sizes of sending and receiving parameters and of return values.

In the situation of a large number of objects involving objects that hold a small amount of stat, the call latency cost far exceeds the marshalling costs. Taking advantage of this reality, the “Fat Operation Technique” involves constructing structure objects which hold an aggregation of related attributes, and using the resulting structures in operation invocation parameters and returns. This amounts to transferring a larger amount of information with each network transaction.

For more information, see *Advanced CORBA Programming with C++* by Henning & Vinoski, 1999 Addison Wesley, Chapter 22.

## Derived From

## Justifies

## Referenced By

CORBA

## Acquisition Phase

Development

## Evaluation Criteria

- |    |                  |   |
|----|------------------|---|
| 1. | <b>Test</b>      | <i>Does the IDL contain function calls which have structure objects that are passed as parameters or returned from operators?</i>   |
|    | <b>Procedure</b> | Inspect the IDL file and manually check for parameters or returns using objects defined as structures, and verify that they are passed from methods also declared in the IDL. |
|    | <b>Examples</b>  | None  |

# G1126

<b>Statement</b>	Validate all <i>WSDL</i> files.		
<b>Rationale</b>	WSDL files can be difficult to produce, and there are different versions of the WSDL specification available. In order to ensure interoperability, the WSDL files need to be validated.		
<b>Derived From</b>			
<b>Justifies</b>			
<b>Referenced By</b>	WSDL guidance		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Are all WSDL files valid?</i>
		<b>Procedure</b>	Download a validation tool and test WSDL files. Tool at ws-i.org: <i><a href="http://www.ws-i.org/deliverables/workinggroup.aspx?wg=testingtools">http://www.ws-i.org/deliverables/workinggroup.aspx?wg=testingtools</a></i> Tool at eclipse.org: <i><a href="http://dev.eclipse.org/viewcvs/indextech.cgi/wsvt-home/main.html?rev=1.20">http://dev.eclipse.org/viewcvs/indextech.cgi/wsvt-home/main.html?rev=1.20</a></i> Tool at xMethods.net: <i><a href="http://xmethods.net/ve2/Tools.po">http://xmethods.net/ve2/Tools.po</a></i>
		<b>Examples</b>	Place any examples here, preceded by a brief description.

# G1127

<b>Statement</b>	Use <i>OASIS UDDI</i> specification 2.0 or higher.	
<b>Rationale</b>	UDDI provides a registration for services, and UDDI 2.0 has become a standard method for publishing discovery services.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	UDDI guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Are the web services registered in a UDDI registry?</i></p> <p><b>Procedure</b> Verify the registration in the UDDI registry.</p> <p><b>Examples</b> None</p>
	<b>2.</b>	<p><b>Test</b> <i>Is the registry UDDI 2.0 or higher?</i></p> <p><b>Procedure</b> Determine if the particular UDDI registry is UDDI Version 2.0 or higher.</p> <p><b>Examples</b> None</p>

# G1131

<b>Statement</b>	All <b>UDDI</b> inquiries should use the standard UDDI APIs.	
<b>Rationale</b>	There is a standard <b>API</b> that uses <b>SOAP</b> messages to communicate with the UDDI registry. To increase compatibility and portability, use this API exclusively.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	UDDI guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Are all the interfaces to the UDDI registry made using the UDDI standard API?</i>
	<b>Procedure</b>	<p>The standard API for UDDI is SOAP based. Requests and responses are passed using <b>XML</b> documents. Test the traffic flow between the client and the UDDI registry for messages that are defined in the UDDI specification. Use standard libraries to send and receive the messages (e.g. JUDDI for Java).</p> <p>Checking for the use of packages like JUDDI does not require the application to be running.</p>
	<b>Examples</b>	<p>The following is an example as provided in the UDDI API reference:  <a href="http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm#_Toc25137712">http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm#_Toc25137712</a>.</p>
		<p><b>find_binding</b></p> <p>The <code>find_binding</code> API call returns a <code>bindingDetail</code> message that contains zero or more <code>bindingTemplate</code> structures matching the criteria specified in the argument list.</p>
		<p><b>Syntax</b></p> <pre>&lt;find_binding   serviceKey="uuid_key"   [maxRows="nn"] generic="2.0"   xmlns="urn:uddi-org:api_v2" &gt;   [&lt;findQualifiers/&gt;]   &lt;tModelBag/&gt; &lt;/find_binding&gt;</pre>
		<p><b>Arguments</b></p> <p><b>serviceKey:</b> This <code>uuid_key</code> is used to specify a</p>

particular instance of a `businessService` element in the registered data. Only bindings in the specific `businessService` data identified by the `serviceKey` passed will be searched.

**maxRows:** This optional integer value allows the requesting program to limit the number of results returned.

**findQualifiers:** This optional collection of `findQualifier` elements can be used to alter the default behavior of search functionality. See the `findQualifiers` appendix for more information.

**tModelBag:** This is a list of `tModel uuid_key` values that represents the technical fingerprint of a `bindingTemplate` structure contained within the `businessService` specified by the `serviceKey` value. Only `bindingTemplates` that contain all of the `tModel` keys specified will be returned (logical AND). The order of the keys in the `tModel` bag is not relevant.

### Returns

This API call returns a `bindingDetail` message upon success. In the event that no matches were located for the specified criteria, the `bindingDetail` structure returned will be empty (i.e., it contains no `bindingTemplate` data.) This signifies a zero match result. If no arguments are passed, a zero-match result set will be returned.

In the event of an overly large number of matches (as determined by each Operator Site), or if the number of matches exceeds the value of the `maxRows` attribute, the Operator site will truncate the result set. If this occurs, the response message will contain the truncated attribute with the value "true".

### Caveats

If any error occurs in processing this API call, a `dispositionReport` element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

**E\_invalidKeyPassed:** signifies that the `uuid_key` value passed did not match with any known `serviceKey` or `tModelKey` values. The error structure will signify which condition occurred first, and the invalid key will be indicated clearly in text.

**E\_unsupported:** signifies that one of the

`findQualifier` values passed was invalid. The invalid qualifier will be indicated clearly in text.

# G1132

<b>Statement</b>	Implement the data tier using readily available <i>COTS RDBMS</i> products that implement the <i>SQL</i> standard and provide a rich set of generic capabilities such as row-level locking, <i>stored procedures</i> , <i>triggers</i> , and a high-level language <i>API</i> interface.	
<b>Rationale</b>	COTS RDBMSs are mature technical products, the capabilities of which are being continually expanded to adapt to and accommodate new technologies. Moreover, there is a large technical community able to develop and maintain data systems based on these products. It is likely that a COTS DBMS will provide all of the data tier capabilities required by the developer.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	<i>Implementations</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b> <i>Is the proposed COTS DBMS product a readily available and supportable COTS product that implements the SQL standard?</i></p> <p><b>Procedure</b> Verify that the COTS DBMS product is widely in use in the DoD environment (e.g., Oracle, SqlServer, or DB2), has a large support community, and is likely to be supported for the lifecycle of the project.</p> <p><b>Examples</b> None</p>

# G1141

**Statement** Use standard *data models* developed by *Communities of Interest (COI)* as *the basis of program or project data models*.

**Rationale** Standard data models are under development in many areas of the DoD and will be stored in and made available from DoD metadata repositories. The use of these models or portions thereof supports interoperability among applications. The *C2IEDM* data model, which is used in the *Command and Control* area, is an example of one of these standard data model development efforts.

**Derived From**

**Justifies**

**Referenced By** *Data modeling*

**Acquisition Phase** Development

<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Have standard data models been considered for use in the system?</i>
		<b>Procedure</b>	Determine whether standard DoD data models exist for the technical areas accommodated in the system requirements. Verify that the data model developed for the application accommodates the use of these data models.
		<b>Examples</b>	None
	<b>2.</b>	<b>Test</b>	<i>If the system is a command-and-control application, has preference been given to the use of the Command &amp; Control Information Exchange Data Model (C2IEDM) rather than locally defined values?</i>
		<b>Procedure</b>	Examine the system data model and verify that the C2IEDM data model has been incorporated.
		<b>Examples</b>	None

# G1144

<b>Statement</b>	Develop a two-level database model. One level captures the conceptual or logical aspects, and the other level captures the physical aspects.	
<b>Rationale</b>	There are a number of modeling tools available that permit the development of Entity-Relationship diagrams. Developers can use these tools to create conceptual models that are independent of the <i>DBMS</i> in which the system is implemented, and to develop the physical models that are translated directly into DDL (data definition language), the <i>SQL</i> code used to create the database. Using a conceptual model permits implementation or reuse of a complex ERD on multiple DBMS products.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	<i>Data modeling</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b>      <i>Has a separate conceptual or logical model been developed?</i></p> <p><b>Procedure</b>      Verify the presence of a conceptual or logical model.</p> <p><b>Examples</b>      None</p>

# G1146

**Statement** The conceptual/logical data model should contain information necessary to generate a data dictionary.

**Rationale** A data dictionary is an integral part of every database system. A description of each data item and the units in which the contents are measured are essential. Database modeling tools provide a mechanism for storing information necessary to produce a data dictionary.

**Derived From**

**Justifies**

**Referenced By** *Data modeling*

**Acquisition Phase** Development

**Evaluation Criteria** 1. **Test** *Has description information been included in the data model?*

**Procedure** Examine the physical data model.

**Examples** None

# G1147

## Statement

*Domain analysis* should define the input-data validation constraints.

## Rationale

Domain analysis is an integral part of any database system. Domains describe the set or range of values that are acceptable for a specific data item. These include, at a minimum:

- Data type
- Precision
- Minimum
- Maximum
- Length

These values are validated in the database via check constraints on the data item. These check constraints are generated from the *physical data model* as part of the DDL.

## Derived From

## Justifies

## Referenced By

*Data modeling*

## Acquisition Phase

Development

## Evaluation Criteria

- |    |                  |   |
|----|------------------|---|
| 1. | <b>Test</b>      | <i>Has domain analysis been included in the data model?</i> |
|    | <b>Procedure</b> | Examine the physical data model.                            |
|    | <b>Examples</b>  | None  |

# G1148

<b>Statement</b>	Normalize the conceptual/logical model.		
<b>Rationale</b>	<i>Normalization</i> is a central <i>tenet</i> of relational database theory. A database should usually be normalized to at least third normal form. Although there are seven normal forms, normalization beyond third normal form is rarely considered in practical database design.		
<b>Derived From</b>			
<b>Justifies</b>			
<b>Referenced By</b>	<i>Data modeling</i>		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Is the database design in third normal form?</i>
		<b>Procedure</b>	Examine the conceptual/logical data model.
		<b>Examples</b>	None

# G1151

<b>Statement</b>	Define declarative foreign keys for all relationships between tables to enforce <i>referential integrity</i> .	
<b>Rationale</b>	<i>Foreign key</i> constraints enforce referential integrity. The principle of referential integrity requires that the foreign key values of a child table are either null or match exactly those of the <i>primary key</i> in the parent table.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	<i>RDBMS internals</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b>      <i>Have foreign-key constraints been incorporated into the database?</i></p> <p><b>Procedure</b>      Examine the database to determine whether foreign-key constraints have been included in the database creation scripts and created in the database.</p> <p><b>Examples</b>      None</p>

# G1154

<b>Statement</b>	Use <i>stored procedures</i> for operations that are focused on the insertion and maintenance of data.
<b>Rationale</b>	Current software design methodologies and architectures call for the implementation of an n-tiered architecture with business rules in the middle tier and data stored in a separate data tier. When multiple applications access a common database, however, the rules may be best located at the data-tier level. Otherwise, changes in one application would have to be coordinated across all applications. Thus their use to implement detailed <i>business logic</i> and algorithms should be limited to enterprise databases used by multiple applications.
<b>Derived From</b>	
<b>Justifies</b>	
<b>Referenced By</b>	<i>RDBMS internals</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	<p><b>1. Test</b> <i>Are database triggers used?</i></p> <p><b>Procedure</b> Check for stored procedures that are triggered on insertion, deletion, and update events.</p> <p><b>Examples</b></p> <pre>CREATE TRIGGER PersonCheckAge AFTER INSERT OR UPDATE OF age ON Person FOR EACH ROW BEGIN     IF (:new.age &lt; 0) THEN         RAISE_APPLICATION_ERROR             ( -20000,               'no negative age allowed'             );     END IF; END;</pre>

# G1155

<b>Statement</b>	Use <i>triggers</i> to enforce <i>referential or data integrity</i> , not to perform complex <i>business logic</i> .	
<b>Rationale</b>	Triggers are fired on events. Current software design methodologies and architectures call for the implementation of an n-tiered architecture with business rules in the middle tier and data stored in a separate data tier. Implementing business logic in triggers, as well as in the middle tier, violates this concept.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	<i>RDBMS internals</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Has business logic been incorporated into database triggers?</i>
	<b>Procedure</b>	Examine the database trigger code to determine whether business logic or calls to stored procedures incorporating business logic have been coded into them.
	<b>Examples</b>	None

## G1190

<b>Statement</b>	Use a build tool.
<b>Rationale</b>	A build tool allows for the encapsulation of building instructions into machine-readable files or sets of files. The instructions can be successfully and consistently repeated.
<b>Derived From</b>	
<b>Justifies</b>	[G1218], [G1219], [G1220], [G1221], [G1222], [G1223], [G1224], [G1225]
<b>Referenced By</b>	<i>Automate the build process</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel guidance to evaluate this guidance.

# G1200

<b>Statement</b>	Define all external resources by using a separate resource-ref element for each resource.	
<b>Rationale</b>	This allows the source code to look up a resource by a "virtual" name that is mapped to the actual <i>JNDI</i> location at deployment time.	
<b>Derived From</b>	[G1079]	
<b>Justifies</b>		
<b>Referenced By</b>	<i>J2EE environment, Java Naming &amp; Directory Interface (JNDI)</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b>      <i>Are there any resource references that are defined in the application code?</i></p> <p><b>Procedure</b>      Check the code for connect operations that do not use a JNDI lookup.</p> <p><b>Examples</b>      None</p>

# G1201

<b>Statement</b>	Define configuration data such as environment variables, parameters, and properties by using <code>resource-env-ref</code> elements.	
<b>Rationale</b>	Configuration data is basically a name-value pair. This allows the tailoring of the application to different contexts without having to modify source code and consequently rebuild and retest.	
<b>Derived From</b>	[G1079]	
<b>Justifies</b>		
<b>Referenced By</b>	<i>J2EE environment, Java Naming &amp; Directory Interface (JNDI)</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Are there any environment variables that must be defined before the application can be run?</i></p> <p><b>Procedure</b> Check OS startup scripts (e.g., <code>bat</code>, <code>cmd</code>, <code>csh</code>, <code>bsh</code>) for the use of any environment variables.</p> <p>Check the OS environment for any installation-defined environment variables.</p> <p><b>Examples</b> None</p>
	<b>2.</b>	<p><b>Test</b> <i>Are there any property files that need to be defined before the application can be run?</i></p> <p><b>Procedure</b> Check for the existence of properties files.</p> <p><b>Examples</b> None</p>
	<b>3.</b>	<p><b>Test</b> <i>Are there any parameters that must be defined before the application can be run?</i></p> <p><b>Procedure</b> Check for any startup parameters provided on the startup command line.</p> <p><b>Examples</b> None</p>

# G1202

**Statement** Use the CORBA Portable Object Adapter instead of the Basic Object Adapter.

**Rationale** The CORBA Basic Object Adapter (BOA) was the CORBA Version 1 specification for the client-server object capability. The BOA specification was found to be so incomplete that vendor-specific interpretations were required for operable implementation. In CORBA Version 2, the Portable Object Adapter (POA) was significantly more complete and flexible. In the current marketplace, POA implementations are standard and, in quality implementations, are not vendor-specific. Consequently, using POA eliminates one significant area of vendor-specific coding.

BOA	POA
<p>Focuses on CORBA server implementations and not CORBA object implementations</p> <p>Naming convention issues on server side</p> <p>Tightly coupled to <b>ORB</b> implementation</p> <p>Non-standardized way to connect to ORB</p> <p>Four activation models for server processes</p>	<p>Services for lifecycle management</p> <p>Abstract layer between ORB and object</p> <p>Standard, portable interface for communicating with ORB runtime</p> <p>Two servant incarnation styles</p>

**Derived From** [G1118]

**Justifies**

**Referenced By** CORBA

**Acquisition Phase** Development

**Evaluation Criteria** 1. **Test** *Does any CORBA application code reference the CORBA : : BOA identifier?*

**Procedure** Review the code for the use of the CORBA : : BOA identifier.

**Examples** 1. **BOA coding example**

a. **Client side** - The code below shows a C++ CORBA client BOA initialization for the ORBIX ORB. Other ORB vendors may have different initialization sequences.

```
int main
( int argc,
```

```

        char **argv
    )
    { MyServer_var MyVar;
      CORBA::ORB_ptr myOrbPtr
        = CORBA::ORB_init(argc,
argv, "Orbix");
      try
      { // The default is the local host:
        MyVar =
MyServer::_bind(":ServerName");
      } // End try
      catch ( CORBA::SystemException &sysEx
)
      { cerr << "Unexpected system
exception" << endl;
        cerr << &sysEx;
        exit(1);
      } // End CORBA::SystemException
      catch(...)
      { // an error occurred while trying
        // to bind to the grid object.
        cerr << "Bind to object failed" <<
endl;
        cerr << "Unexpected exception " <<
endl;
        exit(1);
      } // End catch ...
    } // End main

```

**b. Server side** - Use the code below as a model. This example shows a C++ CORBA server BOA init for the ORBIX ORB. For BOA, other ORBS will have a different initialization sequence.

```

try
{ MyObject::myOrb_
  = CORBA::ORB_init(argc, argv,
"Orbix");
  MyObject::myboa_
    = MyObject::myOrb_->BOA_init(argc,
argv, "Orbix_BOA");
} // End try
catch ( CORBA::SystemException &sysEx )
{ //some exception handling code
} // End catch
try
{ NoeLoggerCfg::myboa_-
>impl_is_ready("MyServiceName",
CORBA::ORB::INFINITE_TIMEOUT);
} // End try
catch ( CORBA::SystemException &sysEx )
{ //exception handling code
}

```

## 2. POA coding example

**a. Client side** - This example shows a C++ CORBA client POA init for the ORBIX ORB. For BOA, other

ORBS will have a different initialization sequence.

```
int main
( int argc,
  char **argv
)
{ CORBA::ORB_var myOrb =
CORBA::ORB_init(argc, argv);
  try
  { CORBA::Object_var obj
    = ... // however you get the
object reference
    if(CORBA::is_nil (obj))
    { cerr << "Nil object reference" <<
endl;
      throw 0;
    } // End if
  } // End try
  catch ( CORBA::SystemException &sysEx
)
  { cerr << "Unexpected system
exception" << endl;
    cerr << &sysEx;
    exit(1);
  } // End catch CORBA::SystemException
  catch ( ... )
  { cerr << "Unexpected system
exception" << endl;
    exit(1);
  } // End catch ...
  myinterface::myobject_var myvar;
  try
  { myvar =
myinterface::myobject::_narrow(obj);
  } // End try
  catch ( CORBA::SystemException &sysEx)
  { cerr << "Unexpected system
exception" << endl;
    cerr << &sysEx;
    exit(1);
  } // End catch CORBA::SystemException
} // End main
```

**b. Server side** - Use the code below as a model. This example shows a C++ CORBA server POA init for the ORBIX ORB. For POA, other ORBS will have a different initialization sequence.

```
int main
( int argc,
  char *argv[ ]
)
{ try
  { // initialize the ORB
    orb_var orb = CORBA::ORB_init(argc,
argv, "Orbix");
    // obtain an object reference for
the root POA
```

```
        object_var obj
            = orb->resolve_initial_references
("RootPOA");
        POA_var poa = POA::_narrow(obj);
        // incarnate a servant
        My_Servant_Impl servant;
        // Implicitly register the servant
with the root POA
        obj = servant._this ();
        //start the POA listening for
requests
        poa -> the_POAManager ()->activate
();
        //run the orb's event loop
        orb->run ();
    } // End try
    catch ( CORBA::SystemException &sysEx
)
    { // some exception handling code
    } // End catch
} // End main
```

# G1203

<b>Statement</b>	Localize frequently used CORBA-specific code in modules that multiple applications can use.	
<b>Rationale</b>	In a family of applications, similar patterns of CORBA <b>ORB</b> invocation sequences frequently arise. This is common in service object initialization, policy association, discovery, binding, and release handling. Implementing this functionality in a utility library paradigm localizes the code to reduce maintenance and facilitate extensibility, and assures consistency across the family of applications.	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b> <i>Do the standard object initialization CORBA invocations occur in more than one module?</i></p> <p><b>Procedure</b> The presence of “CORBA::ORB_var” or “CORBA::ORB_init” in C++ indicates ORB initialization. The presence of “CORBA::Object_var” in C++ indicates ORB access.</p> <p><b>Examples</b> None</p>
	2.	<p><b>Test</b> <i>Do the standard object policy association CORBA invocations occur in more than one module?</i></p> <p><b>Procedure</b> The presence of “CORBA::PolicyList” in C++ indicates policy presence.</p> <p><b>Examples</b> None</p>
	3.	<p><b>Test</b> <i>Do the standard object policy association CORBA invocations occur in more than one module?</i></p> <p><b>Procedure</b> The presence of “CORBA::PolicyList” in C++ indicates policy presence.</p> <p><b>Examples</b> None</p>
	4.	<p><b>Test</b> <i>Do the standard object discovery CORBA invocations occur in more than one module?</i></p> <p><b>Procedure</b> The presence of “Resolve_NamingService( )” in C++ indicates intended access to one of CORBA’s discovery capabilities.</p> <p><b>Examples</b> None</p>

- 5. Test** *Do the standard object binding and release CORBA invocations occur in more than one module?*
- Procedure** The presence of “:\_narrow(obj.in())” or “CORBA::is\_nil(” in C++ indicates activity associated with obtaining and validating an object binding to a legitimate reference. The presence of “CORBA(release)(” in C++ indicates intended release of a CORBA-bound object reference.
- Examples** None

# G1204

**Statement** Create configuration services to provide distributed user control of the appropriate configuration parameters.

**Rationale** For user-modifiable configuration settings that are intended to be accessible by distributed processes at runtime, the appropriate mechanism for implementation involves **CORBA** services. The first form is a network service to be invoked as a client by the target system application at initialization. This can support a consistent, network-wide distribution of startup parameters. The second form is a service implemented by the target application which allows communication to the application during execution (after startup). This allows **real-time** configuration changes for matters such as **POA** instantiation threading policies to address load management.

**Derived From** [G1119]

**Justifies**

**Reference d By** CORBA

**Acquisition Phase** Development

**Evaluation Criteria** **1 Test** *Is a service defined in the **IDL** to obtain the configuration parameters?*  
.

**Procedure** Review the code for a service that can be used to obtain configuration.

**Example** The following code is an example of a CORBA server that instantiates a configuration service. The service manages the individual configuration parameters for the servers on the **ORB**.

### Ada example

```

CORBA.ORB.IIOP_English;
pragma Elaborate_All(CORBA.ORB.IIOP_English);
with CORBA ;
with CORBA.BOA ;
with CORBA.ORB ;
with CORBA.Object ;
with Configuration.Impl ;
with Configuration.Helper ;
with Ada.Exceptions ;
with Ada.Text_IO ;
with my_CORBA ;
with Event_Ada_API ;
procedure Configuration_Server is
  -- required for OrbExpress
  First_Variable : CORBA.ORB.Life_Span ;
  -- declare the object instance
  Configuration_Object : Configuration.Ref ;
  --variables needed for ior writing

```

```

    No_Timeout : constant := 0.0;
    Config_Name : constant String
        := Configuration.Helper.Simple_Name ;
    Config_Host : Corba.String ;
    Config_Port : Corba.String ;
begin -- Configuration_Server
    -- create (and initialize) the object
    -- config file is read and the port needed
    -- is in there
    Configuration_Object
        := Configuration.Impl.Create(Config_Name) ;
    GET_HOSTNAME:
    begin
        Config_Host
            := Configuration.Get_String
                ( Self => Configuration_Object,
                  Name => Corba.To_Corba_String
                    ( "Local_Host_Shortname" )
                );
    exception -- GET_HOSTNAME
        when others =>
            Ada.Text_IO.Put_Line
                ( "ERROR: Missing parameter"
                  & "<Local_Host_Shortname> "
                  & "in the config_parameters.txt file."
                );
    end GET_HOSTNAME;
    GET_CS_PORT:
    begin
        Config_Port
            := Configuration.Get_String
                ( Self => Configuration_Object,
                  Name => Corba.To_Corba_String
                    ( "Config_Service_Port" )
                );
    Exception -- GET_CS_PORT
        when others =>
            Ada.Text_IO.Put_Line
                ( "ERROR: Missing parameter "
                  & "<Config_Service_Port> "
                  & "in the config_parameters.txt file."
                );
    end GET_CS_PORT;
    Ada.Text_IO.Put_Line
        ( "Host => "
          & Corba.To_Standard_String(Config_Host)
          & " Port => "
          & Corba.To_Standard_String(Config_Port)
        );
    --timeout 0 so we can write IOR out
    CORBA.BOA.Impl_Is_Ready
        ( Time_Out           => No_Timeout,
          Server_Instance_Name => Config_Name,
          Listen_On_Endpoints =>
            "tcp://"
            & Corba.To_Standard_String(Config_Host)
            & ":"
        );

```

```

        & Corba.To_Standard_String(Config_Port)
    );
    -----
--
-- HERE IS WHERE CODE FOR THE IOR TO BE
-- USED ON THE C++ ORB
--
--
-- get the IOR and write it to disk
my_CORBA.Write_IOR_To_File
( Server_Name => Config_Name,
  Server_Ref  =>
    CORBA.Object.Ref(Configuration_Object)
  );
READY_BLOCK:
begin
  -- notify subscribers of availability
  -- of configuration parameters via the
  -- event service
  Event_Ada_API.Send
    ( Channel_Name => "Config_Channel",
      Event         => "Configuration Service
Ready."
    );
  Exception - READY_BLOCK
  when others =>
    Ada.Text_IO.Put_line
      ( "Configuration_Server : "
        & Exception sending ready signal."
      );
end READY_BLOCK;
Ada.Text_IO.Put_line
( "Configuration_Server : "
  & Configuration Service Ready."
);
CORBA.BOA.Impl_Is_Ready
( Time_Out      => CORBA.Infinite_Timeout,
  Server_Instance_Name => Config_Name
  );
exception -- Configuration_Server
when X_Other: others =>
  Ada.Text_IO.Put_line
    ( "Configuration_Server : "
      & Ada.Exceptions.Exception_Name(X_Other)
    );
end Configuration_Server ;

```

### C++ example

The following code snippets depict a C++ server that instantiates a version collection service for an About box. It uses the IORs from the servers on the Ada ORB via the IOR files, and invokes those objects to get version information. It uses the utility templates for binding. It exemplifies the approach described in Encapsulate CORBA ORB operations for C++.

**Note:** This was done on the ORBIX C++ and Ada ORBs.

```

#include <iostream.h>
#include <rw/cstring.h>
#ifndef _STDIO_H
#include <stdio.h>
#endif
#ifndef _STRING_H
#include <string.h>
#endif
#ifndef _STDLIB_H
#include <stdlib.h>
#endif
#ifndef _ASSERT_H
#include <assert.h>
#endif
// Include files for all the objects desired for
// collecting version information
//Ada configuration service
#ifndef configuration_hh
#include <configuration.hh>
#endif
// include files for other desired services;
// removed for brevity
// other support objects and utilities
#ifndef _CORBA_UTILS__
#include <corba_utils.h>
#endif
#ifndef __LOG_API_H__
#include <log_api.h>
#endif
#ifndef _VERSION_AGENT_GLOBALS_H_
#include "version_agent_globals.h"
#endif
const RWCString
  Version_Agent_i::MSG_VERSION_NOT_FOUND_
  = "Version Info. not found for ";
const CORBA::ULong Version_Agent_i::MAXSERVERS_
  = 12;
Version_Agent_i::Version_Agent_i():
theVersionInfoPtr_(0)
{ theVersionInfoPtr_
  = new versionInfoType(MAXSERVERS_);
  theVersionInfoPtr_->length(MAXSERVERS_);
} // End constructor
Version_Agent_i::~~Version_Agent_i()
{ // Do nothing
} // End destructor
/*****
*****
FUNCTION NAME: createVersions
PURPOSE: helper function that gets the version info
INPUT:
OUTPUT:
*****/
void Version_Agent_i::createVersions ()
{ char *iorString;
  int bBindOk = 0;

```

```

int versionCnt = 0;
versionInfoType* rl = theVersionInfoPtr_;
CORBA::ULong MAXSERVERS
Version_Agent_i::MAXSERVERS_;
// server variables for all the objects desired
// for collecting version information
// most declarations removed for brevity
EventServiceFactory_var es_var;
// Ada configuration service
Configuration_var cfg_var;
// == load the versions of the individual
components
// Code for other services removed for brevity
// This is an ADA service using the IOR string
{ //***** config service
*****
    logMsg
      ( "get config service version",
        Log_Api::DEBUG_1_MSG
      );
    RWCString errMsg
      (
Version_Agent_i::MSG_VERSION_NOT_FOUND_.data()
      );
    errMsg.append ( "Configuration Service" );
    // here we get the IOR from the ADA orb using
    // the helper methods
    iorString = getIorFile("Configuration");
    //template class to hide binding issues to the
ADA ORB
    If ( iorString )
    { Ada_Binder < Configuration,
      Configuration_var > bo ( iorString );
      bBindOk = bo.bindToAda(&cfg_var) ;
      // get the version info and load it
      If ( bBindOk
          && !( CORBA::is_nil(cfg_var))
        )
      { try
        { char* str = cfg_var->version();
          if ( str )
          { (*theVersionInfoPtr_)[versionCnt]
            = CORBA::string_dup(str);
            delete str;
          } // End if
          else
          { (*theVersionInfoPtr_)[versionCnt]
            = CORBA::string_dup(errMsg.data());
          } // End else
        } // End try
      } // End if
      catch(...)
      { (*theVersionInfoPtr_)[versionCnt]
        = CORBA::string_dup(errMsg.data());
      } // End catch
      cfg_var->_closeChannel();
    } // End if
    else

```

```

        { (*theVersionInfoPtr_)[versionCnt]
          = CORBA::string_dup(errMsg.data());
        } // End else
        if(iorString)
        { free (iorString);
          iorString = NULL;
        } // End if
    } //endif iorstring
else
    { (*theVersionInfoPtr_)[versionCnt]
      = CORBA::string_dup(errMsg.data());
    } // End else
    //leaving scope releases the corba object
} //end cfg_svf
bBindOk = 0;
versionCnt++;
assert(versionCnt <= MAXSERVERS);
} // End createVersions
/*****
*****
FUNCTION NAME: start
PURPOSE: handle startup specific stuff
INPUT:
OUTPUT:
*****/
void Version_Agent_i:: start
( CORBA::Environment &IT_env
  ) throw (CORBA::SystemException)
{ //get all the version info
  createVersions();
} // End start
/*****
*****
FUNCTION NAME: stop
PURPOSE: handle stop specific stuff
INPUT:
OUTPUT:
*****/
void Version_Agent_i:: stop
( CORBA::Environment &IT_env
  ) throw (CORBA::SystemException)
{ // Release info
  // Let CORBA time out the service
  logMsg ( "stop received" );
  VersionAgentGlobals::myboa->setNoHangup ( 0 );
  VersionAgentGlobals::myboa->deactivate_impl
    ( "Version_Agent" );
} //end version impl

```

# G1205

<b>Statement</b>	Use non-source code persistence to store all user-modifiable <b>CORBA</b> service configuration parameters.	
<b>Rationale</b>	<p>For user-modifiable configuration settings that are host-specific and that are not intended to be accessible by distributed processes at runtime, the appropriate mechanism for implementation involves local persistent storage. The appropriate form of local storage depends on the local host architecture and may be file- or host-DBMS oriented. It is important that such parameters are not stored in source code that requires build processes for modification.</p> <p>It should be noted that for <b>SOA</b> services, configuration parameters relating to invoked services should not be service-host-specific at the invoking client application.</p>	
<b>Derived From</b>	[G1119]	
<b>Justifies</b>		
<b>Referenced By</b>	CORBA	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b>      <i>Are there any user-modifiable configuration parameters hard coded in the non-auto-generated files?</i></p> <p><b>Procedure</b>      Inspect the code for constant strings or constants that contain configuration parameters.</p> <p><b>Examples</b>      None</p>

# G1208

<b>Statement</b>	Add new functionality rather than redefining existing interfaces in a manner that brings incompatibility.	
<b>Rationale</b>	By not replacing old methods of objects, library functionality consumers can continue to operate and not be forced to upgrade.	
<b>Derived From</b>	[G1004]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b> <i>Are methods that are being replaced marked with deprecated tags?</i></p> <p><b>Procedure</b> Check revision history to make sure that methods are deprecated and not removed unless they have expired. "Expired" means that they have passed the expected shelf life, as defined by the project standards or other standards documentation.</p> <p><b>Examples</b> None</p>
	2.	<p><b>Test</b> <i>Do new methods being added contain information on methods they are replacing?</i></p> <p><b>Procedure</b> Check to make sure newly added methods contain information and rationale on the methods they are replacing.</p> <p><b>Examples</b> None</p>

## G1209

<b>Statement</b>	For Java, use <b>JDK</b> logging facilities.	
<b>Rationale</b>	Java has a built-in logging framework that is portable across platforms, projects, and installations.	
<b>Derived From</b>	[G1010]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b>      <i>Does the application use anything other than the specified logging frameworks?</i></p> <p><b>Procedure</b>      Check for use of logging frameworks other than the JDK.</p> <p><b>Examples</b>      None</p>

# G1210

<b>Statement</b>	For <i>.NET</i> , use Debug and Trace from the <code>System.Diagnostics namespace</code> .		
<b>Rationale</b>	.NET has a built-in logging framework that is portable across .NET projects and installations.		
<b>Derived From</b>	[G1010]		
<b>Justifies</b>			
<b>Referenced By</b>			
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Does the application use anything other than the specified logging frameworks?</i>
		<b>Procedure</b>	Check for use of logging frameworks other than <code>System.Diagnostics</code> .
		<b>Examples</b>	None

# G1211

<b>Statement</b>	For Java, use <b>JDBC</b> .		
<b>Rationale</b>	JDBC is Java's standard <b>API</b> for accessing databases.		
<b>Derived From</b>	[G1014]		
<b>Justifies</b>			
<b>Referenced By</b>			
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	1.	<b>Test</b>	<i>Does the application use an API other than JDBC to access the database?</i>
		<b>Procedure</b>	Check for vendor-specific APIs such as Oracle's OCI.
		<b>Examples</b>	None
	2.	<b>Test</b>	<i>Does the application use a vendor specific extension that is not ANSI-compliant <b>SQL</b>?</i>
		<b>Procedure</b>	Check for non- <b>ANSI</b> -compliant SQL.
		<b>Examples</b>	None

# G1212

<b>Statement</b>	For C/C++ and <i>.NET</i> use <i>ODBC</i> .	
<b>Rationale</b>	ODBC is C/C++ Window's standard <i>API</i> for accessing databases.	
<b>Derived From</b>	[G1014]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b> <i>Does the application use an API other than ODBC to access the database?</i>
		<b>Procedure</b> Check for vendor-specific API.
		<b>Examples</b> None
	<b>2.</b>	<b>Test</b> <i>Does the application use vendor-specific extension that is not ANSI-compliant <i>SQL</i>?</i>
		<b>Procedure</b> Check for non- <i>ANSI</i> -compliant SQL..
		<b>Examples</b> None

# G1213

<b>Statement</b>	Provide an architecture design document.	
<b>Rationale</b>	An architectural design document provides the evaluators with a roadmap of the application. This helps the evaluator verify that the application follows guidance such as using the Model View Controller model.	
<b>Derived From</b>	[G1020]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b>      <i>Do the project deliverables for evaluation include a document that contains the architectural design of the application?</i></p> <p><b>Procedure</b>      See if an architectural design document exists.</p> <p><b>Examples</b>      None</p>

# G1214

<b>Statement</b>	Provide a document with a plan for deprecating obsolete interfaces.	
<b>Rationale</b>	This information allows users to phase out deprecated interfaces. For instance, Sun plans to maintain backward compatibility for the <b>JDK</b> for seven years. This means developers can count on deprecated methods not being removed for seven years.	
<b>Derived From</b>	[G1020]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Do the project deliverables for evaluation include a document that contains a plan for deprecating obsolete interfaces?</i>
	<b>Procedure</b>	See if a document with a plan for deprecating obsolete interfaces exists.
	<b>Examples</b>	None.

# G1215

<b>Statement</b>	Provide a coding standards document.	
<b>Rationale</b>	The standards ensure a consistent code base. A coding standards document defines rules to keep code readable and maintainable.	
<b>Derived From</b>	[G1020]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Do the project deliverables for evaluation include a coding standards document?</i>
	<b>Procedure</b>	See if a coding standards document exists.
	<b>Examples</b>	None

# G1216

<b>Statement</b>	Provide a software release plan document.	
<b>Rationale</b>	The release plan document ensures that there is a formal process for releasing the software. It includes a description of how to acquire the software from SCM and how to build, label, and release it.	
<b>Derived From</b>	[G1020]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b>      <i>Do the project deliverables for evaluation contain a release plan document?</i></p> <p><b>Procedure</b>      See if a software release plan exists.</p> <p><b>Examples</b>      None</p>

# G1217

<b>Statement</b>	<i>Components</i> should be externally configurable.
<b>Rationale</b>	To be portable and to accommodate reuse, components must be configurable using external descriptors usually defined in <i>XML</i> . Examples of things that might need to be configured include: <ul style="list-style-type: none"> <li>• A data source for the component to obtain a <i>JDBC</i> connection</li> <li>• The location of a service that the component must communicate with</li> <li>• The location of implementation classes that the component uses</li> </ul>
<b>Derived From</b>	
<b>Justifies</b>	
<b>Referenced By</b>	<i>Implement a component-based architecture</i> , [G1002]
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	<p><b>1.Test</b>     <i>Are deployment descriptors used?</i></p> <p><b>Procedure</b> Check for the existence of deployment descriptors in the appropriate directories. Usually the file is named <b>web.xml</b>.</p> <p><b>Examples</b> None</p>

# G1218

<b>Statement</b>	Support operation in an automated mode.	
<b>Rationale</b>	During testing, human interaction can be a cause of error and unrepeatable results. Operating in automated mode can eliminate these errors.	
<b>Derived From</b>	[G1190]	
<b>Justifies</b>	[G1002]	
<b>Referenced By</b>	<i>Automate the build process</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Does the tool have a build all target?</i>
	<b>Procedure</b>	Check the build scripts or descriptors of the build tool for the ability to build the entire project, system, or application.
	<b>Examples</b>	None

# G1219

<b>Statement</b>	Check out files from configuration control.	
<b>Rationale</b>	To make sure all the parts of the build are under configuration control, compare all files with the configuration baseline, and download the appropriate files.	
<b>Derived From</b>	[G1190]	
<b>Justifies</b>	[G1002]	
<b>Referenced By</b>	<i>Automate the build process</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Does the tool have a checkout target?</i>
	<b>Procedure</b>	Check the build scripts or descriptors of the build tool for the ability to check out the entire project, system, or application.
	<b>Examples</b>	None

# G1220

<b>Statement</b>	Compile source code and dependencies that have been modified.		
<b>Rationale</b>	To limit the changes made between builds, only compile code that has been modified. If there are no intermediate files, then compile all files.		
<b>Derived From</b>	[G1190]		
<b>Justifies</b>	[G1002]		
<b>Referenced By</b>	<i>Automate the build process</i>		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	1.	<b>Test</b>	<i>Does the tool have a compile target?</i>
		<b>Procedure</b>	Check the build scripts or descriptors of the build tool for the ability to compile the entire project, system, or application.
		<b>Examples</b>	None
	2.	<b>Test</b>	<i>Do all the intermediate files (e.g., .obj or .class) have the same date and time stamps?</i>
		<b>Procedure</b>	Scan the files for date and time stamps.
		<b>Examples</b>	None

# G1221

<b>Statement</b>	Create libraries or archives after all required compilations are completed.	
<b>Rationale</b>	Libraries should be able to be recreated independently of any executables and should always verify that any intermediate files are not stale.	
<b>Derived From</b>	[G1190]	
<b>Justifies</b>	[G1002]	
<b>Referenced By</b>	<i>Automate the build process</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Does the tool have a generate library target?</i>
	<b>Procedure</b>	Check the build scripts or descriptors of the build tool for the ability to generate the composing libraries or archives.
	<b>Examples</b>	None

# G1222

<b>Statement</b>	Create executables		
<b>Rationale</b>	An executable is dependent on many files, including source files, intermediate files, and libraries or archives. The building of the executable must support a control process that includes configuration management, compiling, and testing.		
<b>Derived From</b>	[G1190]		
<b>Justifies</b>			
<b>Referenced By</b>	<i>Automate the build process</i>		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Does the tool have an executable target?</i>
		<b>Procedure</b>	Check the build scripts or build tool descriptors for the ability to build the executables for the entire project, system, or application.
		<b>Examples</b>	None

# G1223

<b>Statement</b>	Capable of running unit tests.	
<b>Rationale</b>	All code should be able to be tested independently of creating intermediate files, libraries, or executables.  Tests should be unit tests as well as system-level tests.	
<b>Derived From</b>	[G1190]	
<b>Justifies</b>		
<b>Referenced By</b>	<i>Automate the build process</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b>      <i>Does the tool have a test target?</i></p> <p><b>Procedure</b>      Check the build scripts or descriptors of the build tool for the ability to test the entire project, system, or application.</p> <p><b>Examples</b>      None</p>

# G1224

<b>Statement</b>	Clean out intermediate files that can be regenerated.						
<b>Rationale</b>	For security reasons, all files that comprise the build need to be under configuration control. Cleaning out all files is essential in ensuring that only approved code is incorporated into the build.						
<b>Derived From</b>	[G1190]						
<b>Justifies</b>	{List of guidance statements that this guidance statement justifies (i.e., children) <b>Note:</b> should be hypertext links}						
<b>Referenced By</b>	<i>Automate the build process</i>						
<b>Acquisition Phase</b>	Development						
<b>Evaluation Criteria</b>	<table border="0" style="margin-left: 20px;"> <tr> <td style="vertical-align: top;"><b>1. Test</b></td> <td><i>Does the tool have a clean target?</i></td> </tr> <tr> <td style="vertical-align: top;"><b>Procedure</b></td> <td>Check the build scripts or descriptors for the build tool for the ability to remove the entire project, system, or application files.</td> </tr> <tr> <td style="vertical-align: top;"><b>Examples</b></td> <td>None</td> </tr> </table>	<b>1. Test</b>	<i>Does the tool have a clean target?</i>	<b>Procedure</b>	Check the build scripts or descriptors for the build tool for the ability to remove the entire project, system, or application files.	<b>Examples</b>	None
<b>1. Test</b>	<i>Does the tool have a clean target?</i>						
<b>Procedure</b>	Check the build scripts or descriptors for the build tool for the ability to remove the entire project, system, or application files.						
<b>Examples</b>	None						

# G1225

<b>Statement</b>	The build tool should be independent of the Integrated Development Environment.		
<b>Rationale</b>	Some build tools are tightly coupled with an Integrated Development Environment ( <i>IDE</i> ) that causes vendor lock-in and license issues when the software is delivered to the government.		
<b>Derived From</b>	[G1190]		
<b>Justifies</b>			
<b>Referenced By</b>	<i>Automate the build process</i>		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Does the build tool require a license?</i>
		<b>Procedure</b>	Check for files with the name <b>makefile</b> .
		<b>Examples</b>	None
	<b>2.</b>	<b>Test</b>	<i>Is the build tool one of the recognized standards, such as ant?</i>
		<b>Procedure</b>	Check for files named <b>build.xml</b> .
		<b>Examples</b>	None
	<b>3.</b>	<b>Test</b>	<i>Is the build tool one of the recognized standards, such as make or nmake?</i>
		<b>Procedure</b>	Check for files with the name <b>makefile</b> .
		<b>Examples</b>	None

# G1236

<b>Statement</b>	Do not hard-code the <i>endpoint</i> of a web-service vendor.	
<b>Rationale</b>	An endpoint is the URL or location of the <i>web service</i> on the <i>Internet</i> . A major benefit of web services is the ability to relocate a web service to another location, or dynamically discover and use a web service using registry facilities. Some web service vendors hard- code the URL of the web service, which causes maintenance and portability problems.	
<b>Derived From</b>	[G1091]	
<b>Justifies</b>		
<b>Referenced By</b>	Insulation and structure guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Are there any hard-coded web service vendor endpoints in the client code?</i></p> <p><b>Procedure</b> Parse the code and look for hard-coded endpoints. These endpoints look just like a normal HTTP web address.</p> <p><b>Examples</b> None</p>

# G1237

<b>Statement</b>	Do not hard-code the configuration data of a web-service vendor.	
<b>Rationale</b>	Some vendors generate code that passes web-service vendor-specific configuration data during initialization or startup. This reduces the portability of the code and can cause maintenance problems later.	
<b>Derived From</b>	[G1091]	
<b>Justifies</b>		
<b>Referenced By</b>	Insulation and structure guidance	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1.</b>	<p><b>Test</b> <i>Is there any web-service vendor-specific configuration data in the client code?</i></p> <p><b>Procedure</b> Parse the code and look for hard-coded configuration data that might be used to configure the vendor's web service.</p> <p><b>Examples</b> None</p>

# G1239

<b>Statement</b>	Vendor-dependent connections to the enterprise should isolate vendor-specifics using design patterns (e.g., <i>façade</i> , <i>proxy</i> , or <i>adapter</i> ) or property files.		
<b>Rationale</b>	Increases maintainability. Guidance [G1071] asserts that vendor-neutral connection mechanisms should be used. When vendor-specific connection mechanisms are unavoidable, this guidance will apply.		
<b>Derived From</b>	[G1071]		
<b>Justifies</b>			
<b>Referenced By</b>			
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Is the connection mechanism vendor-dependent?</i>
		<b>Procedure</b>	Examine the source code for vendor-specific imports or includes.  Make sure that all references to the vendor-specific connection mechanisms are isolated to a single class (like a helper) or set of methods that are used as part of an isolation design pattern such as façade, proxy, or adapter.  Also, look for hard-coded vendor-specific connection strings.
		<b>Examples</b>	None

# G1245

<b>Statement</b>	Isolate the web-service portlet from platform dependencies using the <i>OASIS WSRP Specification 1.0</i> protocol.
<b>Rationale</b>	The OASIS <b>WRSP</b> 1.0 Specification accounts for the fact that <i>producers</i> and <i>consumers</i> may be implemented on very different platforms, such as a J2EE -based web service, a web service implemented on Microsoft's .Net platform, or a <i>portlet</i> published directly by a <i>portal</i> .
<b>Evaluation Criteria</b>	<p>1.    <b>Test</b>            <i>Does the web service implement the WRSP Markup interface?</i></p> <p>          <b>Procedure</b>    Look for the definition of the <code>getMarkup</code>, <code>performBlockingInteraction</code>, <code>initCookie</code> and <code>releaseSessions</code> methods as defined in the OASIS WSRP Markup <b>API</b> Specification.</p> <p>          <b>Examples</b>     <pre>public MarkupResponse getMarkup     ( RegistrationContext       registrationContext,         PortletContext portletContext,         RuntimeContext runtimeContext,         UserContext userContext,         MarkupParams markupParams       ) throws java.lang.Exception public void performBlockingInteraction     ( RegistrationContext       registrationContext,         PortletContext portletContext,         RuntimeContext runtimeContext,         UserContext userContext,         MarkupParams markupParams,         InteractionParams       interactionParams       ) throws java.lang.Exception public Extension[] initCookie     ( RegistrationContext       registrationContext       ) throws java.lang.Exception public Extension[] releaseSessions     ( RegistrationContext       registrationContext,         java.lang.String[] sessionIDs       ) throws java.lang.Exception</pre></p> <p>2.    <b>Test</b>            <i>Does the web service implement the WRSP Service Description interface?</i></p> <p>          <b>Procedure</b>    Look for the occurrence of the <code>getService</code>, <code>register</code>, and <code>getServiceDescription</code> methods as defined in the OASIS WSRP Service Description API Specification.</p>

- Examples**
- ```
public static ServiceDescriptionService
getService
    ( java.lang.String baseEndpoint
    ) throws java.lang.ExceptionThrows:
jpublic ServiceDescription
getServiceDescription
    ( RegistrationContext
registrationContext,
    java.lang.String[] desiredLocales
    ) throws java.lang.Exception
```
- 3. Test** *Does the web service implement the WSRP Portlet Configuration interface?*
- Procedure** Look for the occurrence of the getService, getPortletDescription, clonePortlet, destroyPortlets, setPortletProperties, getPortletProperties and getPortletPropertyDescription methods as defined in the OASIS WSRP Portlet Configuration API Specification.
- Examples**
- ```
public static PortletManagementService
getService
    ( java.lang.String baseEndpoint
    ) throws java.lang.Exception
public PortletDescriptionResponse
getPortletDescription
    ( RegistrationContext
registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] desiredLocales
    ) throws java.lang.Exception
public PortletContext clonePortlet
    ( RegistrationContext
registrationContext,
    PortletContext portletContext,
    UserContext userContext
    ) throws java.lang.Exception
public DestroyPortletsResponse
destroyPortlets
    ( RegistrationContext
registrationContext,
    java.lang.String[] portletHandles
    ) throws java.lang.Exception
public PortletContext
setPortletProperties
    ( RegistrationContext
registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    PropertyList propertyList
    ) throws java.lang.Exception
public PropertyList
getPortletProperties
```

```

    ( RegistrationContext
registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] names
    ) throws java.lang.Exception
public
PortletPropertyDescriptionResponse
getPortletPropertyDescription
    ( RegistrationContext
registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] desiredLocales
    ) throws java.lang.ExceptionThrows

```

**4. Test**

*Does the web service implement the WSRP  
Registration interface?*

**Procedure**

Look for the occurrence of the getService, register, deregister, and modifyRegistration methods as defined in the OASIS WSRP Specification.

**Examples**

```

public static RegistrationService
getService
    ( java.lang.String baseEndpoint
    ) throws java.lang.Exception
public RegistrationContext register
    ( java.lang.String consumerName,
    java.lang.String consumerAgent,
    boolean methodGetSupported,
    java.lang.String[] consumerModes,
    java.lang.String[]
consumerWindowStates,
    java.lang.String[]
consumerUserScopes,
    java.lang.String[]
customUserProfileData,
    Property[] registrationProperties
    ) throws java.lang.Exception
public ReturnAny deregister
    ( java.lang.String
registrationHandle,
    byte[] registrationState
    ) throws java.lang.Exception
public RegistrationState
modifyRegistration
    ( RegistrationContext
registrationContext,
    RegistrationData registrationData
    ) throws java.lang.Exception

```



---

# Best practices

## **Best practices details**

This section contains a complete set of the numbered best practices that are referenced elsewhere in this guide.

## BP1038

<b>Statement</b>	Use one of these standard fonts in web pages, in this order of preference: Verdana, Universal, Sans Serif. Do not use Times New Roman.
<b>Rationale</b>	Web pages are easier to read with suggested fonts.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

## BP1039

<b>Statement</b>	Do not underline any text unless it is a link.
<b>Rationale</b>	Underlined text is the default behavior of an <i>HTML</i> link. Many users consider this the norm and may find a <i>web page</i> difficult to read if other items are underlined.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

## BP1040

<b>Statement</b>	Use hex codes for all colors (e.g., #FFFF33), never the color name (e.g., yellow). For an online hexadecimal color chart, see <a href="http://webmonkey.wired.com/webmonkey/reference/color_codes/">http://webmonkey.wired.com/webmonkey/reference/color_codes/</a> .
<b>Rationale</b>	Increases compatibility between browsers. Industry standard.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

## BP1041

<b>Statement</b>	Do not change the default colors of the links.
<b>Rationale</b>	<i>Web pages</i> are easier to read because users have become accustomed to the default colors.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

# BP1042

<b>Statement</b>	Do not build a <i>web page</i> where the horizontal width is greater than the screen. Vertical scrolling is fine. Plan for the lowest common denominator to be super-VGA resolution or 600 x 800.
<b>Rationale</b>	This enables you to print pages on most printers and render pages on most displays.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

## BP1054

<b>Statement</b>	Use standard controls that provide input choices for the user. These controls might include radio buttons, check boxes, list boxes, and drop-downs.
<b>Rationale</b>	Reduces user input errors.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

## BP1075

<b>Statement</b>	All application developers should use the <i>Ant</i> build tool to build, package, and deploy <i>J2EE</i> applications in their development environments.
<b>Rationale</b>	There are several good <i>IDEs</i> on the market to support developing J2EE applications. However, different IDEs tend to auto-generate code that does not port to other IDEs, creating a problem when sharing code between groups using different IDEs. To minimize these compatibility issues and development environment turf wars, common building tools need to be used.
<b>Referenced By</b>	<i>Automate the build process</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

## BP1076

<b>Statement</b>	When deploying a new application to a WebLogic application server (e.g., <i>ear, war, rar</i> ), do not edit the WebLogic startup file to add application-specific information. This file is used for server startup only and should not contain application-specific logic. The system administrator must approve and coordinate all updates to this file.
<b>Rationale</b>	Server startup should not depend on an individual application.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

# BP1077

<b>Statement</b>	Do not edit the <b>config.xml</b> file manually. The <b>config.xml</b> file is the persistent store used by the WebLogic server to store runtime configuration parameters. Instead, use the WebLogic management console to configure the WebLogic server. Any edits done through the management console will be written to <b>config.xml</b> .
<b>Rationale</b>	Editing the <b>config.xml</b> file manually can introduce unpredictable server errors and cause loss of important configuration data.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

# BP1097

<b>Statement</b>	Use the <code>System.Text.StringBuilder</code> class for repetitive string modifications such as appending, removing, replacing, or inserting characters.	
<b>Rationale</b>	Strings in <i>.NET</i> are immutable. This means that every time a string is created as a result of a string operation such as concatenation, a new string is created for each intermediate string in a set of operations. This has a lot of string management overhead. <code>StringBuilder</code> avoids these problems.	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Are there repetitive string operations that use string operations instead of <code>StringBuilder</code> operations?</i>
	<b>Procedure</b>	Scan all C# code for repetitive string operations such as appending, removing, replacing, or inserting characters.
	<b>Examples</b>	None

# BP1098

<b>Statement</b>	Write all <i>.NET</i> code in C#.	
<b>Rationale</b>	Because of the high degree of similarities between C# and Java, .NET code written in C# is easily ported to Java. .NET has removed most of the advantages of one language (C#, C++, J++, VB) over another.	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Are any .NET languages delivered other than C#?</i>
	<b>Procedure</b>	Scan delivered code for registered .NET file extensions other than C#.
	<b>Examples</b>	None

# BP1100

<b>Statement</b>	Compile all code using the <i>.NET</i> Just-In-Time compiler.							
<b>Rationale</b>	There are two different ways to generate machine code within the .NET environment: <i>Just-In-Time (JIT)</i> and <i>Native Image Generator (NGEN)</i> . The NGEN method provides performance advantages by using the native image cache portion of the global assembly cache, which is specific to the machine where the .NET <i>common language runtime</i> is installed. It is machine-dependent and is less portable.							
<b>Acquisition Phase</b>	Development							
<b>Evaluation Criteria</b>	<b>1.</b>	<table> <tr> <td><b>Test</b></td> <td><i>Is ngen.exe used?</i></td> </tr> <tr> <td><b>Procedure</b></td> <td>Scan all delivered code for the use of ngen.exe or the ngen command.</td> </tr> <tr> <td><b>Examples</b></td> <td>None</td> </tr> </table>	<b>Test</b>	<i>Is ngen.exe used?</i>	<b>Procedure</b>	Scan all delivered code for the use of ngen.exe or the ngen command.	<b>Examples</b>	None
<b>Test</b>	<i>Is ngen.exe used?</i>							
<b>Procedure</b>	Scan all delivered code for the use of ngen.exe or the ngen command.							
<b>Examples</b>	None							

# BP1109

<b>Statement</b>	Use Windows unattended setup to install Message Queuing software by remotely using an answer file.
<b>Rationale</b>	It should not be necessary to have a “human in the loop” when installing the Message queuing software. This reduces errors during installation and helps establish a uniform installation base.
<b>Derived From</b>	
<b>Justifies</b>	[BP1226], [BP1227], [BP1228], [BP1229], [BP1230]
<b>Referenced By</b>	
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel best practices to evaluate this guidance.

# BP1111

<b>Statement</b>	Mark all MSMQ messages as recoverable.	
<b>Rationale</b>	MSMQ normally only stores the contents of messages in memory, which will be lost if a power, hardware, or software failure occurs. By marking messages as recoverable, messages are also stored to disk so the contents can be recovered after a failure.	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Are all messages and message queues marked as recoverable?</i>
	<b>Procedure</b>	Scan the code for the creation of messages and message codes, and make sure each has the <code>recoverable</code> attribute set to <code>true</code> .
	<b>Examples</b>	None

# BP1112

<b>Statement</b>	Specify all MSMQ queues as transactional if they support multiple-step processes.
<b>Rationale</b>	<i>Transactions</i> allow multi-step processes to behave correctly when a <i>rollback</i> occurs.
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

## BP1116

<b>Statement</b>	If using Java-based messaging (e.g., <i>JMS</i> ), register destinations in <i>JNDI</i> so message clients can use JNDI to look up these destinations.
<b>Rationale</b>	JNDI is an industry standard for Java-based applications.
<b>Referenced By</b>	<i>Java Naming &amp; Directory Interface (JNDI)</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

# BP1122

<b>Statement</b>	When using <b>CORBA</b> strings, follow the best practice guidelines in the child documents listed below.
<b>Rationale</b>	Aids in memory management by reducing memory leaks and memory-related errors.
<b>Justifies</b>	[BP1231], [BP1232], [BP1233], [BP1234], [BP1235]
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel best practices to evaluate this guidance.

# BP1139

<b>Statement</b>	Adhere to a core set of <i>SQL</i> features. Minimize use of proprietary extensions to the SQL standard.	
<b>Rationale</b>	It is almost impossible to use Oracle, SQL Server, or DB-2 without using proprietary extensions to the SQL standards. In many cases, however, these extensions are later incorporated into the standard.	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Have the developers adhered to a core set of features and minimized use of proprietary extensions to the SQL standard?</i>
	<b>Procedure</b>	Examine a representative sample of database scripts and stored procedures.
	<b>Examples</b>	None

# BP1140

<b>Statement</b>	Use SQL-2003 features in preference to <i>SQL-92</i> or <i>SQL-99</i> .	
<b>Rationale</b>	SQL-2003 includes many <i>XML</i> and <i>OODB</i> extensions and features. Use it in preference to SQL-99 or SQL-92 entry-level features, to justify the recommendations against using native XML databases and OODB databases.	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Have the developers used SQL-2003 features rather than SQL-92 or SQL-99 features?</i>
	<b>Procedure</b>	Examine a representative sample of database scripts and stored procedures.
	<b>Examples</b>	None

# BP1143

<b>Statement</b>	Use a database modeling tool that supports a two-level model ( <i>Conceptual/Logical</i> and <i>Physical</i> ) and ISO-11179 <i>data exchange</i> standards.	
<b>Rationale</b>	ISO-11179 is a metadata repository standard. The tools we have been using operate in a mode where the model is stored locally in an <i>XML</i> file or in a vendor-specific repository. For many applications, there is no need to use the repository at all. CM could be affected by checking the model in and out of a tool such as Source Safe. Entity-Relationship data model is synonymous with a Conceptual Data Model.	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Is a database modeling tool being used and does it support the ISO-11179 data exchange standards?</i>
	<b>Procedure</b>	Verify that the requirement for a database modeling tool is included in the system requirements. If ISO-11179 standard-based metadata repository products become available, determine whether the product provides an interface thereto.
	<b>Examples</b>	None

# BP1145

<b>Statement</b>	<i>Conceptual and logical models</i> should be vendor-neutral whenever possible.		
<b>Rationale</b>	The leading database vendors do not have a common set of data types or object name length limitations, and there are no <i>ANSI</i> standards that address these issues. To maintain vendor-neutral models, vendor-specific features will not be accepted.		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Has the data model been designed using vendor-neutral design criteria?</i>
		<b>Procedure</b>	Examine the conceptual/logical data model.
		<b>Examples</b>	None

## BP1177

**Statement** To ensure decoupling from the visualization layer, do not develop to the ATLAS *APIs*. Develop to either the JMTK COE APIs, or to the *OGC* open-standards APIs (*GO-1*: an OGC abstraction layer added to ATLAS that allows developers to use OGC GO-1/GEOBJECTSAPI calls and Geobjects). C2PC bindings allow developers to use either strategy.

**Rationale**

**Acquisition Phase** Development

**Evaluation Criteria** None

# BP1226

<b>Statement</b>	Locate the Answer file for the MSMQ in the MSMQ installation directory on the computer from where the unattended setup will be initiated.	
<b>Rationale</b>	This allows the installation process to be consistently repeated. See ( <i>MSMQ Concepts 3.6</i> ).	
<b>Derived From</b>	[BP1109]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Is the answer file in the MSMQ installation directory?</i>
	<b>Procedure</b>	Find out where the MSMQ answer files are located. If the location is not provided, search for a file that contains one of the answer file settings listed in this guidance.
	<b>Examples</b>	None

# BP1227

<b>Statement</b>	Do not allow dependent clients to be installed.		
<b>Rationale</b>	<p>MSMQ-dependent clients require synchronous access to an MSMQ server and create performance issues on the server. Consequently, dependent clients cannot operate if they are disconnected from the rest of the <i>enterprise</i> networks.</p> <p>Dependent clients cannot be run under local accounts.</p> <p>Dependent clients leave all encrypted messages in plain text between the client and server.</p>		
<b>Derived From</b>	[BP1109]		
<b>Justifies</b>			
<b>Referenced By</b>			
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	1.	<b>Test</b>	<i>Is msmq_LocalStorage = ON in the MSMQ answer file?</i>
		<b>Procedure</b>	Scan the answer file for the setting.
		<b>Examples</b>	None
	2.	<b>Test</b>	<i>Is SupportingServer set in the MSMQ answer file?</i>
		<b>Procedure</b>	Scan the answer file for the setting.
		<b>Examples</b>	None

# BP1228

<b>Statement</b>	Do not use the features found in MSMQ v3.0 HTTP transport.		
<b>Rationale</b>	This is an extension of the Internet Information Services ( <b>IIS</b> ) and should be avoided.		
<b>Derived From</b>	[BP1109]		
<b>Justifies</b>			
<b>Referenced By</b>			
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Is msmq_HTTPSupport = OFF in the MSMQ answer file?</i>
		<b>Procedure</b>	Scan the answer file for the setting.
		<b>Examples</b>	None

## BP1229

<b>Statement</b>	Do not use the features found in MSMQ v3.0 message queue triggering.	
<b>Rationale</b>	This is an extension of the Internet Information Services ( <i>IIS</i> ) and should be avoided.	
<b>Derived From</b>	[BP1109]	
<b>Justifies</b>		
<b>Referenced By</b>		
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Is msmq_TriggersService = OFF in the MSMQ answer file?</i>
	<b>Procedure</b>	Scan the answer file for the setting.
	<b>Examples</b>	None

# BP1230

<b>Statement</b>	Do not use the SupportLocalAccountsOrNT4 feature.		
<b>Rationale</b>	This entry enables weakened security for Active Directory on a <i>domain</i> controller, which is then replicated to all other domain controllers in every domain in your forest.  See ( <i>MSMQ Concepts 3.6</i> )		
<b>Derived From</b>	[BP1109]		
<b>Justifies</b>			
<b>Referenced By</b>			
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Is SupportLocalAccountsOrNT4 = FALSE in the MSMQ answer file?</i>
		<b>Procedure</b>	Scan the answer file for the setting.
		<b>Examples</b>	None

# BP1231

<b>Statement</b>	Use <code>CORBA::String_var</code> in <i>IDL</i> to pass string types in C++.		
<b>Rationale</b>	To correct memory management and reduce memory leaks and runtime faults.		
<b>Derived From</b>	[BP1122]		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Is <code>String_var</code> used in the implementation code that was not auto generated?</i>
		<b>Procedure</b>	Check implementation code that was not autogenerated for all occurrences of "string" and verify that they are <code>String_var</code> .
		<b>Examples</b>	None

# BP1232

<b>Statement</b>	Do not pass or return a zero or null pointer; instead, pass an empty string.	
<b>Rationale</b>	To correct memory management and reduce memory leaks and runtime faults.	
<b>Derived From</b>	[BP1122]	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Are there any returns that contain pointers that are assigned zero?</i>
	<b>Procedure</b>	Check code to make sure that all strings returned always have a safety check for zero or null pointers, and assign them to empty strings.
	<b>Examples</b>	None

## BP1233

<b>Statement</b>	Do not assign <code>CORBA::String_var</code> type to INOUT method parameters.	
<b>Rationale</b>	To correct memory management and reduce memory leaks and runtime faults.	
<b>Derived From</b>	[BP1122]	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Are there any <b>IDL</b> implementation classes using methods that contain <code>CORBA::String_var</code>?</i>
	<b>Procedure</b>	Inspect CORBA code to make sure INOUT parameters are not assigned to <code>CORBA::String_var</code> values.
	<b>Examples</b>	None

# BP1234

<b>Statement</b>	Assign string values to OUT, INOUT, or RETURN parameters using operations to allocate or duplicate values rather than creating and deleting values.		
<b>Rationale</b>	Correct memory management and reduce memory leaks and reduce runtime faults.		
<b>Derived From</b>	[BP1122]		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	1.	<b>Test</b>	<i>Are string_dup, string_alloc and string_free being used?</i>
		<b>Procedure</b>	Search CORBA code for the use of string_dup, string_alloc and string_free.
		<b>Examples</b>	None
	2.	<b>Test</b>	<i>Are new and delete operators being used for strings being assigned to OUT, INOUT or RETURN parameters?</i>
		<b>Procedure</b>	Inspect CORBA code to make sure OUT, INOUT, and RETURN parameters are not using strings managed with the new and delete operators.
		<b>Examples</b>	None

## BP1235

<b>Statement</b>	Assign string values to returned-as-attribute values using operations to allocate or duplicate values rather than creating and deleting values.	
<b>Rationale</b>	To correct memory management and reduce memory leaks and runtime faults.	
<b>Derived From</b>	[BP1122]	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	1.	<p><b>Test</b> <i>Are string_dup, string_alloc and string_free being used?</i></p> <p><b>Procedure</b> Search <b>CORBA</b> code for the use of string_dup, string_alloc and string_free.</p> <p><b>Examples</b> None</p>
	2.	<p><b>Test</b> <i>Are new and delete operators being used for strings being returned-as-attribute?</i></p> <p><b>Procedure</b> Inspect CORBA code to make sure returned-as-attribute string values are not using strings managed with the new and delete operators.</p> <p><b>Examples</b> None</p>

## BP1240

**Statement**

Present complete and coherent sets of concepts to the user.

**Rationale**

The *interface* should not require the consumer to continually implement multiple interfaces when a single interface can accomplish the same thing.

## BP1241

**Statement**

Design *interfaces* to be statically typed.

**Rationale**

Designing a statically typed interface allows consumers to use early binding rather than late binding. This minimizes the risk for runtime errors due to late binding.

## BP1242

**Statement**

Minimize the *interface's* dependencies on other interfaces.

**Rationale**

Minimizing the dependency of an interface on other interfaces simplifies the use of the interface by consumers.

## BP1243

**Statement**

Express *interfaces* in terms of application-level types.

**Rationale**

Use application-level types to maintain the meaning of values used with the interface. This enables data validation and other runtime safety checks against the data.

# BP1244

<b>Statement</b>	Use assertions only to aid development and <i>integration</i> .
<b>Rationale</b>	<p>Assertions allow you to evaluate Boolean expressions to determine if the code is executing within the proper operating constraints. For example, if a calculated temperature is supposed to be between -273 degrees and +1,000 degrees, you can test the results of the calculation with an assertion. Once the code is tested and/or integrated, this calculation no longer needs to occur after each calculation.</p> <p>Assertion execution is integrated into the <i>compiler</i>. Consequently, you can add it into the executable or eliminate it by setting compiler options (i.e. switches). Assertions are therefore ideal for adding code that is useful during development or integration, but wasteful in delivered code.</p>

## Derived From

## Justifies

**Referenced By** Public interface design

**Acquisition Phase** Development

**Evaluation Criteria**      **1.    Test**      *Do public methods that implement interfaces have assertions?*

**Procedure**      Check all implementations of public interfaces to ensure that all public methods that are part of the interface do not use the `assert` command.

**Examples**      The following example shows a correct implementation of a public method in a public interface.

```
public interface NameInterface is
public String getName
( int nameID )
  Throws IllegalArgumentException
  {
    /* precondition check */
    if ( nameID <= 0
        || nameID > MAX_NAMES
        )
      { throw new
        IllegalArgumentException
          ("Illegal id number: " +
nameID);
        }
    . . .// Do the computation
    return theResult;
  } // End getName
```

```
} // NameInterface
```

The following example shows an incorrect

implementation of a public method in a public interface. Do not use the implementation exemplified by the red code.

```
public interface NameInterface is
public String getName
( int nameID )
{
    /* precondition check */
    assert nameID <= 0
        || nameID > MAX_NAMES
        : "Illegal id number: " +
nameID);
... . . . // Do the computation
    return theResult;
} // End getName

} // NameInterface
```

# BP1246

**Statement**

Java-based portlets should be based on *JSR 168*.

**Rationale**

JSR 168 enables *interoperability* between Java *portlets* and *portals*. This specification defines a set of *APIs* for portal computing that addresses the areas of aggregation, *personalization*, presentation, and security.

<http://www.jcp.org/en/jsr/detail?id=168>

## BP1247

**Statement**

Encapsulate Java-based *portlets* in a *WAR* file.

**Rationale**

Storing *JSR-168*-compliant code in the portal container improves *interoperability* and code reuse.

# BP1248

**Statement** Do not assign the same name to multiple database objects such as databases, schema, users, tables, views, or indices.

**Rationale** The names of schemas, users, tables, and columns need to be unique and descriptive. Unfortunately, it is possible (but undesirable) to give the same name to multiple objects: for example, assigning the name “employee” to a database, table, and column. Many naming conventions get around this by appending a suffix that indicates the kind of object: for example, Employee\_Db, Employee\_Tbl, Employee\_Id, Employee\_Indx.

Avoid generic column names such as “ID.” Systems often have many kinds of IDs, and even if the system really only does have a single ID, it will be more difficult to merge with other databases if they have also used the column name “ID.”

Some DBMSs support mixed-case names of unlimited length, while others are case-insensitive. For portability, assume that names are case-insensitive and limited to 30 characters. Do not use reserved words from the *SQL-92*, *SQL:1999*, or SQL:2003 standards.

**Derived From**

**Justifies** *BP1249, BP1250, BP1251, BP1252, BP1253, BP1254*

**Referenced By** *RDBMS internals*

**Acquisition Phase** Development

**Evaluation Criteria**

- Test** *Is there a naming convention?*

**Procedure** Check for the existence of a document that governs naming conventions, or look for patterns in the database metadata.

**Examples** Use database commands to look at the database metadata:

```
select username from all_users

select table_name from user_tables

select index_name from user_indexes
```

# BP1249

**Statement** Do not use generic names for database objects such as databases, schema, users, tables, views, or indices.

**Rationale** Assigning generic names to user-defined objects within a database can lead to confusion and unexpected results. For example, naming a database “instance” within the *RDBMS* database is confusing to the humans who have to read commands that reference the database. In addition, the RDBMS software may parse *it* incorrectly.

Note: Although some RDBMS interpreters allow you to use a generic or reserved word to name objects if the name is surrounded with quotes, you should not do this either.

**Derived From** *BP1248*

**Justifies**

**Referenced By** *RDBMS internals*

**Acquisition Phase** Development

**Evaluation Criteria** **1. Test** *Are any generic names used for user-defined objects?*

**Procedure** Examine the RDBMS metadata for generic names such as database, table, entity, column, attribute, select, view, etc.

**Examples** `select table_name from user_tables where table_name in ('database', 'entity', ...)`

`select column_name from user_tab_columns where column_name in ('database', 'entity', ...)`

# BP1250

**Statement** Use case-insensitive names for database objects such as databases, schema, users, tables, views, and indices.

**Rationale** The *SQL* standard does not require names to be case-sensitive. Consequently, some DBMSs are not case-sensitive. Using case-sensitive names therefore makes portability more difficult.

**Derived From** *BP1248*

**Justifies**

**Referenced By** *RDBMS internals*

**Acquisition Phase** Development

**Evaluation Criteria**

1. **Test** *Are the names of database objects case-sensitive?*
- Procedure** Examine the database metadata for “run-on” names. If the database supports case-sensitive names, check to see if it is using camel-back capitalization.
- Examples** EMPLOYEEBENEFITSTBL  
EmployeeBenefitsTbl

# BP1251

<b>Statement</b>	Separate words with underscores.							
<b>Rationale</b>	The <i>SQL</i> standard does not require names to be case-sensitive. Consequently, some DBMSs are not case-sensitive. Using case-sensitive names therefore makes portability more difficult. To avoid these problems, use underscores to separate words (employee_benefits_tbl) rather than camel-back capitalization (EmployeeBenefitsTbl).							
<b>Derived From</b>	<i>BP1248</i>							
<b>Justifies</b>								
<b>Referenced By</b>	<i>RDBMS internals</i>							
<b>Acquisition Phase</b>	Development							
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Are underscores used between the words in the names of database objects?</i>						
	<b>Procedure</b>	Examine the database metadata and look for names that do not have underscores separating words.						
	<b>Examples</b>	<table> <tr> <td>EMPLOYEEBENEFITSTBL</td> <td><i>versus</i></td> <td>EMPLOYEE_BENEFITS_TBL</td> </tr> <tr> <td>EmployeeBenefitsTbl</td> <td><i>versus</i></td> <td>Employee_Benefits_Tbl</td> </tr> </table>	EMPLOYEEBENEFITSTBL	<i>versus</i>	EMPLOYEE_BENEFITS_TBL	EmployeeBenefitsTbl	<i>versus</i>	Employee_Benefits_Tbl
EMPLOYEEBENEFITSTBL	<i>versus</i>	EMPLOYEE_BENEFITS_TBL						
EmployeeBenefitsTbl	<i>versus</i>	Employee_Benefits_Tbl						

# BP1252

<b>Statement</b>	Do not use names with more than 30 characters.		
<b>Rationale</b>	Not all DBMSs support unlimited name lengths. For example, Oracle limits object names to 30 characters. Therefore, using names longer than 30 characters can reduce portability by limiting the DBMSs that the system can be deployed on.		
<b>Derived From</b>	<i>BP1248</i>		
<b>Justifies</b>			
<b>Referenced By</b>	<i>RDBMS internals</i>		
<b>Acquisition Phase</b>	Development		
<b>Evaluation Criteria</b>	<b>1.</b>	<b>Test</b>	<i>Are any of the database object names more than 30 characters in length?</i>
		<b>Procedure</b>	Examine the database metadata and look for names that are longer than 30 characters.
		<b>Examples</b>	<p>.....1.....2.....3.....4</p> <p>W2_EMPLOYEE_BENEFITS_FOR_FAMILIES_TBL</p>

# BP1253

**Statement** Do not use the *SQL:1999* or *SQL:2003* reserved words as names for database objects such as databases, schema, users, tables, views, or indices.

**Rationale** Using reserved words as the names of database objects can cause ambiguities and errors. It limits your ability to upgrade or port the code to other systems.

**Derived From** *BP1248*

**Justifies**

**Referenced By** *RDBMS internals*

**Acquisition Phase** Development

**Evaluation Criteria** 1. **Test** *Are any of the SQL:1999 or SQL:2003 reserved words used to name objects in the database?*

**Procedure** Examine the database metadata for names that are in the list of *SQL:1999* or *SQL:2003* reserved words

**Examples** Look for any of these words:

```
ABS ABSOLUTE ACCESS ACQUIRE ACTION ADA
ADD ADMIN AFTER AGGREGATE ALIAS ALL
ALLOCATE ALLOW ALTER AND ANY ARE ARRAY
AS ASC ASENSITIVE ASSERTION ASUTIME
ASYMMETRIC AT ATOMIC AUDIT
AUTHORIZATION AUX AUXILIARY AVG
```

```
BACKUP BEFORE BEGIN BETWEEN BIGINT
BINARY BIT BIT_LENGTH BLOB BOOLEAN BOTH
BREADTH BREAK BROWSE BUFFERPOOL BULK BY
```

```
CALL CALLED CAPTURE CARDINALITY CASCADE
CASCADED CASE CAST CATALOG CCSID CEIL
CEILING CHAR CHAR_LENGTH CHARACTER
CHARACTER_LENGTH CHECK CHECKPOINT CLASS
CLOB CLOSE CLUSTER CLUSTERED COALESCE
COLLATE COLLATION COLLECT COLLECTION
COLLID COLUMN COMMENT COMMIT COMPLETION
COMPRESS COMPUTE CONCAT CONDITION
CONNECT CONNECTION CONSTRAINT
CONSTRAINTS CONSTRUCTOR CONTAINS
CONTAINSTABLE CONTINUE CONVERT CORR
CORRESPONDING COUNT COUNT_BIG COVAR_POP
COVAR_SAMP CREATE CROSS CUBE CUME_DIST
CURRENT CURRENT_COLLATION CURRENT_DATE
CURRENT_DEFAULT_TRANSFORM_GROUP
CURRENT_LC_PATH CURRENT_PATH
CURRENT_ROLE CURRENT_SERVER
CURRENT_TIME CURRENT_TIMESTAMP
CURRENT_TIMEZONE
CURRENT_TRANSFORM_GROUP_FOR_TYPE
```

CURRENT\_USER CURSOR CYCLE

DATA DATABASE DATALINK DATE DAY DAYS  
 DB2GENERAL DB2SQL DBA DBCC DBINFO  
 DBSPACE DEALLOCATE DEC DECIMAL DECLARE  
 DEFAULT DEFERRABLE DEFERRED DELETE  
 DENSE\_RANK DENY DEPTH Deref DESC  
 DESCRIBE DESCRIPTOR DESTROY DESTRUCTOR  
 DETERMINISTIC DIAGNOSTICS DICTIONARY  
 DISALLOW DISCONNECT DISK DISTINCT  
 DISTRIBUTED DLNEWCOPY DLPREVIOUSCOPY  
 DLURLCOMPLETE DLURLCOMPLETEONLY  
 DLURLCOMPLETEWRITE DLURLPATH  
 DLURLPATHONLY DLURLPATHWRITE  
 DLURLSCHEME DLURLSERVER DLVALUE DO  
 DOMAIN DOUBLE DROP DSSIZE DUMMY DUMP  
 DYNAMIC

EACH EDITPROC ELEMENT ELSE ELSEIF END  
 END-EXEC EQUALS ERASE ERLVL ESCAPE  
 EVERY EXCEPT EXCEPTION EXCLUSIVE EXEC  
 EXECUTE EXISTS EXIT EXP EXPLAIN  
 EXTERNAL EXTRACT

FALSE FENCED FETCH FIELDPROC FILE  
 FILLFACTOR FILTER FINAL FIRST FLOAT  
 FLOOR FOR FOREIGN FORTRAN FOUND FREE  
 FREETEXT FREETEXTTABLE FROM FULL  
 FUNCTION FUSION

GENERAL GENERATED GET GLOBAL GO GOTO  
 GRANT GRAPHIC GROUP GROUPING

HANDLER HAVING HOLD HOLDLOCK HOST HOUR  
 HOURS

IDENTIFIED IDENTITY IDENTITY\_INSERT  
 IDENTITYCOL IF IGNORE IMMEDIATE IMPORT  
 IN INCLUDE INCREMENT INDEX INDICATOR  
 INITIAL INITIALIZE INITIALLY INNER  
 INOUT INPUT INSENSITIVE INSERT INT  
 INTEGER INTEGRITY INTERSECT  
 INTERSECTION INTERVAL INTO IS ISOBID  
 ISOLATION ITERATE

JAR JAVA JOIN

KEY KILL

LABEL LANGUAGE LARGE LAST LATERAL  
 LC\_CTYPE LEADING LEAVE LEFT LESS LEVEL  
 LIKE LIMIT LINENO LINKTYPE LN LOAD  
 LOCAL LOCALE LOCALTIME LOCALTIMESTAMP  
 LOCATOR LOCATORS LOCK LOCKSIZE LONG  
 LOOP LOWER

MAP MATCH MAX MAXEXTENTS MEMBER MERGE  
METHOD MICROSECOND MICROSECONDS MIN  
MINUS MINUTE MINUTES MOD MODE MODIFIES  
MODIFY MODULE MONTH MONTHS MULTISSET

NAME NAMED NAMES NATIONAL NATURAL NCHAR  
NCLOB NEW NEXT NHEADER NO NOAUDIT  
NOCHECK NOCOMPRESS NODENAME NODENUMBER  
NONCLUSTERED NONE NORMALIZE NOT NOWAIT  
NULL NULLIF NULLS NUMBER NUMERIC  
NUMPARTS

OBID OBJECT OCTET\_LENGTH OF OFF OFFLINE  
OFFSETS OLD ON ONLINE ONLY OPEN  
OPENDATASOURCE OPENQUERY OPENROWSET  
OPENXML OPERATION OPTIMIZATION OPTIMIZE  
OPTION OR ORDER ORDINARILITY OUT OUTER  
OUTPUT OVER OVERLAPS OVERLAY

PACKAGE PAD PAGE PAGES PARAMETER  
PARAMETERS **PART** PARTIAL PARTITION  
PASCAL PATH PCTFREE PCTINDEX PERCENT  
PERCENT\_RANK PERCENTILE\_CONT  
PERCENTILE\_DISC PIECESIZE PLAN POSITION  
POSTFIX POWER PRECISION PREFIX PREORDER  
PREPARE PRESERVE PRIMARY PRINT PRIOR  
PRIQTY PRIVATE PRIVILEGES PROC  
PROCEDURE PROGRAM PSID PUBLIC

QUERYNO

RAISERROR RANGE RANK RAW READ READS  
READTEXT REAL RECONFIGURE RECOVERY  
RECURSIVE REF REFERENCES REFERENCING  
REGR\_AVGX REGR\_AVGY REGR\_COUNT  
REGR\_INTERCEPT REGR\_R2 REGR\_SLOPE  
REGR\_SXX REGR\_SXY REGR\_SYY RELATIVE  
RELEASE RENAME REPEAT REPLICATION RESET  
RESIGNAL RESOURCE RESTORE RESTRICT  
RESULT RETURN RETURNS REVOKE RIGHT ROLE  
ROLLBACK ROLLUP ROUTINE ROW ROW\_NUMBER  
ROWCOUNT ROWGUIDCOL ROWID ROWNUM ROWS  
RRN RULE RUN

SAVE SAVEPOINT SCHEDULE SCHEMA SCOPE  
SCRATCHPAD SCROLL SEARCH SECOND SECONDS  
SECQTY SECTION SECURITY SELECT  
SENSITIVE SEQUENCE SESSION SESSION\_USER  
SET SETS SETUSER SHARE SHUTDOWN SIGNAL  
SIMILAR SIMPLE SIZE SMALLINT SOME  
SOURCE SPACE SPECIFIC SPECIFICTYPE SQL  
SQLCA SQLCODE SQLERROR SQLEXCEPTION  
SQLSTATE SQLWARNING SQRT STANDARD START  
STATE STATEMENT STATIC STATISTICS STAY  
STDDEV\_POP STDDEV\_SAMP STOGROUP STORES  
STORPOOL STRUCTURE STYLESUBPAGES

SUBSTRING SUCCESSFUL SUM SYMMETRIC  
SYNONYM SYSDATE SYSTEM SYSTEM\_USER

TABLE TABLESPACE TEMPORARY TERMINATE  
TEXTSIZE THAN THEN TIME TIMESTAMP  
TIMEZONE\_HOUR TIMEZONE\_MINUTE TO TOP  
TRAILING TRAN TRANSACTION TRANSLATE  
TRANSLATION TREAT TRIGGER TRIM TRUE  
TRUNCATE TSEQUAL TYPE

UID UNDER UNDO UNION UNIQUE UNKNOWN  
UNNEST UNTIL UPDATE UPDATETEXT UPPER  
USAGE USE USER USING

VALIDATE VALIDPROC VALUE VALUES VAR\_POP  
VAR\_SAMP VARCHAR VARCHAR2 VARIABLE  
VARIANT VARYING VCAT VIEW VOLUMES

WAITFOR WHEN WHENEVER WHERE WHILE  
WIDTH\_BUCKET WINDOW WITH WITHIN WITHOUT  
WLM WORK WRITE WRITETEXT

YEAR YEARS

ZONE

## BP1254

<b>Statement</b>	For command-and-control systems, use the names defined in the <i>C2IEDM</i> for data exposed to the outside communities.	
<b>Rationale</b>	The <i>Command and Control (C2) COI</i> has developed a <i>data model</i> to facilitate the exchange of data within the community and by consumers of their data outside the community. Therefore, data that is to be exposed from the database to the COI community or its data consumers should defer to the data model whenever possible. The data model defines the data units as well as the names and structure of the data.	
<b>Derived From</b>	<i>BP1248</i>	
<b>Justifies</b>		
<b>Referenced By</b>	<i>RDBMS internals</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>If this is a C2 system, does it use C2IEDM data elements for the data that is exposed to the outside world?</i>
	<b>Procedure</b>	Review all the data that is exposed to the outside world and confirm that it conforms to the C2IEDM specifications.
	<b>Examples</b>	None.

# BP1255

<b>Statement</b>	Use surrogate keys.
<b>Rationale</b>	<p>A surrogate key, also referred to as a system-generated key, database-sequence number, or arbitrary unique identifier, is a unique, arbitrary <i>primary key</i>. It is usually generated by the <i>RDBMS</i>, but can also be generated by a database access layer such as the middle tier. It is arbitrary because it is not derived from any data that exists within the table or the database. Some other options for surrogate keys are:</p> <p>Universally Unique Identifiers (UUIDs) (<a href="http://en.wikipedia.org/wiki/Universally_Unique_Identifier">http://en.wikipedia.org/wiki/Universally_Unique_Identifier</a>)</p> <p>Globally Unique Identifiers (GUIDs) (<a href="http://en.wikipedia.org/wiki/Globally_Unique_Identifier">http://en.wikipedia.org/wiki/Globally_Unique_Identifier</a>)</p>
<b>Derived From</b>	
<b>Justifies</b>	<i>BP1256, BP1257</i>
<b>Referenced By</b>	<i>RDBMS internals</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel guidance for evaluation criteria.

# BP1256

**Statement** Use surrogate keys as the *primary key*.

**Rationale** Instead of using the natural keys to uniquely identify each record, use a surrogate key. This allows the natural key information to be modified independently of the primary key and any foreign-key references to the key.

**Derived From**

**Justifies**

**Referenced By** *RDBMS internals*

**Acquisition Phase** Development

**Evaluation Criteria** 1. **Test** *Are surrogate keys used instead of natural keys?*

**Procedure** Look at the database metadata and determine if it uses surrogate or natural keys.

**Examples** The following example shows natural keys. The primary keys are made up completely or in part from naturally occurring data in the tables.

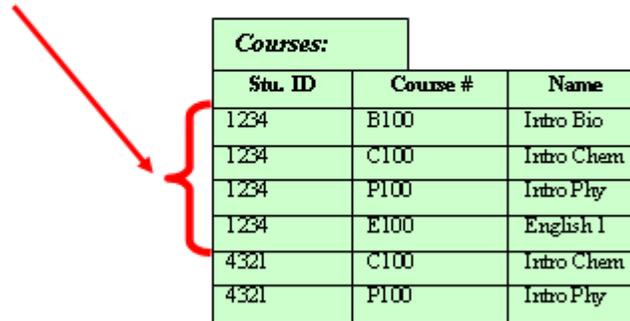
<i>Students:</i>			<i>Natural Keys</i>		
Name	Address	Phone	Name	Course #	Name
John Public	200 Ash St, Hometown, USA	800-555-1234	Jane Doe	B100	Intro Bio
Jane Doe	170 Elm Ave, Hometown, USA	800-555-1212	Jane Doe	C100	Intro Chem
			Jane Doe	P100	Intro Ply
			Jane Doe	E100	English I
			John Public	C100	Intro Chem
			John Public	P100	Intro Ply

If the student name "Jane Doe" changes, all occurrences of the name must be changed.

The following example shows a surrogate key being used instead of a natural key. Maintaining data is less complex than it is with natural keys and consequently less error-prone.

**Students:** *Surrogate Keys*

Stu. ID	Name	Address	Phone
4321	John Public	200 Ash St, Hometown, USA	800-555-1234
1234	Jane Doe	170 Elm Ave, Hometown, USA	800-555-1212



**Courses:**

Stu. ID	Course #	Name
1234	B100	Intro Bio
1234	C100	Intro Chem
1234	P100	Intro Phy
1234	E100	English 1
4321	C100	Intro Chem
4321	P100	Intro Phy

If the student name "Jane Doe" changes, only one occurrence of the name must be changed.

# BP1257

<b>Statement</b>	Place a unique key constraint on the natural key fields or secondary key.	
<b>Rationale</b>	Surrogate keys make it easier to maintain data. However, a column or set of columns should still uniquely identify the row in the table. This column or set of columns is the “natural key” or “secondary key.” This natural key should still be protected by the uniqueness constraint normally associated with a <i>primary key</i> .	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	<i>RDBMS internals</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Is there a unique key index for all tables that includes a column or set of columns not including the primary key?</i>
	<b>Procedure</b>	Look at the database metadata to ensure that each table has a unique key, and that the columns in the unique key are not also part of the primary key.

# BP1258

<b>Statement</b>	All data transferred via <i>XML</i> should explicitly define the encoding style.	
<b>Rationale</b>	By default, XML is encoded using <i>Unicode</i> . Consequently, data transferred via XML should explicitly specify the encoding style. Assuming the default can cause <i>interoperability</i> problems between implementations. For example, the ASCII coding style is: {insert}	
<b>Derived From</b>		
<b>Justifies</b>		
<b>Referenced By</b>	<i>RDBMS internals</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>{OPTIONAL: Place a question here that can be used to evaluate the best practice statement. One question per test. If there are multiple questions required to evaluate a guidance statement, indicate if the tests are to be “And’d” or “Or’d”}</i>
	<b>Procedure</b>	<i>{OPTIONAL: Place the procedure to evaluate the test question here. The procedure can be multiple steps}</i>
	<b>Examples</b>	Look for the following XML tag as the first line returned from queries that return XML from the database.  <code>&lt;?xml version="1.0" encoding="UTF-8"?&gt;</code>

## BP1259

<b>Statement</b>	Use indexes.
<b>Rationale</b>	<p>An index in an <i>RDBMS</i> is a summary of information organized to minimize the search time. Indexes summarize the information in a table. So, an employee table might have an index of last names, or last name and first name.</p> <p>Having additional indexes on tables involves a tradeoff between query performance and insert/update/delete performance, which requires underlying index maintenance.</p>
<b>Derived From</b>	
<b>Justifies</b>	<i>BP1260, BP1261, BP1262</i>
<b>Referenced By</b>	<i>RDBMS internals</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	See sublevel guidance for evaluation criteria.

# BP1260

<b>Statement</b>	All tables should have a <i>primary key</i> , which is generally enforced via an underlying index.	
<b>Rationale</b>	By definition, a primary key uniquely defines each row within a table. To optimize the use of the table and to find records by the primary key, there should be an index that enforces the uniqueness of the key.	
<b>Derived From</b>	<i>BP1259</i>	
<b>Justifies</b>		
<b>Referenced By</b>	<i>RDBMS internals</i>	
<b>Acquisition Phase</b>	Development	
<b>Evaluation Criteria</b>	<b>1. Test</b>	<i>Is there a primary key defined for each table listed in the database?</i>
	<b>Procedure</b>	Examine the database metadata to ensure there is a primary key for each table in the database.

## BP1261

<b>Statement</b>	Monitor and tune indexes according to the response time during normal operations in the production environment.
<b>Rationale</b>	<p>Index efficiency depends on the data being indexed. Common variables include:</p> <ul style="list-style-type: none"><li>• A sparsely populated table versus a densely populated table</li><li>• Data added in an presorted order versus a random order</li></ul> <p>Consequently, as the data changes, the efficiency of the index changes.</p>
<b>Derived From</b>	<i>BP1259</i>
<b>Justifies</b>	
<b>Referenced By</b>	<i>RDBMS internals</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	There are no tests to determine if the database has been monitored and tuned accordingly.

# BP1262

**Statement** In the case of Oracle, define indexes against the FK columns to avoid contention and locking issues.

**Rationale**

**Derived From** *BP1259*

**Justifies**

**Referenced By** *RDBMS internals*

**Acquisition Phase** Development

**Evaluation Criteria** None

## BP1263

<b>Statement</b>	Gather storage requirements in the planning phase, and then allocate twice the estimated storage space.
<b>Rationale</b>	Storage space on the disk always poses a problem for databases, so it is necessary to plan storage space carefully.
<b>Derived From</b>	
<b>Justifies</b>	
<b>Referenced By</b>	<i>RDBMS internals</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	None

# BP1264

<b>Statement</b>	For <i>high availability</i> , use hardware solutions when geographic proximity permits.
<b>Rationale</b>	There are many ways to achieve high availability. Some are based on hardware and others on software. As a general rule, hardware solutions use simple redundancy and are consequently less complex and fragile. If geographic proximity is not an issue, the hardware solution is preferable.
<b>Derived From</b>	
<b>Justifies</b>	
<b>Referenced By</b>	<i>RDBMS internals</i>
<b>Acquisition Phase</b>	Development
<b>Evaluation Criteria</b>	There are no tests for this best practice.

# BP1265

<b>Statement</b>	<i>XML</i> validation is the responsibility of the XML document generator.						
<b>Rationale</b>	All XML passed between two systems or services must be valid. The XML document generator is responsible for ensuring that the document is valid and <i>well-formed</i> . If there are problems, the document generator is the only user that can effectively change the document.						
<b>Derived From</b>							
<b>Justifies</b>							
<b>Referenced By</b>	<i>Parsing XML strategies</i>						
<b>Acquisition Phase</b>	Development						
<b>Evaluation Criteria</b>	<table border="0" style="margin-left: 20px;"> <tr> <td style="vertical-align: top;"><b>1. Test</b></td> <td><i>Are all the XML documents exported from the system or service valid and well-formed?</i></td> </tr> <tr> <td style="vertical-align: top;"><b>Procedure</b></td> <td>Capture all the documents and validate them, using a product similar to XMLSpy.</td> </tr> <tr> <td style="vertical-align: top;"><b>Examples</b></td> <td>None</td> </tr> </table>	<b>1. Test</b>	<i>Are all the XML documents exported from the system or service valid and well-formed?</i>	<b>Procedure</b>	Capture all the documents and validate them, using a product similar to XMLSpy.	<b>Examples</b>	None
<b>1. Test</b>	<i>Are all the XML documents exported from the system or service valid and well-formed?</i>						
<b>Procedure</b>	Capture all the documents and validate them, using a product similar to XMLSpy.						
<b>Examples</b>	None						

---

# Appendices

This section contains information on the following topics:

- *References*
- *Testing*
- *Namespace management procedures*
- *Mobile code*
- *Java developer programs*
- *Navy-specific guidelines*
- *Cross-reference between NESI and other initiatives*
- *Open-source tools*

# Technical References

## Books

### Design patterns

Alur, Deepak, John Crupi and Dan Malks. *Core J2EE Patterns: Best Practices and Design Strategies*

ISBN:0131422464.

Douglass, Bruce Powel. *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*. ISBN 0201699567.

Fowler, Martin. *Analysis Patterns: Reusable Object Models*. ISBN 0201895420

Fowler, Martin. *Patterns of Enterprise Application Architecture*. ISBN 0321127420

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Object-Oriented Software*. ISBN 0201633612

Monday, Paul B. *Web Services Patterns: Java Edition*. ISBN 1590590848

Pattern languages of programming design (4 volumes):

Coplien, James O. and Douglas C. Schmidt. *Pattern Languages of Programming Design*. ISBN 0201607344

Vlissides, John M., James O. Coplien, Norman L. Kerth, and Norman Kerth. *Pattern Languages of Programming Design 2*. ISBN 0201895277

Martin, Robert C., Dirk Riele, and Frank Buschman. *Pattern Languages of Programming Design 3*. ISBN 0201310112

Harrison, Neil, Brian Foote, and Hans Rohnert. *Pattern Languages of Programming Design 4*. ISBN 0201433044

Pattern-oriented software architecture (3 volumes):

Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. ISBN 0471958697

Schmidt, Douglas, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects*. ISBN 0471606952

Kircher, Michael and Prashant Jain. *Pattern-Oriented Software Architecture, Patterns for Resource Management*. ISBN 0470845252

### Design and usability

Krug, Steve. *Don't Make Me Think: A Common Sense Approach to Web Usability*. ISBN 0789723107.

Nielsen, Jakob. *Designing Web Usability: The Practice of Simplicity*. ISBN 156205810X.

Sather, Andrew, Ardith Ibanez, and Bernie Dechant. *Creating Killer Interactive Web Sites: The Art of Integrating Interactivity and Design*. ISBN 1568303734.

## Frameworks

Fayad, Mohamed E. and Ralph E. Johnson (eds). *Domain-Specific Application Frameworks: Frameworks Experience by Industry*. ISBN 0471332801

Fayad, Mohamed E., Douglas Schmidt, and Ralph E. Johnson (eds). *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. ISBN 0471248754

Fayad, Mohamed E., Douglas Schmidt, and Ralph E. Johnson (eds). *Implementing Application Frameworks: Object-Oriented Frameworks at Work*. ISBN 0471252018

## Microsoft

Ballinger, Keith. *.NET Web Services: Architecture & Implementation*. ISBN 0321113594

Chappell, David. *Understanding Microsoft Windows 2000 Distributed Services*. ASIN 157231687X

Chappell, David. *Understanding .NET*. ISBN 0201741628

Chen, Xin. *BizTalk Server 2002: Design and Implementation*. ISBN 1590590341

Freeman, Adam and Allen Jones. *Microsoft .NET XML Web Services: Step By Step*. ISBN 0735617201

Guest, Simon. *Microsoft .NET and J2EE Interoperability Toolkit*. ISBN 0735619220

Honeycutt, Jerry. *Introducing Microsoft Windows Server 2003*. ISBN 0735615705

Kanalakis, John. *Developing .NET Enterprise Applications*. ISBN 1590590465

Kemp, Christine, Richard Kemp, and Marcus Goncalves. *Designing Enterprise Solutions With Microsoft Technologies*. ASIN 013086756X

Lowe-Norris, Alistair G. *Windows 2000 Active Directory*. ISBN 0596004664

MacDonald, Matthew. *Microsoft .NET Distributed Applications: Integrating XML Web Services and .NET Remoting*. ISBN 0735619336

Mohr, Stephen. *Designing Distributed Applications with XML, ASP, IE5, LDAP and MSMQ*. ISBN 1861002270

Peltzer, Dwight. *.NET and J2EE Interoperability*. ISBN 0072230541

Platt, David S. *Understanding COM+: The Architecture for Enterprise Development Using Microsoft Technologies*. ASIN 0735606668

Sessions, Roger. *COM+ and the Battle for the Middle Tier*. ASIN 0471317179

Sharma, Chetan. *Wireless Internet Enterprise Applications*. ASIN 0471393827

Short, Scott. *Building XML Web Services for the Microsoft .NET Platform*. ASIN 0735614067

Stanek, William R. *Microsoft SQL Server 2000: Administrators Pocket Consultant*. ISBN 0735611297

**Process**

Ezran, Michel, Maurizio Morisio, Colin Tully, and C. J. Tully. *Practical Software Reuse*. ISBN 1852335025.

Reifer, Donald J. *Practical Software Reuse*. ISBN 0471578533.

**Other**

Foster, Lonnon R. *Palm OS Programming Bible*. ISBN 0764549618.

**Web sites****Commercial**

Topic or Group	Site
Application Development magazine	<a href="http://www.appdevadvisor.co.uk/">http://www.appdevadvisor.co.uk/</a>
ASN (Abstract System Notation) and XML	<a href="http://asn1.elibel.tm.fr/xml/">http://asn1.elibel.tm.fr/xml/</a>
IT white papers, web casts, and case studies	<a href="http://www.itpapers.com/">http://www.itpapers.com/</a>
Java technologies, latest releases	<a href="http://java.sun.com/products/index.html">http://java.sun.com/products/index.html</a>
J2EE	<a href="http://java.sun.com/j2ee">http://java.sun.com/j2ee</a>
Model-driven architecture	<a href="http://www.omg.org/mda">http://www.omg.org/mda</a>
.NET resources	<a href="http://microsoft.com/net/">http://microsoft.com/net/</a>
Objective Technology Group	<a href="http://www.theotg.com/">http://www.theotg.com/</a>
Object Management Group	<a href="http://www.omg.org">http://www.omg.org</a>
Public NMCI web site	<a href="http://www.nmci-isf.com">http://www.nmci-isf.com</a>
UDDI	<a href="http://www.uddi.org">http://www.uddi.org</a>
Web services	<a href="http://webservices.org">http://webservices.org</a> <a href="http://www.xml.com/pub/a/2001/04/04/webservices">http://www.xml.com/pub/a/2001/04/04/webservices</a>
WSDL	<a href="http://www.w3.org/tr/wsdl">http://www.w3.org/tr/wsdl</a>

WS-I	<a href="http://www.ws-i.org">http://www.ws-i.org</a>
XACML	<a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml">http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml</a>

### Government

Topic or Group	Site
Cookie/privacy policy	<a href="http://www.c3i.osd.mil/org/cio/doc/cookies.html">http://www.c3i.osd.mil/org/cio/doc/cookies.html</a>
DoD mobile code policy	<a href="http://www.dod.mil/nii/org/cio/doc/mobile-code11-7-00.html">http://www.dod.mil/nii/org/cio/doc/mobile-code11-7-00.html</a>
DoD PKI policy	<a href="http://www.c3i.osd.mil/org/cio/doc/may172001.pdf">http://www.c3i.osd.mil/org/cio/doc/may172001.pdf</a>
DoD web content policy	<a href="http://www.defenselink.mil/webmasters">http://www.defenselink.mil/webmasters</a>
INFOSEC	<a href="http://infosec.navy.mil">http://infosec.navy.mil</a>
Section 508 Compliance	<a href="http://www.section508.gov">http://www.section508.gov</a>

### Microsoft

Site	Description
<a href="http://www.microsoft.com/technet/prodtechnol/">http://www.microsoft.com/technet/prodtechnol/</a>	TechNet is an information and community resource for IT professionals. The TechNet program includes the TechNet web site on various .NET Product Technologies and also includes technical briefings, events, and webcasts.
<a href="http://msdn.microsoft.com/library/">http://msdn.microsoft.com/library/</a>	The Microsoft Developer Network (MSDN) is a set of online and offline services for developers using Microsoft products and technologies. It includes the .NET Code Wise Community ( <a href="http://www.gotdotnet.com/content/codewise/default.aspx">http://www.gotdotnet.com/content/codewise/default.aspx</a> ).
<a href="http://msdn.microsoft.com/webservices/">http://msdn.microsoft.com/webservices/</a>	This site provides help with web services enhancements for Microsoft .NET (WSE). It is a supported add-on to Microsoft Visual Studio .NET and the Microsoft .NET Framework. It provides the latest web-services capabilities to keep pace with the evolving web services protocol specifications.
<a href="http://searchwebservices.techtarget.com/">http://searchwebservices.techtarget.com/</a>	This site provides web services and XML developer tips vis-à-vis .NET and J2EE technologies. There was a great article in June of 2004 by Peter Aiken on ".NET Tools for Working with XM" that was

	used for this paper.
<a href="http://www.theserverside.net/">http://www.theserverside.net/</a>	This site provides news, discussions, technical articles, interactive chats, and case studies on .NET technologies. These range from ADO.NET, ASP.NET, the .NET Compact Framework to web services specifications and XML tools.
<a href="http://www.franklins.net/dotnetrocks">http://www.franklins.net/dotnetrocks</a>	Carl Franklin and Rory Blyth interview experts to bring you insights into .NET technology and the state of software development. Especially helpful for ASP.NET and custom controls section of the paper.
<a href="http://www.code-magazine.com">http://www.code-magazine.com</a>	CoDe (Component Developer) Magazine, written by .NET developers for .NET developers, is one of the favorite magazines for developers involved in Microsoft technologies. In-depth articles with practical code samples satisfy the search for great technical information. Each bi-monthly issue contains detailed explanations of Visual Studio .NET and the .NET Framework.
<a href="http://www.gotdotnet.com/">http://www.gotdotnet.com/</a>	This site includes the .NET Code Wise Community ( <a href="http://www.gotdotnet.com/content/codewise/default.aspx">http://www.gotdotnet.com/content/codewise/default.aspx</a> ). It provides good examples of ASP.NET and ADO.NET and some interesting tools to use within a .NET development environment.
<a href="http://www.dotnetjohn.com/">http://www.dotnetjohn.com/</a>	This site provides good examples of both ASP.NET and ADO.NET. It shares .NET programming information with other developers.
<a href="http://www.developer.com/net/">http://www.developer.com/net/</a>	This site shares .NET programming information with other developers and provides good examples of ASP.NET, XML and web services, and C#.
<a href="http://www.informit.com/">http://www.informit.com/</a>	This site shares .NET programming information with other developers and provides good examples of ASP.NET, SQL Server, LDAP (Active Directory), and C#.
<a href="http://www.15seconds.com/">http://www.15seconds.com/</a>	This site shares .NET programming information with other developers and provides good examples of ASP.NET, SQL Server, MSMQ, ADSI, C#, and ADO.NET.
<a href="http://www.schema.net/">http://www.schema.net/</a>	This site was part of a family of XML-related sites started in 1997. It was the first site dedicated to cataloging XML DTDs and schemas. It was actively maintained until around 2000 and may become

	active again soon.
<a href="http://xml.com/">http://xml.com/</a>	This site provides good examples of XML and web services that apply to both .NET and J2EE platforms.
<a href="http://www.JNBridge.com/">http://www.JNBridge.com/</a>	This is a third-party vendor site dedicated to an interoperability bridge between the J2EE platform and the .NET platform.
<a href="http://j-integra.intrinsyc.com/">http://j-integra.intrinsyc.com/</a>	This site is a third-party vendor site dedicated to an interoperability bridge between the J2EE platform and the .NET platform.

Automated testing increases the speed at which applications are delivered by creating standardized testing, which minimizes errors and promotes reusability of components and services in various deployment environments.

## Automated testing tools

Some examples of automated tools include:

- WinRunner (<http://www.mercury.com/us/>)
- JUNIT (<http://www.junit.org/index.htm>)
- Rational Robot (<http://www.rational.com>)
- Compuware QARun (<http://www.compuware.com>)
- Empirix eTest (<http://www.empirix.com>)
- SilkTest (<http://www.segure.com>)

# Environments

Consider the following environments when testing your applications:

<i>Enterprise and web application environments</i>	BEA Web Logic 7.1 Sun ONE Release 7 JBoss Tomcat/Apache
<i>Operating systems</i>	Win 2K Win XP Solaris 2.8
<i>Browsers</i>	Internet Explorer 5.5 and above

## Security testing tools

Security testing must meet *COE* I&RTS security requirements. To meet these requirements, use the following *DISA* security tools:

- UNXSCP\_1.2.0.0 (Solaris)
- WINSXP\_1200 (Win2K)

# Namespace management procedures

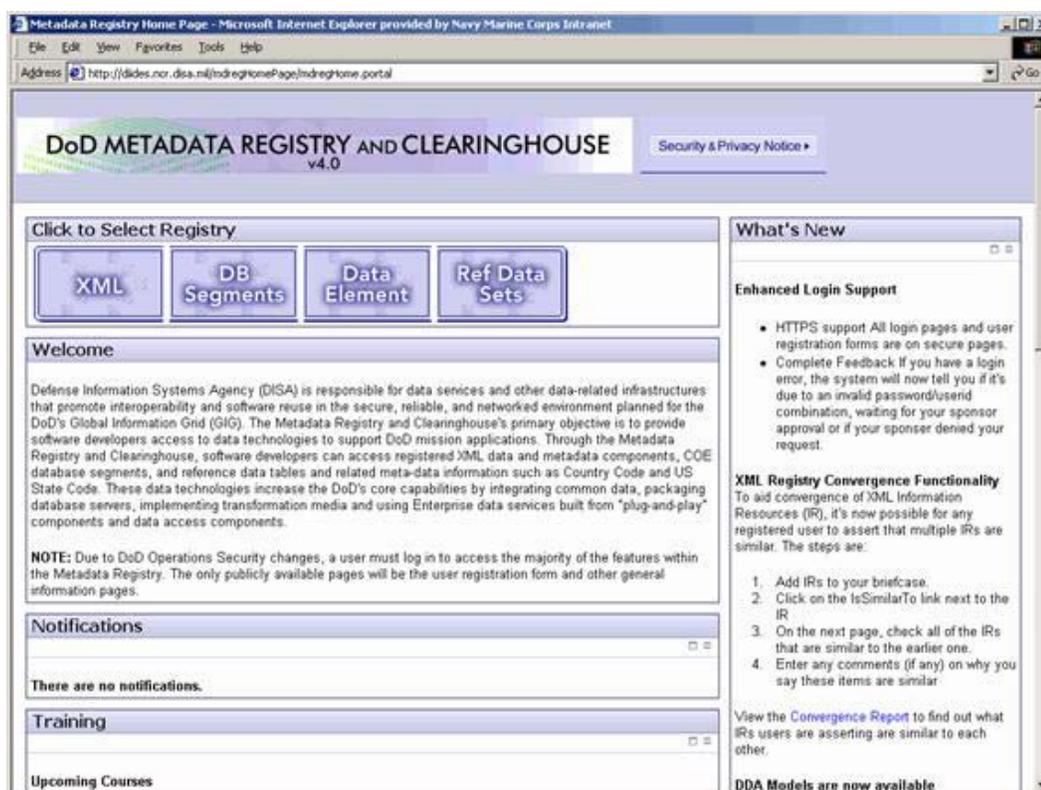
The following process outlines how to retrieve available application or service names from the DoD Metadata Registry and Clearinghouse. Developers should contact the appropriate *namespace* manager to check for exceptions to this process.

## To determine available application or service names:

1. Log in to the DoD Metadata Registry and Clearinghouse web site at <http://diides.ncr.disa.mil/>.

**Note:** Namespace management procedures are undergoing change at this time. Updated procedures will be available on this web site.

2. Click the **XML** button in the **Click to Select Registry** section.



3. Click **Login** to log in to the XML Registry.
  - If you already have a username and password, you can log in from this page.
  - If you do not have an account, click the **Registration** link to request one. A military or government sponsor is required to validate your account request.

Once you log in, you will be returned to the XML Registry page.

4. Click the **Show me the Namespaces** link. You can view the general list of namespaces without logging in, but you will not be able to view any details about the namespaces until you log in.



5. Click the **Namespace** link for the functional area of your application. You will need a valid login to view namespace information details.

DoD XML Registry - Namespace Information - Microsoft Internet Explorer provided by Navy Marine Corps Intranet

Address: http://dides.ncr.dsa.mil/xmlreg/user/namespace.cfm?namespace\_id=17

**DoD METADATA REGISTRY AND CLEARINGHOUSE**

DoD XML Registry  
Session validated for Cara Zylla [Logout](#)

**Namespace Information**

**Abbreviation:** MET  
**Namespace:** Meteorological and Oceanographic  
**Network:** NIPRNET  
**Status:** Approved  
**Manager:** [Tom Nabors](#)  
**E-Mail:** [tn52@crnac.navy.mil](mailto:tn52@crnac.navy.mil)  
[Business Rules for Statuses of IRs in this Namespace](#)

**View Information Resources for Meteorological and Oceanographic Namespace using the selection criteria below.**

Word or Phrase:

Fields: All

Submitter: All

Status: All

Info Resource Type: All

Version: All

Effective Date: Start Date: 11/17/1998 End Date: 08/20/2003

6. Search the XML registry to verify that the application/service name does not already exist. To do this, use the form at the bottom of the Namespace page to create a query that filters the information resources for the selected namespace.

Queries can determine existing XML elements, attributes, and schemas as well as packages and domain values. This can help you determine a unique application or service name and also those pieces complimentary to the service being developed.

7. Click the email link to contact the appropriate namespace manager to confirm the name acquisition and update the appropriate repository. The email must contain the following information:
  - Point of contact (POC) information
  - Program name
  - Application name
  - Namespace selected

**Note:** This site is also available on the SIPRNET and JWICS.

## Mobile code

There are two documents for mobile code guidance:

- The Mobile Code Policy document. This guidance does not cover mobile devices. You can download the policy document from <http://www.dod.mil/nii/org/cio/doc/mobile-code11-7-00.html>. To protect DoD systems from malicious or improper use of mobile code, developers must assess and control the risks. The DoD Mobile Code Policy should be the first step in an iterative process to reduce these risks. It categorizes mobile code technologies and restricts their application based on their potential to cause damage if used maliciously.
- For additional policy guidance and usage restrictions, see Assistant Secretary of Defense (C3I) Memorandum, Subject: "Policy Guidance for use of Mobile Code Technologies in Department of Defense (DOD) Information Systems," 7 November 2000.

## Java developer programs

Sun has three developer programs that may be of interest to *NESI* developers:

- **Java.net:** An open-source program that doesn't require a login.
- **Sun Developer's Network:** Provides access to Sun's services, product downloads, and community support forums.
- **Java Community Process:** A site with news about Java development, training, and events.

### To access Java.net:

1. Go to *http://java.sun.com*.
2. Click **java.net** under **Sun Resources**.

### To register for the Sun Developer's Network:

1. Go to *http://java.sun.com*.
2. Click **Join a Sun Developer Network Community**. A registration page appears. Follow the on-screen prompts.

### To access the Java Community Process:

1. Go to *http://java.sun.com*.
2. Click **Java Community Process** under **Sun Resources**.

### To get the latest Java environment:

1. Go to *http://java.sun.com/j2se/1.4.2/download.html* and download either:
  - The **Java Runtime Environment (JRE)**, which allows end-users to run Java applications
  - The **Java Software Developer's Kit (SDK)**, which allows you to create J2SE applications and also contains the JRE (recommended)
2. Install the software, using the following options:
  - Install the Java Runtime Environment in your target directory, along with all other open-source products.
  - Make sure to register your browser(s), so they can use Java plug-ins.  
**Note:** For all other options, use the default settings or the settings you know to be appropriate.

## Navy-specific guidelines

This appendix lists tools and information that are specific to the Navy.

### COE-M build lists

#### Solaris build list

*Global Command and Control Systems Maritime (GCCS-M)* Integration Product Standard Flash Load Build 11.1 (Solaris 8) Version 3.0, dated 20 May 2004.

Segments and Configuration	Prefix	Version #
Solaris Operating System		2.8 7/03
Solstice DiskSuite		4.2.1
Solaris Software Companion CD		2/02
Solaris 8 Software Supplement CD		7/03
DII COE Kernel		4.2.0.9
DII COE Kernel Patch		Patch 11 Beta 4
Replace the <i>libAPM.so</i> File		
Solaris Security Patch Update		Update
Java Platform 2	JAVA2	4.7.0.1
J2SE JRE 1.4.1 _03	J2JRE	4.7.1.0
Netscape Browser	NSWEB	4.7.0.1
Perl	PERL	4.2.0.1
Adobe Acrobat Reader	ACRORD	4.7.1.0
SecurityPolicyConfigurationTool	SPCFG	4.9.0.0
COE Update System Security Level	UPDTSL	4.7.0.0 Beta
COE Security Banner	SECBNR	4.6.0.0
DII COE System Menu Interface	SMB	4.0.0.8
OnlineDocs	ONDOC	4.2.1.0

NSS Libraries	NSSLIB	4.7.1.0 Beta
COE Security Services	COESS	4.7.1.0 Beta
SSAF Application	SSAFAP	4.7.1.0 Beta
TCP Wrappers	TCPW	4.7.0.0
VirusScan for UNIX	VSCANU	4.7.0.0
VSCANU Update Fix		
Crack	CRACK	4.7.0.0
Swatch-M	SWATCM	3.2.3.0
SPCFG Maritime Segment	SPCFGM	1.0.0.6
SPCFGM Patch 1	SPCFGM	1.0.0.6P1
JMTK Utilities Segment	JMU	4.7.0.1
ICSF Bundle (Note 2)	ICSF	4.5.2.0
IFLFIX Segment	IFLFIX	1.0.0.3
ICSFPatchP5Bundle (Note 3)	ICSF	4.5.2.0P5
JMTK-V Map Data	JMVMD	4.5.2.0
JMTK SDBM	JMS	4.7.0.1
JMTK Analysis	JMA	4.7.0.1
JMS Draw Module	JDM	4.7.0.1
BEA WebLogic Server	BEAWLS	4.7.1.1
BEA Client-Side JAR File	BEAJAR	4.7.1.0
XVFB	XVFB	1.0.0.0
I3 Configure Middle Tier	I3CMT	4.7.1.4
Modular Embed Doc Utility Arch	MEDULA	4.7.1.3
Standard Int On-site Present Sys	SINOPS	4.7.1.3
Composite System Level Doc Data Navy	CSLDDN	4.7.1.4
C4I Common Extensions	CCE	4.5.5.0

C4I Maritime Extensions	CME	4.5.5.1
Air Tasking Exchange Runtime	ATXRUN	4.7.0.1
Tadil-A/B Interface	Link11	4.5.2.8
Extensible Information Model	XIM	4.5.3.0
Extensible Information Views	XIV	4.5.3.0
Extensible Data Source Interface	XDSI	4.5.3.0
Extensible Web Tier Support	XWEB	4.5.3.0
Extensible Chart Integration	XCI	4.5.3.0
XIM Patch 2	XIM	4.5.3.0P2
XIV Patch 2	XIV	4.5.3.0P2
XDSI Patch 2	XDSI	4.5.3.0P2
XWEB Patch 2	XWEB	4.5.3.0P2
XCI Patch 2	XCI	4.5.3.0P2
CCE Network Time Protocol	CNTP	4.5.2.0
DII COE Print Services Server	PRINTS	4.7.1.0 Beta 2
DII COE Print Services Drivers	PRINTD	4.7.1.0 Beta 2
DII COE Print Services Client	PRINTC	4.7.1.0 Beta 2
NIS+ Admin Tool	NISADM	4.4.0.0
TCLTK	TCLTK	4.0.0.0
ZIRCON Internet Relay Chat Clt	ZIRCC	4.0.0.2
ZIRCON Internet Relay Chat Svr	ZIRCS	4.0.0.1
MTC/TADIL J Interface	MTC	4.5.8.5
TBM Computer-Based Training	TBMCBT	4.6.0.0
TBM Warning and Display	TBMWD	6.3.1.1
Intelligence Shop Interfaces	ISHOPI	4.7.1.2
Intelligence Shop Client-Tier	ISHOPC	4.7.1.3

Sybase System SW Asset Mgmt	SYSAM	1.0.1.0
Sybase Adaptive Svr Enterprise	SYBADP	12.5.0.0
Sybase Adaptive Svr Enterprise Patch 1	SYBADP	12.5.0.0P1
Sybase ASE Config for I3	SYBI3C	4.7.1.2
Database Administrator Runtime	DBAdmR	4.0.0.0
Database Administrator Server	DBAdmS	4.0.0.0
Sybase Administration	SYBADM	4.7.1.2
NITFS Services	NITFS	4.7.2.0
Image Transformation Services	IMX	4.7.2.0
ITS Client	ITS	4.7.2.0
Automated Image Import Module	AIIM	4.7.2.0
Universal Data Import and Export	UDIE	4.7.2.0
Java Image Video Exploitation	JIVE	4.7.2.0
Intelligence Shop Middle-Tier	ISHOPM	4.7.1.2
ITSWEB	ITSWEB	4.7.2.0
Alerts Services Client Runtime	ALTCLT	4.5.2.5
CAPL Framework Segment	CAPFW	4.2.0.1
CAPL COE Clients Segment	CAPCC	4.2.0.1
CAPL ICSF Clients Segment	CAPIC	4.2.0.1
DII COE Message Processor	CMP	4.4.0.0
Waterspace DB Layer	WSMDFL	4.5.2.5
Waterspace Message Services	WSMMS	4.5.2.5
Waterspace Visualization	WSMV	4.5.2.5
Waterspace Plotting	WSMPLT	4.5.2.5
Waterspace PMI Server	WSMPMI	4.5.2.5
Waterspace Voyage Monitor	WSMVM	4.5.2.5

Space Common Tactical Dataset	SCTD	4.1.0.0
GALE-Lite Interface Segment	GLIS	4.5.4.0
Joint MTI Client	JMTIC	4.0.2.0
Audit Viewer	AUDVWR	4.0.0.1
Global Backup and Restore	GBAR	1.0.0.2
WebCop	WEBCOP	3.8.0.4
ITS Server	ITSSVR	4.7.2.0
General Military Intelligence DB	GMIDB	4.7.0.1
Common Track Data Store	CTDS	4.7.1.2
Image Management Database	IMDB	4.7.1.2
Navy Emitter Reference File DB	NERF	4.7.1.2
ELINT Parameters List Database	EPL	4.7.1.2
I3GMI	I3GMI	4.7.1.2
EDSS Database Install	CDEDSS	2.6.0.0
MCMSEG (Mainstream)	MSMIW	9.2.2.0
Tactical Warning Database	CDTWF	4.7.1.1
Tactical Track Archiver Database	CDTKAR	4.7.1.1
Track Warning Database	CDTRKW	4.7.1.1
Message Warning Database	CDMSGW	4.7.1.1
IMPACTS Toolkit	IMPXTK	4.7.1.1
Tactical Warning Framework	TWF	4.7.1.1
Tactical Track Archiver	TKAR	4.7.1.1
Tactical Warning Toolkit	TWTK	4.7.1.1
Track Warning	TRKW	4.7.1.1
Message Warning	MSGW	4.7.1.1

**COE-M Solaris Build 9 and Build 10 lists**

Build 10 list as of 12/02/02

<b>Segment Name</b>	<b>Build 9 Version</b>	<b>Build 10 Version</b>
Solaris 8 2/02		
DII COE Kernel	4.2.0.5	4.2.0.5
Solaris Patch Update	4.4.4.0	4.4.4.0
DII COE Kernel 4.2 Patch - 4.2.0.OP8A	4.2.0.OP8A	4.2.0.OP8A
Java Platform 2 (Java2)	4.6.0.0	4.6.0.0
J2SE JRE 1.4 - 4.6.0.0 (J2JRE)	4.6.0.0	4.6.0.0
NIS+ Admin Tool (NISADM)	4.3.0.0	4.3.0.0
Netscape Web Browser (WEBBr)	4.6.0.0	4.6.0.0
PERL	4.2.0.1	4.2.0.1
SecurityPolicyConfigurationTool (SPCFG)	4.2.0.1	4.2.0.1
SPCFG Data Segment (SPCFGD)	4.6.0.1	4.6.0.1
SPCFG Vulnerability Fixes Data (SPVULD)	4.6.0.0	4.6.0.0
COE Update System Security Level (UPDTSL)	4.6.0.0	4.6.0.0
COE Security Banner (SECBNR)	4.6.0.0	4.6.0.0
COE Security Banner Template (SBDATA)	4.6.0.0	4.6.0.0
DII COE System Menu Interface (SMB)	4.2.1.0	4.2.1.0
OnlineDocs (ONDOC)	4.2.1.0	4.2.1.0
Alerts Services Server (ALTSVR)	4.2.0.1	4.2.0.1
Alerts Services Client Runtime (ALTCLT)	4.2.0.1	4.2.0.1
CAPL Framework (CAPFW)	4.1.4.0	4.1.4.0
CAPL COE Clients (CAPCC)	4.1.4.0	4.1.4.0
NSS Libraries (NISSLIB)	4.5.0.0	4.5.0.0

SSAF Application (SSAFAP)	4.4.0.0	4.4.0.0
COE Security Services (COESS)	4.5.0.0	4.5.0.0
TCP Wrappers (TCPW)	4.6.0.0	4.6.0.0
VirusScan for UNIX (VSCANU)	4.5.0.0	4.5.0.0
Berkley Internet Name Domain (BIND)	4.0.5.0	4.0.5.0
Crack	4.0.0.1	4.0.0.1
JMTK Utilities Segment (JMU)	4.6.0.1	4.6.0.1
Integrated Foundation Library (IFL)	4.5.1.0	4.5.1.1
JMTK Visualization (JMV)	4.5.1.0	4.5.1.1
Application Framework (AFW)	4.5.1.0	4.5.1.1
Universal Comms Processor (UCP)	4.5.1.0	4.5.1.1
Tactical Management System (TMS)	4.5.1.0	4.5.1.1
TMS Visualization (TMSV)	4.5.1.0	4.5.1.1
ICSF C4I (IC4I)	4.5.1.0	4.5.1.1
Integrated Foundation Library (IFL)	4.5.1.0P1	
JMTK Visualization (JMV)	4.5.1.0P1	
Application Framework (AFW)	4.5.1.0P1	
Universal Comms Processor (UCP)	4.5.1.0P1	
Tactical Management System (TMS)	4.5.1.0P1	
TMS Visualization (TMSV)	4.5.1.0P1	
JMTK SDBM (JMS)	4.6.0.1	4.6.0.1
JMTK Analysis (JMA)	4.6.0.1	4.6.0.1
CAPL ICSF Client (CAPIC)	4.1.4.1	4.1.4.1
JMTK-V Map Data (JMVMD)	4.5.1.0	4.5.1.1
ICSF Online Documentation (ICSFDC)	4.5.1.0	4.5.1.1
DII COE Print Services Server (PRINTS)	4.4.1.1	4.4.1.1

DII COE Print Services Client (PRINTC)	4.4.1.1	4.4.1.1
DII COE Print Services Drivers (PRINTD)	4.4.1.1	4.4.1.1
Internet Relay Chat Server (IRCS)	4.0.0.0	4.0.0.0
Internet Relay Chat Client (IRCC)	4.3.0.0	4.3.0.0
C4I Common Extensions (CCE)	4.5.2.3	4.5.3.1
C4I Maritime Extensions (CME)	4.5.2.3	4.5.3.1
Tactical Information Best Svc (TIBS)	4.5.1.0	4.5.1.0
Air Tasking Exchange Runtime (ATXRUN)	4.5.5.1	4.5.5.1
Air Tasking Exchange Help (ATXHLP)	4.5.5.0	4.5.5.0
Tadil-A/B Interface (LINK11)	4.5.0.5	4.5.0.5
COP Synchronization Tools (CST)	4.5.6.0	4.5.6.0
COP Synchronization Tools (CST)	4.5.6.0P1	4.5.6.0P1
TMS Secret Data (TMSSD)	4.5.0.3	4.5.0.3
Extensible Information Systems (XIS)	4.5.0.3	4.5.0.3
XIS Map Integration (XISMI)	4.5.0.3	4.5.0.3
XIS Patch (XISP4)	4.5.0.0P4	4.5.0.0P4
XISMI Patch (XISMIP4)	4.5.0.0P4	4.5.0.0P4
Common Internet File System (CIFS)	4.4.0.0	4.4.0.0
CCE Network Time Protocol (CNTTP)	4.3.0.0	4.3.0.0
TreTabular Decoders (TTDEC)	4.5.0.0	4.5.0.0

**Windows build list**

Segment name	Prefix	Version #
	COMPOSE	2.0.1
	UAM	
DII COE Kernel	COE	4.2.0.5

DII COE Kernel 4.2 Patch - 4.2.0.OP8	K42P8B	4.2.0.OP8B
W2K Patch Update	W2KPTH	4.5.0.0
Java Platform 2	Java2	4.6.0.0
J2SE JRE 1.4 - 4.6.0.0	J2JRE	4.6.0.0
<b>*Netscape Web Browser</b>	<b>WEBBr</b>	<b>4.6.0.0</b>
COE Update System Security Level	UPDTSL	4.6.0.0
COE Security Banner	SECBNR	4.6.0.0
NSS Libraries	NSSLIB	4.5.0.0
COE Security Services	COESS	4.5.0.0
SSAF Application	SSAFAP	4.4.0.0
<b>*Norton Anti Virus</b>	<b>NAV</b>	<b>4.5.2.0</b>
OnlineDocs	ONDOC	4.2.1.0
Alerts Services Server	ALTSRV	4.2.0.2
Alerts Services Client Runtime	ALTCLT	4.2.0.2
<b>*Microsoft Office Pro 2000</b>	<b>OFFICE</b>	<b>4.4.0.0</b>
JMTK Utilities Segment	JMU	4.6.0.1 ED
Integrated Foundation Library	IFL	4.5.1.0
JMTK Visualization	JMV	4.5.1.0
Application Framework	AFW	4.5.1.0
Universal Comms Processor	UCP	4.5.1.0
Tactical Management System	TMS	4.5.1.0
TMS Visualization	TMSV	4.5.1.0
ICSF C4I	IC4I	4.5.1.0
Integrated Foundation Library	IFL	4.5.1.OP1
JMTK Visualization	JMV	4.5.1.OP1
Application Framework	AFW	4.5.1.OP1

Universal Comms Processor	UCP	4.5.1.0P1
Tactical Management System	TMS	4.5.1.0P1
TMS Visualization	TMSV	4.5.1.0P1
JMTK SDBM	JMS	4.6.0.1 ED
JMTK Analysis	JMA	4.6.0.1 ED
CAPL Framework	CAPFW	4.1.4.0
CAPL COE Clients	CAPCC	4.1.4.0
CAPL ICSF Client	CAPIC	4.1.4.1
ICSF Online Documentation	ICSFDC	4.5.1.0
JMTK-V Map Data	JMVMD	4.5.1.0
Internet Relay Chat Client	IRCC	4.0.1.0
C4I Common Extensions	CCE	4.5.2.3 ED
C4I Maritime Extensions	CME	4.5.2.3 ED
Tactical Information Best Svc	TIBS	4.5.1.0 ED
Air Tasking Exchange Runtime	ATXRUN	4.5.5.1 ED
Air Tasking Exchange Help	ATXHLP	4.5.5.0 ED
Tadil-A/B Interface	Link11	4.5.0.3 ED
COP Synchronization Tools	CST	4.5.6.0
COP Synchronization Tools	CST	4.5.6.0P1
Extensible Information Systems	XIS	4.5.0.3
XIS Map Integration	XISMI	4.5.0.3
XIS Patch	XISP4	4.5.0.0P4
XISMI Patch	XISMIP4	4.5.0.0P4
CCE Network Time Protocol	CNTP	4.3.0.0
<b>*MS Security Config Templates</b>	<b>W2KCET</b>	<b>4.6.0.1</b>

## COMPOSE software list

COMPOSE software build list version 2.03

<b>Platform</b>	<b>Application name (if <i>COTS</i>, provide vendor name)</b>	<b>Version</b>
Server	Symantec Live Update Admin Tool	2.0
Server	Adobe Acrobat Reader	6.0
Server	Netscape Communicator	7.02
Server	RealOne Player	2.0
Server	Macromedia Shockwave	8.5.1
Server	Flash Player	6.0.79
Server	Apple QuickTime Movie and Audio Viewer	6.3
Server	Microsoft Netmeeting	3.01
Server	Microsoft Windows Media Player	9
Server	Microsoft Windows 2000 Advanced Server	2000
Server	SP4 for Microsoft Windows 2000	SP4
Server	Microsoft ISA Server Enterprise Edition	2000
Server	Microsoft Exchange Enterprise Edition	2000
Server	Symantec Norton AntiVirus Corporate Edition	8.1
Server	Symantec AVF for Exchange	3.05
Server	NicoMak WinZip	8.1SR1
Server	Ipswitch WS-FTP Pro	8.01
Server	Veritas Backup Exec	9.0
Server	Microsoft Internet Explorer	6.0
Server	Microsoft SQL Server 2000 Standard Edition	2000
Server	SP3a for Microsoft SQL Server 2000	SP3a

Server	Symantic AntiVirus Systems Center Console	
Workstation	Microsoft Windows 2000 Professional	2000
Workstation	SP4 for Microsoft Windows 2000	SP4
Workstation	Microsoft Internet Explorer	6.0SP1
Workstation	Norton AntiVirus Corporate Edition	8.1
Workstation	NicoMak WinZip	8.1SR1
Workstation	Ipswitch WS-FTP Pro	8.01
Workstation	Adobe Acrobat Reader	6.0
Workstation	RealOne Player	2.0
Workstation	Macromedia Shockwave	8.5.1
Workstation	Flash Player	6.0.79
Workstation	Apple QuickTime Movie and Audio Viewer	6.3
Workstation	Microsoft Netmeeting	3.01
Workstation	Microsoft Windows Media Player	9
Workstation	Microsoft ISA Server Enterprise Edition Client	2000
Workstation	Microsoft Office 2000 Professional	2000
Workstation	ActivCard Gold	2.2
Workstation	Personal Security Manager for Netscape	1.4
Workstation	Microsoft Windows Active Directory Client	5.6
Workstation	Netscape	7.02
Workstation	Microsoft Office XP Professional	SP2
Workstation	Microsoft Office Professional (with FrontPage)	SP2
Workstation	Java Runtime Environment	1.4.2
Workstation	Microsoft Office 2000 Professional	SP3
Workstation	Microsoft Office 2000 Premium	SP3
Workstation	COE 4209 Kernel	4.2.0.9

## **Network security policy guidance**

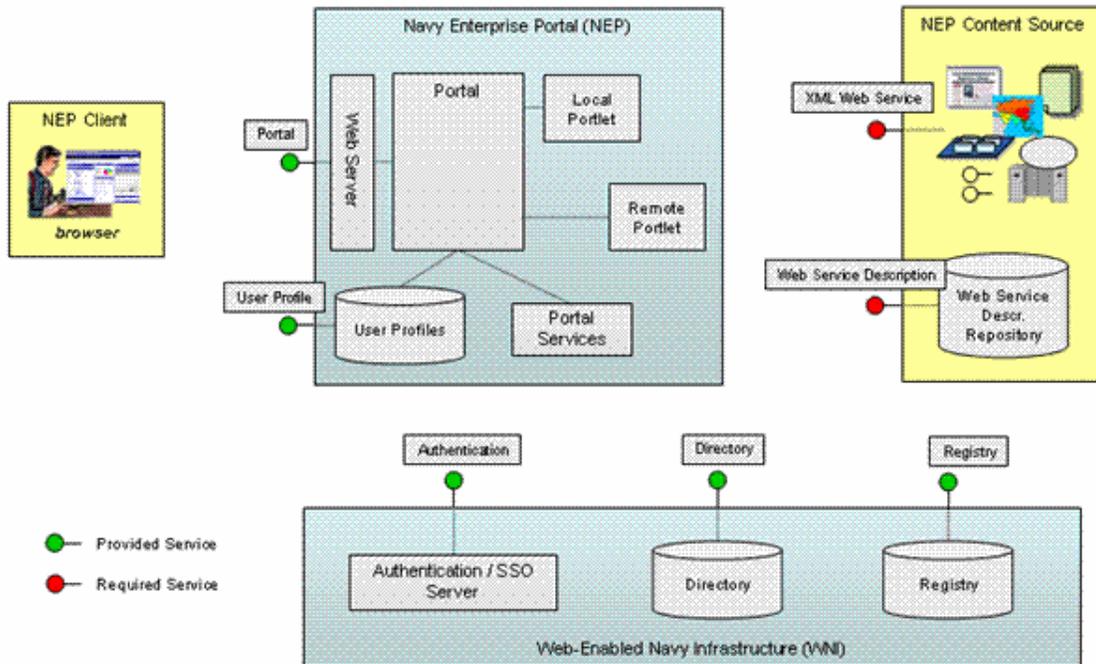
Follow the UTNProtect Policy Document CNO614 / HQMC C4 for network security policy guidance.

# Cross-reference between NESI and other initiatives

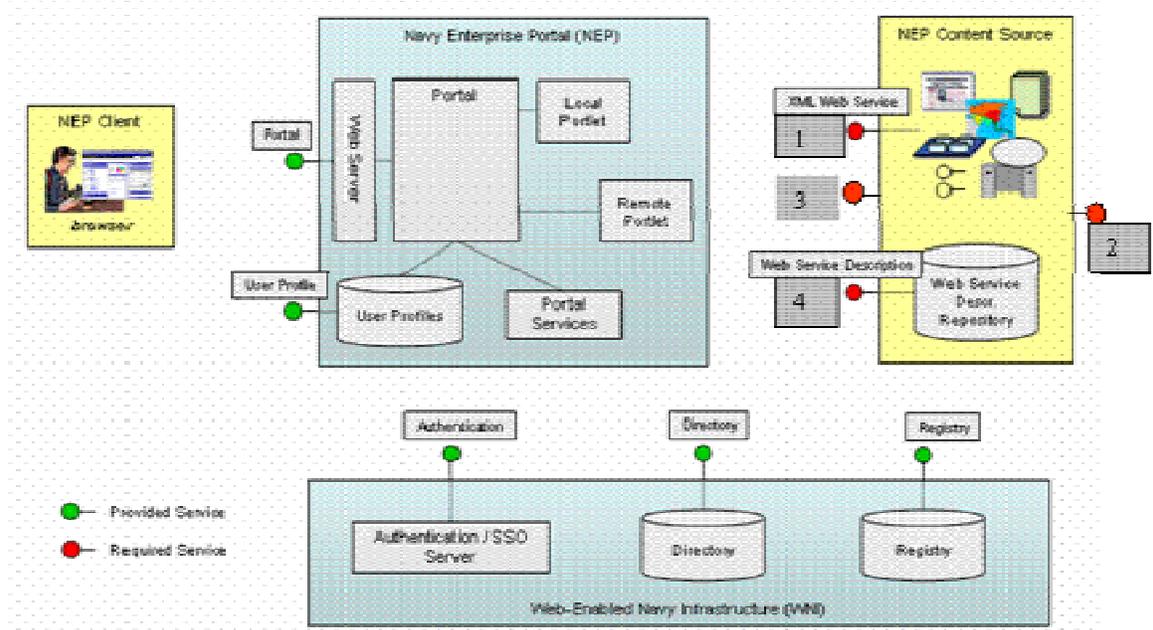
This section shows how *NESI* guidance overlaps or does not overlap with the developer's guides for other DoD initiatives such as *Navy Enterprise Portal (NEP) Architecture*.

## Navy Enterprise Portal (NEP) architecture

### Architecture diagrams



The figure below shows four areas where *NESI* guidance fits in to this architecture:



### Correlation between NEADG and NESI

Topics	Navy Enterprise Architecture Developer's Guide (NEADG)	NESI Focus Area	NESI Guide
General building and structuring <i>web services</i>		1 & 4	<i>Web services</i>
Insulated <i>interface</i> guidance		1 & 2	Interface design
<i>Web applications</i> structural guidance			Web applications
<i>Style sheets</i>	Appendix F		GUI design guidelines
<i>Portal</i> integration levels	Chapter 2		Web portals
Security	Appendix G	1 & 3	Testing security Web security Web services security

*NEP* URL rewriting    Appendix H

*UDDI* for web  
services

4

UDDI

## Open-source tools

This section explains how to obtain and use the *open-source* tools referred to in various examples throughout this guide.

*Disclaimer:* This example uses open-source products, since *NESI* itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

### Apache Ant

Apache Ant is a Java-based build tool that automates the build process. Ant uses *XML* descriptor files to capture the build process. It has the advantage of being interoperable with all other build tools.

For information about Apache Ant, including a manual on how to use it, see <http://ant.apache.org>. If you install Ant, the documentation is in the `<Ant install_dir>\docs` directory. See Generating Javadoc for an example of generating Javadoc with an Ant `build.xml` file.

### Installing Ant

#### To install Ant:

1. Go to <http://ant.apache.org/>.
2. Under **Download**, click **Binary Distributions**.
3. Scroll down to the **Current Release of Ant** section and download the appropriate version.
4. Expand the zip file into your target directory. WINZIP creates a subdirectory named **apache-ant-<version>**.
5. Follow the setup instructions to install Ant.
6. Make sure to add `<Ant install_dir>\bin` to the Windows system path variable.  
**Note:** If any of the directories in the path before the new path contain an `ant.bat` file, it will take precedence over the one you just added.
7. Make sure that you have defined an `ant_home` and a `java_home` environment variable.
8. Open a command window and type `ant -version` to verify the installation.

### Guidance

1. Operate on each directory independently. When you create a patch or new software release, it should only contain the new or modified classes. All unmodified classes remain untouched. This eliminates the need to perform a complete regression test and security scan.
2. Define a set of standard targets, such as:
  - `init`

- compile
- validate
- package
- deploy
- test
- clean
- doc

### Custom Ant extensions

Ant lets you customize the build process. For information on how to create custom extensions, see the Ant reference manual.

## Apache Axis

### To install Axis:

1. Go to <http://ws.apache.org/axis/index.html>.
2. Download and install version 1.1.

### To test Axis:

1. Open a web browser on the same machine as the Tomcat server and go to a URL such as <http://localhost:8080/axis/>, where:
  - `localhost` is the host name if the console is running on the same machine as the WebLogic server
  - `8080` is the default HTTP port number for Tomcat
  - `axis` is the name of the web application

The Apache-AXIS home page appears.



2. Click **Validate** to open the Axis Happiness page. All the checks should be successful.



Depending on your Java configuration, you may see two warnings about optional **JARs** that you need to install:

- **mail.jar** from <JavaMail install\_dir> to <Axis install\_dir>\WEB-INF\lib
- **xmlsec.jar** from xml-security-bin-<version>.zip into <Axis install\_dir>\WEB-INF\lib

3. Follow the links provided and install the components.

## Tomcat

Tomcat is developed by Jakarta, which is a project of the Apache Software Foundation. For information about Tomcat, see <http://jakarta.apache.org/tomcat/index.html>.

### Installing Tomcat

The Tomcat 4.x server implements the *JSP* and *Servlet* specifications. Different versions of Tomcat apply to different specifications, so it is important to get the correct version. Also, a web application in Tomcat should work in any other server that is compliant with the specification. This installation uses version 4 of Tomcat.

#### Prerequisites

To install and run Tomcat, you must install a Java Software Development Kit for version 1.2 (or later).

#### To install the Java Software Developer Kit:

1. Download and install the Java *SDK*, version 1.2 or later.
2. Set a `JAVA_HOME` environment variable to the the installation directory.

#### To install Tomcat:

1. Go to <http://jakarta.apache.org/tomcat/index.html>. Scroll down to view a comparison of features in the different versions.
2. In the **Release Builds** section, download the latest release of the Tomcat server.
3. Install the release.

## Using Tomcat

#### To start Tomcat:

1. Define the `JAVA_HOME` environment variable and set it to `<Java install_dir>`.
2. Define the `CATALINA_HOME` environment variable and set it to `<Tomcat install_dir>`.  
**Note:** Some applications may require additional Tomcat environment variables, such as `CATALINA_OPTS`. One thing this variable does is set the web server memory requirements. If you use it for that, set `CATALINA_OPTS = -Xmx256m`.
3. After the server has successfully started, view a local Tomcat home page by going to the `http://localhost:8080/` link from a web browser. (If you changed your port number, substitute it for 8080.)

## Xalan-Java

*Xalan-Java* is an XSLT processor made by Apache. XSLT processors transform *XML* documents into HTML, text, or other XML document types. Xalan-Java implements *XSL Transformations (XSLT) Version 1.0* (<http://www.w3.org/TR/xslt>) and *XML Path Language (XPath) Version 1.0* (<http://www.w3.org/TR/xpath>).

You can use Xalan-Java from the command line, in an *applet* or a *servlet*, or as a *module* in other program.

For Xalan-Java FAQs, documentation, and sample applications, go to <http://xml.apache.org/xalan-j/>.

#### To install Xalan-Java:

1. Go to <http://xml.apache.org/xalan-j/downloads.html>.
2. Click the **xalan-j distribution directory** link to download the binary distributions.
3. Select a mirror site, then download the zip file that is appropriate for your site.
4. Expand the zip file into your target directory. WINZIP creates a subdirectory named **xalan-<version>** that contains the Java XSLT software.

## Xerces2 Java Parser

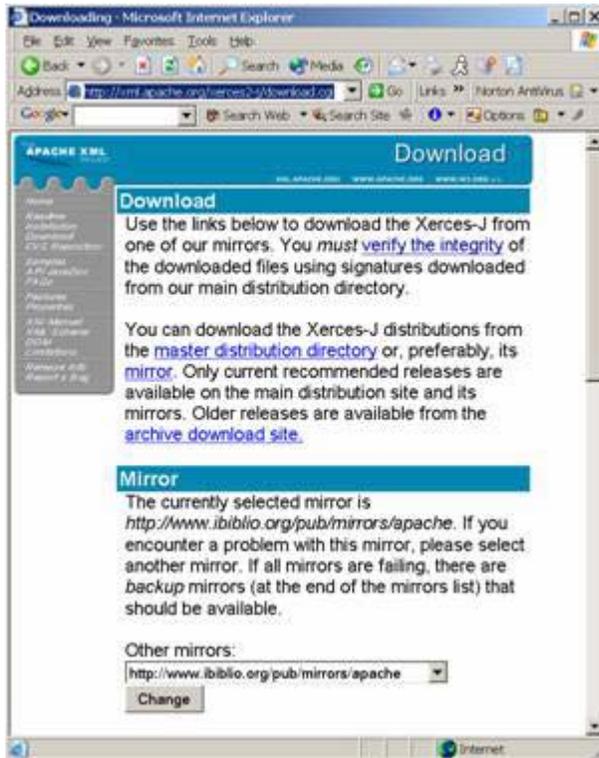
Xerces2 is the next generation of high-performance, fully compliant *XML parsers* in the Apache Xerces family. This new version of Xerces introduces the Xerces Native Interface (XNI), a complete framework for building parser *components* and configurations that is extremely modular and easy to program. Xerxes2 is one of the products you can install to run the *UDDI browser*.

The Apache Xerces2 parser is the reference implementation of XNI. You can write other parser components, configurations, and parsers using the Xerces Native Interface. For complete design and implementation documents, refer to the *XNI Manual*. Xerxes2 is a fully conforming XML schema processor.

To obtain FAQs and documentation for Xerxes2, go to <http://xml.apache.org/xerces2-j/index.html>.

#### To install the Xerxes2 Java Parser:

1. Go to <http://xml.apache.org/xerces2-j/download.cgi>.



2. Download the appropriate file. This example shows the **Xerces-J-bin.2.6.0.zip** file.



3. Expand the zip file into your target directory. WINZIP creates a subdirectory named **xerxes-<version>** that contains the XML parser software.
4. Configure Xerxes for your environment, including setting your class path.

## jUDDI

*Disclaimer:* This example uses open-source products, since **NESI** itself is built on the open-source philosophy. However, the products described are not necessarily the best choice for every circumstance.

jUDDI (pronounced "Judy") is an open-source Java implementation of the Universal Description, Discovery, and Integration (**UDDI**) specification for web services. It uses the *Tomcat* environment. If you use jUDDI, you must program with *UDDI4J* to insert entries into the registry, since jUDDI does not come with an insertion tool.

**Note:** This is not the DoD Enterprise Registry. This example is intended only to help familiarize developers with the technology.

You must have the following things running to use jUDDI:

- Database
- **JDBC** interface
- Browser

Although the specifics may differ, most UDDI registries use a database, browser, and some kind of server.

### Key characteristics

- Compliant with UDDI version 2.0
- Works with any relational database that supports **ANSI**-standard **SQL** (MySQL, DB2, Sybase, JDataStore, etc.)
- Deployable on any Java application server that supports the Servlet 2.3 specification (Jakarta Tomcat, WebSphere, WebLogic, Borland Enterprise Server, JRun, etc.)
- Supports a clustered deployment configuration
- Integrates with existing **authentication** systems

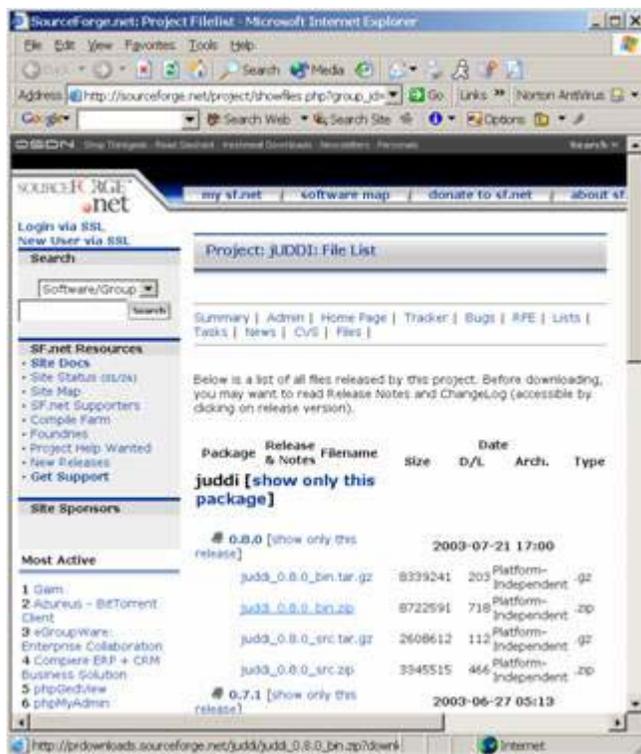
### Example: Setting up a jUDDI registry

The pages in this section explain how to set up a UDDI registry, using jUDDI as an example.

## Installing jUDDI

### Installing the jUDDI service

1. Go to [http://sourceforge.net/project/showfiles.php?group\\_id=42875](http://sourceforge.net/project/showfiles.php?group_id=42875) and download the latest binaries.



2. Expand the zip file into your target directory. WINZIP creates a **juddi** subdirectory.
3. Ensure that Tomcat is not running.
4. Copy the **juddi.war** file from the newly extracted **juddi\bin\build** directory to **<Tomcat install\_dir>\webapps\juddi**. This enables you to configure jUDDI.
5. Ensure that the `java_home` environment variable is *defined*.
6. Start the Tomcat server, then shut it down again.

## Configuring the properties files

Follow these directions to configure the jUDDI properties files. Other than these changes, leave the initial configuration as is.

### ***juddi.properties***

This file sets the database configuration.

1. Go to: **<drive letter>:\<Tomcat install\_dir>\webapps\juddi\WEB-INF\classes\juddi.properties**.
2. Find the following lines and remember the database user name and password for later reference. If you change them here, you must also change them in the database.

```
# The jUDDI ConnectionPool properties (Optional)
juddi.useConnectionPool=false
juddi.jdbcDriver=com.MySQL.jdbc.Driver
juddi.jdbcURL=jdbc:MySQL://localhost/juddi
juddi.jdbcUser=juddi
juddi.jdbcPassword=juddi
```

3. Find the following lines and enter the correct ports:

```
# jUDDI Proxy Properties (Used by RegistryProxy)
# see http://www.juddi.org for more information
juddi.inquiryURL=http://localhost:8080/juddi/inquiry
juddi.publishURL=http://localhost:8080/juddi/publish
juddi.adminURL=http://localhost:8080/juddi/admin
```

### ***log4j.properties***

This file handles setup logging.

1. Go to <drive letter>:\<Tomcat install\_dir>\webapps\juddi\WEB-INF\classes\log4j.properties.
2. Find the following lines and indicate the level of logging you want:

```
# Set root category priority to DEBUG and its appender to
LOGFILE.
log4j.rootCategory=WARN, LOGFILE
```

### ***web.xml***

This file sets up the server.

1. Go to :\\webapps\juddi\WEB-INF\web.xml.
2. Find the following lines and uncomment them:

```
<!-- Uncomment to enable jUDDI's Administrative Services →
<servlet-mapping>
  <servlet-name>jUDDIAdminServlet</servlet-name>
  <url-pattern>/admin</url-pattern>
</servlet-mapping>
<!-- -->
```

3. Find the following lines and check the database name:

```
<resource-ref>
  <description>jUDDI DataSource</description>
  <res-ref-name>jdbc/juddiDB</res-ref-name>
  javax.sql.DataSource
  <res-auth>CONTAINER</res-auth>
</resource-ref>
```

4. Find the following lines and update the paths:

```
<env-entry>
  <env-entry-name>log4j.propsFile</env-entry-name>
  <env-entry-value>
    <drive letter>:\<Tomcat install_dir>\webapps\juddi\WEB-
INF\classes
  </env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>juddi.propsFile</env-entry-name>
  <env-entry-value>
    <drive letter>:\<Tomcat install_dir>\webapps\juddi\WEB-
INF\classes
  </env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
```

```
</env-entry>
</web-app>
```

## Testing the server

1. Start Tomcat.
2. Go to `localhost:8080/juddi/index.html`.
3. Click **Validate**. The server runs a number of tests and displays the results. The data source check fails because you have not connected the database yet. Everything else should be up and running.

## Installing MySQL for jUDDI persistence

The jUDDI service needs a database. jUDDI supports several Relational Database Management Systems (*RDBMS*). This section shows how to use MySQL to provide data persistence.

### UDDI scripts

jUDDI provides *UDDI* scripts for creating tables in various databases. Go to `<jUDDI install_dir>\ddl\` to find the following support files.

Script	Database
<code>juddi_ase.ddl</code>	Sybase
<code>juddi_db2.ddl</code>	IBM's DB2
<code>juddi_hsql.ddl</code>	HSQL
<code>juddi_mysql.ddl</code>	Open-source MySQL

### Creating a database

#### To set up MySQL for UDDI

1. Make sure you have installed MySQL and the MySQL Control Center.
2. Make sure you have installed the MySQL *JDBC* drivers.
3. Open a command window and set the default directory to `<MySQL install_dir>\bin`.
4. Start the MySQL command processor as root with the command:

```
MySQL --user=root MySQL
```

5. To create a new database named `juddiDB`, enter:

```
CREATE DATABASE IF NOT EXISTS juddiDB;
```

6. This name must be the same name as the one in the `juddi.properties` file. Note that *it* is case-sensitive.
7. Enter the following command to create a root user who can access the `juddiDB` database from outside:

```
GRANT ALL PRIVILEGES ON juddiDB.* TO juddi@'%'
```

```
IDENTIFIED BY 'juddi' WITH GRANT OPTION;
```

8. Enter the following command to create a root user who can access the juddiDB database locally.

```
GRANT ALL PRIVILEGES ON juddiDB.* TO juddi@localhost
IDENTIFIED BY 'juddi' WITH GRANT OPTION;
```

The user name and password are set in juddi.properties. If you use different names, you will not be able to connect to the database.

9. Log out of the command processor.

### Creating tables

After you create the jUDDI database, you need to create tables for the UDDI registry information. There are two ways to do this:

- from a *command window*
- from the *MySQL command center* (MySQLcc)

#### To create tables from a command window:

1. Open a command window and set the default directory to <MySQL install\_dir>\bin.
2. Log in to the database you defined above (juddiDB) with the UDDI database user name and password.

```
MySQL --database=juddiDB --user=juddi --password=juddi
```

3. Go to <jUDDI install\_dir>\ddl\ and run the MySQL script, **juddi\_MySQL.ddl**.
4. Type `show tables` to see a list of the tables you just created:

```
C:\WINDOWS\System32\cmd.exe - mysql --database=juddiDB --user=juddi
C:\OpenSource\MySQL\4.0\bin>mysql --database=juddiDB --user=juddi --password=juddi
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12 to server version: 4.0.13-max-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show tables;
Empty set (0.00 sec)

mysql> source C:\OpenSource\juddi_0.9.0\bin\ddl\juddi_mysql.ddl
ERROR 1051: Unknown table 'business_descr'
ERROR 1051: Unknown table 'business_category'
ERROR 1051: Unknown table 'business_identifier'
ERROR 1051: Unknown table 'business_name'
ERROR 1051: Unknown table 'discovery_url'
ERROR 1051: Unknown table 'address_line'
ERROR 1051: Unknown table 'address'
ERROR 1051: Unknown table 'phone'
ERROR 1051: Unknown table 'email'
ERROR 1051: Unknown table 'contact_descr'
ERROR 1051: Unknown table 'contact'
ERROR 1051: Unknown table 'service_descr'
ERROR 1051: Unknown table 'service_category'
ERROR 1051: Unknown table 'service_name'
ERROR 1051: Unknown table 'binding_descr'
ERROR 1051: Unknown table 'instance_details_descr'
ERROR 1051: Unknown table 'instance_details_doc_descr'
ERROR 1051: Unknown table 'tmodel_category'
ERROR 1051: Unknown table 'tmodel_descr'
ERROR 1051: Unknown table 'tmodel_doc_descr'
ERROR 1051: Unknown table 'tmodel_identifier'
ERROR 1051: Unknown table 'tmodel_instance_info_descr'
ERROR 1051: Unknown table 'tmodel_instance_info'
ERROR 1051: Unknown table 'publisher_assertion'
ERROR 1051: Unknown table 'tmodel'
ERROR 1051: Unknown table 'binding_template'
ERROR 1051: Unknown table 'business_service'
ERROR 1051: Unknown table 'business_entity'
ERROR 1051: Unknown table 'publisher'
ERROR 1051: Unknown table 'auth_token'
Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)
```

It is normal to see a long list of Unknown Table errors. The tables are not there to begin with, and the script creates them after doing some other setup procedures.

5. Enter the following code to create a jUDDI publisher. Not all UDDI registries have a publisher.

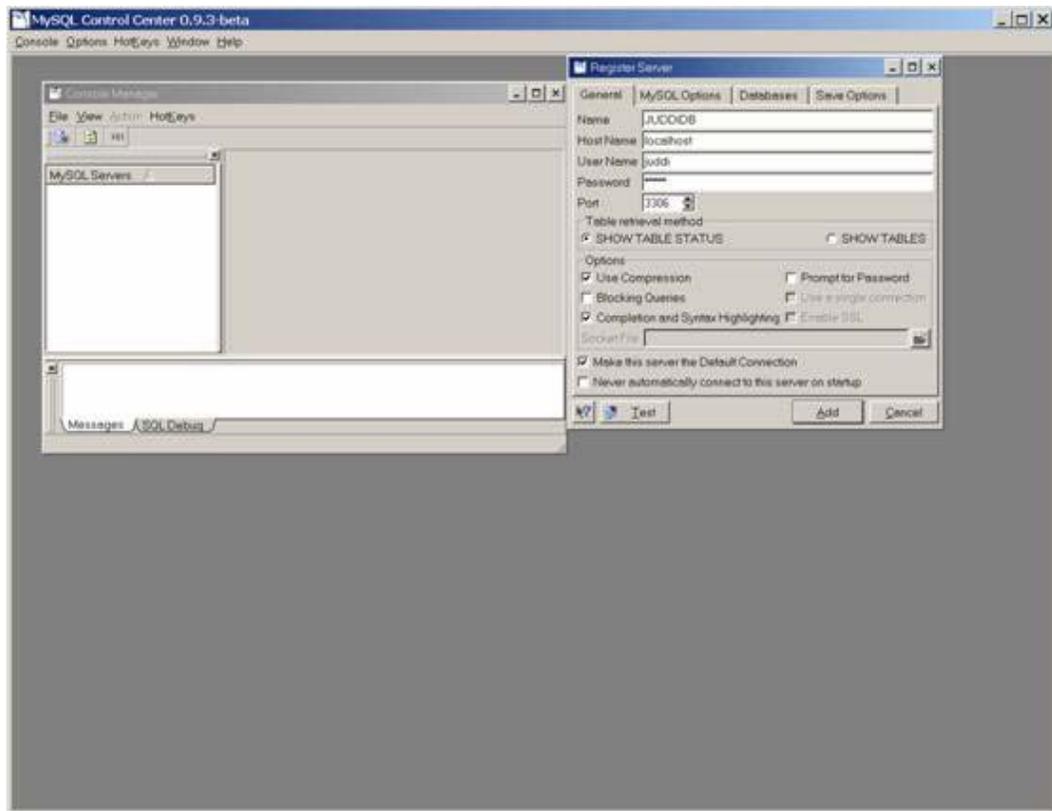
```
DELETE FROM PUBLISHER
  WHERE PUBLISHER_ID = 'juddi';

INSERT INTO PUBLISHER
  ( PUBLISHER_ID,
    PUBLISHER_NAME,
    ADMIN
  )
VALUES
  ('juddi',
   'Juddi user',
   'false'
  );
```

The DELETE statement may fail if the PUBLISHER\_ID of juddi does not already exist. Ignore this error.

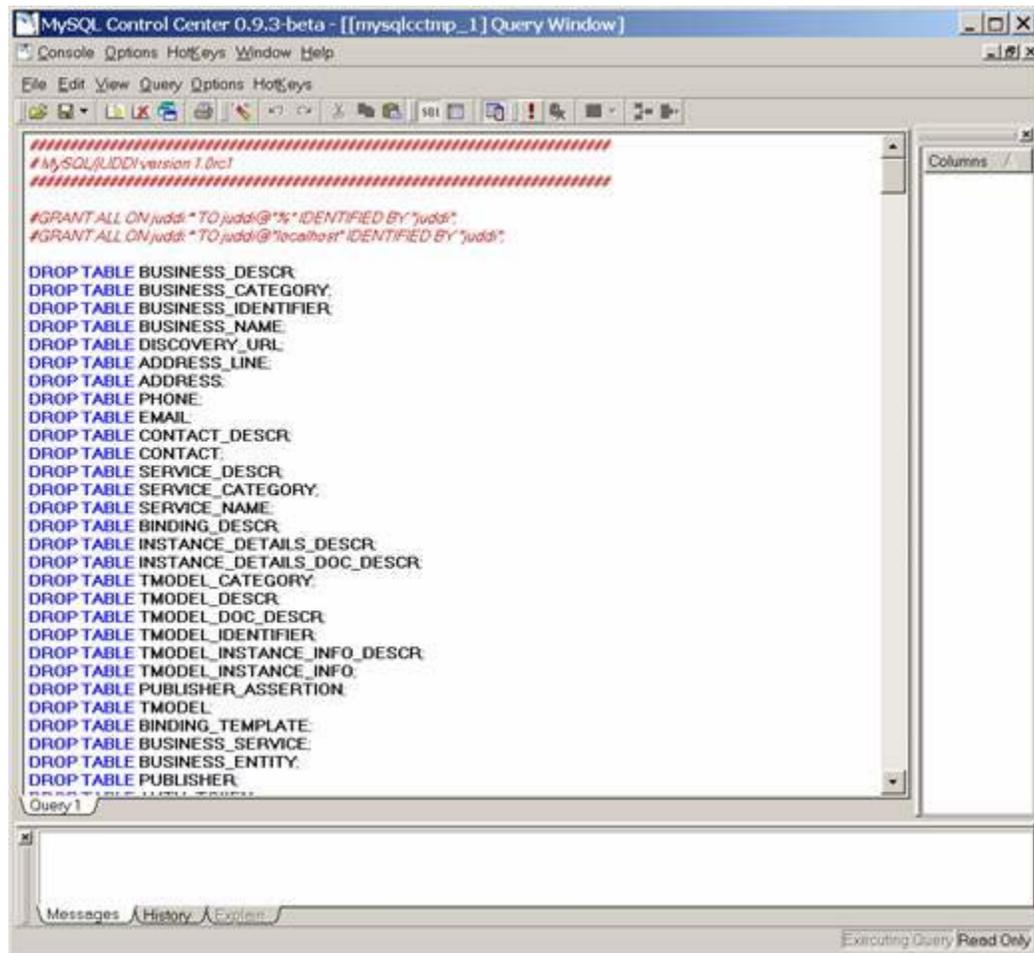
#### To create tables from MySQLcc:

1. To start the MySQLcc, double click the desktop icon or click **Start** and select **Program Files > MySQL Command Center > MySQL Command Center**.
2. Start and register the MySQL service. A registration dialog should open automatically the first time you start MySQLcc. This example uses the juddiDB database on a local machine (i.e. localhost). The user name and password are both juddi.



3. Maximize the MySQLcc window in the Console Manager.
4. Select **File > SQL Query** to open a SQL command window.
5. From MySQLcc, go to `<jUDDI install_dir>\bin\ddl` and open the MySQL script, **juddi\_MySQL.ddl**. This opens the JUDDI MySQL Data Definition Language (DDL) file.

6. Select **Query > Execute** to execute the jUDDI table creation script.

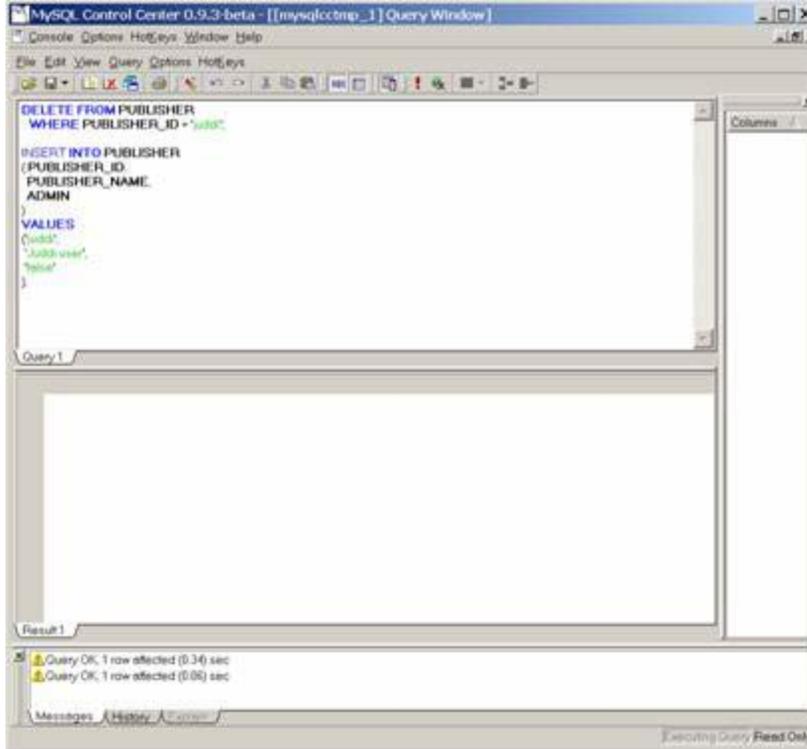


7. After executing the creation script, you need to define a *publisher*. Select **Query > Clear Query** to clear the buffer.
8. Enter the following command into the query command area:

```
DELETE FROM PUBLISHER
  WHERE PUBLISHER_ID = 'juddi';

INSERT INTO PUBLISHER
  ( PUBLISHER_ID,
    PUBLISHER_NAME,
    ADMIN
  )
VALUES
  ('juddi',
   'Juddi user',
   'false'
  );
```

This command inserts a single record into the Publisher table that identifies the juddi user. The first DELETE statement may fail if you have not previously run the INSERT statement, because it is trying to delete the [nonexistent] juddi user.



### **Publisher table**

The PUBLISHER table contains the following columns:

Column name	Description
<i>PUBLISHER_ID</i>	The user ID the publisher uses when authenticating. <b>IMPORTANT:</b> This should be the same value used to authenticate with the external authentication service.
<i>PUBLISHER_NAME</i>	The publisher's name (or in UDDI-speak, the Authorized Name).
<i>ADMIN</i>	Indicates if the publisher has administrative privileges. Valid values for this column are true or false. The ADMIN value is currently not used.

### **Configuring jUDDI**

Now you need to add some configuration parameters to the web server.

1. Go to <drive letter>:\<Tomcat install\_dir>\conf\ and open **server.xml**.
2. Copy the following code to the end of the file. In the <Resource section, ensure that the name matches that of the database.

```
<Context
  className="org.apache.catalina.core.StandardContext"
  backgroundProcessorDelay="-1"
```

```

    cachingAllowed="true"
    charsetMapperClass="org.apache.catalina.util.CharsetMapper"
    configFile="<drive letter>:\<Tomcat
install_dir>\conf\Catalina\localhost\juddi.xml"
    cookies="true"
    crossContext="true"
    debug="5"
    displayName="JUDDI"
    docBase="juddi"
    domain="Catalina"
    engineName="Catalina"
    j2EEApplication="none"
    j2EEServer="none"
    lazy="true"
    managerChecksFrequency="6"
    path="\juddi"
    privileged="false"
    reloadable="true"
    startupTime="30"
    swallowOutput="false"
    tldScanTime="0"
    useNaming="true"
    wrapperClass="org.apache.catalina.core.StandardWrapper">
<Logger className="org.apache.catalina.logger.FileLogger"
    debug="0"
    directory="logs"
    prefix="localhost_juddiDB_log"
    suffix=".txt"
    timestamp="true"
    verbosity="1"/>
<Resource
    auth="Container"
    name="jdbc/juddiDB"
    scope="Shareable"
    type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/juddiDB">
  <parameter>
    <name>factory</name>
    <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
  </parameter>
  <parameter>
    url
    <value>jdbc:mysql://localhost:3306/jUDDIDB?autoReconnect=true</v
alue>
  </parameter>
  <parameter>
    <name>password</name>
    <value>juddi</value>
  </parameter>
  <parameter>
    <name>maxWait</name>
    <value>10000</value>
  </parameter>
  <parameter>
    <name>maxActive</name>
    <value>100</value>
  </parameter>

```

```

<parameter>
  <name>driverClassName</name>
  <value>org.gjt.mm.MySQL.Driver</value>
</parameter>
<parameter>
  <name>username</name>
  <value>juddi</value>
</parameter>
<parameter>
  <name>maxIdle</name>
  <value>30</value>
</parameter>
</ResourceParams>
</Context>

```

## Testing jUDDI

1. Make sure that the MySQL service is up and running.
2. Stop and start Tomcat.
3. Open a browser and go to `http://localhost:8080/juddi/happyjuddi.jsp`. For the port, use the value from `juddi.xml`.

The Happiness page should appear, and all tests should report positive results. If not, correct the problems until all tests report positive results.



The jUDDI registry is now set up. The next step is to connect to it using *UDDI4J* or a *UDDI browser*.

## Installing a UDDI API

Install UDDI4J to create a **UDDI** publisher for the jUDDI registry, which does not come with its own publisher. UDDI4J is a Java class library that provides an **API** to interact with a UDDI registry. For more information, go to <http://www.uddi4j.org>.

### To install UDDI4J:

1. Go to <http://www-124.ibm.com/developerworks/oss/uddi4j/>.
2. Click **Downloads/Releases** and download the appropriate binary.
3. Expand the zip file into your target directory. WINZIP creates a subdirectory named **uddi4j** that contains the Java UDDI4J software.

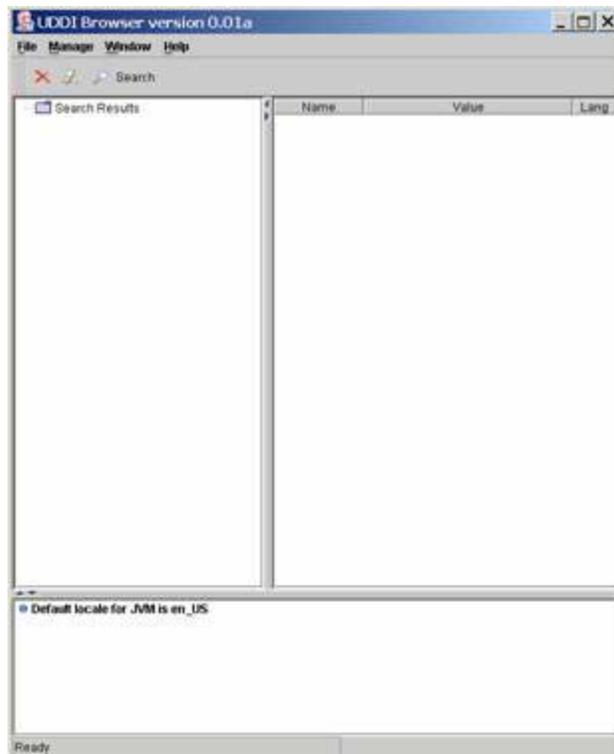
## UDDI browsers

The **UDDI** browser is an open-source project that lets users review and manipulate the contents of public and private UDDI registries via a web browser. The UDDI browser can talk to any UDDI registry, such as *jUDDI*.

### About the UDDI browser

The UDDI browser (shown below) contains the following elements:

- a menu bar
- a hierarchical tree for the search result nodes
- the details portion that gives the name, value, and language values associated with the search result nodes
- a message area that provides feedback to the caller



## Installing the UDDI browser

### Prerequisites

To install the **UDDI** browser, you must first install the following software:

<i>Apache Ant</i>	A platform-independent <b>XML</b> build tool. This is not required if you use the binaries for the browser.
<i>Apache Axis</i>	A <b>SOAP</b> transport mechanism.
<b>JavaMail</b>	A set of abstract classes that model a mail system. You do not need this if your Java environment already has mail.
Java Standard Secure Socket Edition ( <b>JSSE</b> )	This is already present if you have installed the Java Developer's Kit version 1.4 or later.
<i>UDDI4J</i>	A Java class library with an <b>API</b> that interacts with a UDDI registry.
<i>Xalan-Java</i>	An <b>XSLT</b> that transforms XML documents into HTML, text, or other XML document types
<i>Xerces2 Java Parser</i>	A <b>parser</b> for XML schemas

### Installing the browser

#### To install the UDDI browser:

1. Go to <http://uddibrowser.org>.
2. Click **Download** and download the latest version.
3. Expand the zip file into your target directory. WINZIP creates a subdirectory named **ub-<version>** that contains the browser software.

### **Configuring the batch file**

The simplest way to start the browser is to execute a batch file.

#### **To configure ub.bat:**

1. Go to <UB\_install\_dir>\bin and open **ub.bat**.
2. Enter the following line at the top of the file:

```
CD ..\
```

3. Update all the paths in **ub.bat**.

## **Running and testing the UDDI browser**

### **Starting the browser**

#### **To start the UDDI browser:**

1. Make sure that the MySQL database server and the Tomcat web server are up and running.
2. Execute *ub.bat*.

### **Testing the browser against a public registry**

To test the browser, run a query against a public registry and see if you get results.

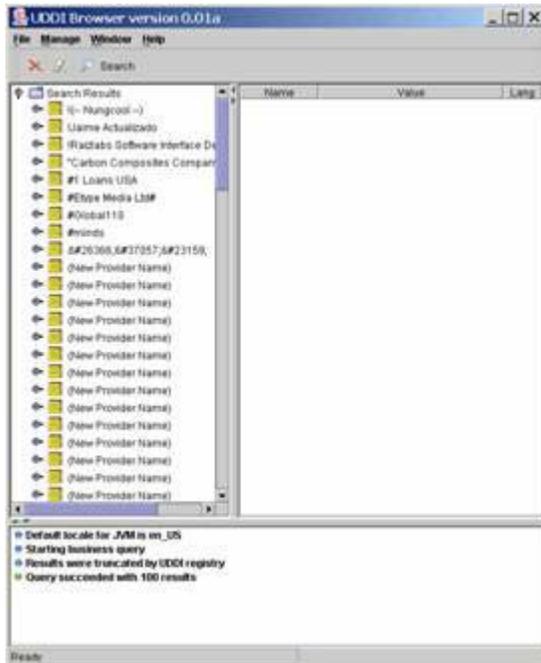
#### **To query a public registry:**

1. Click **Search** in the browser window.
2. In the Find dialog, scroll down to the desired registry and click **Search**.

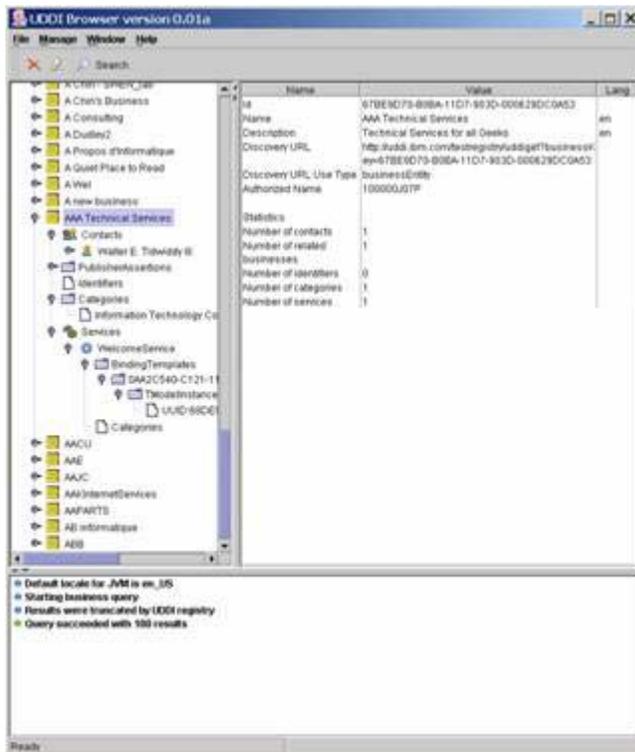


The results appear in the browser window. The list of nodes is on the left side, and the

query status information is at the bottom.



3. Select nodes on the left to view details about them on the right.

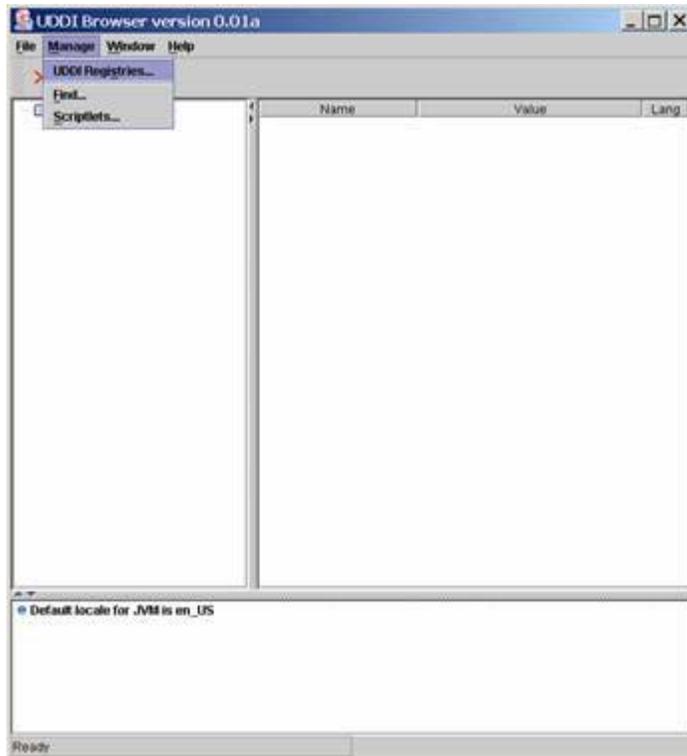


### **Testing the browser against a private registry**

To connect to the jUDDI registry:

The UDDI browser lets you view two types of registry: public and private. Public registries come predefined in the browser. jUDDI is a private registry because you *installed it locally*.

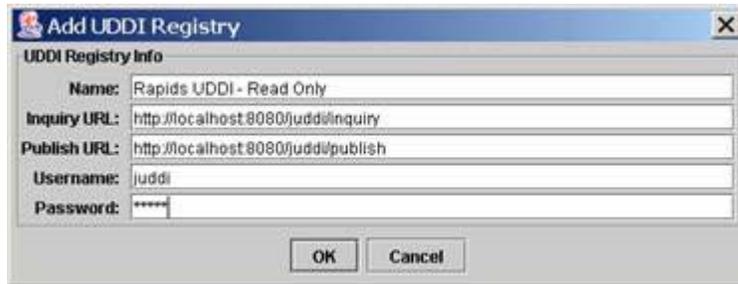
1. Select **Manage Registries > UDDI Registries**.



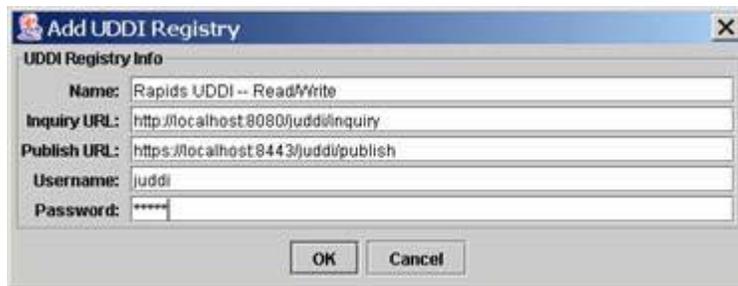
2. Click **Add**.



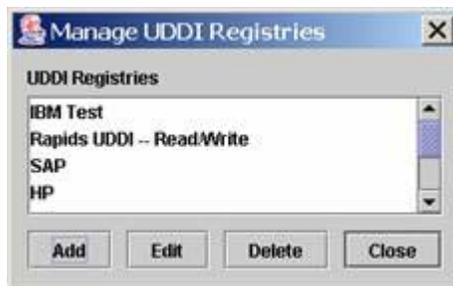
- To add a read-only query, define a read-only UDDI registry entry that uses the Tomcat Standard HTTP Port 8080. Provide the user name and password you defined earlier (user name: juddi, password: juddi). The Inquiry and Public URLs must point to the registry you just installed.



- To add a read-write query, define a read-write UDDI registry entry that uses the Tomcat Standard HTTP port 8080 for reading and SSL port 8443 for writing. Provide the username and password defined earlier (user name: juddi, password: juddi).



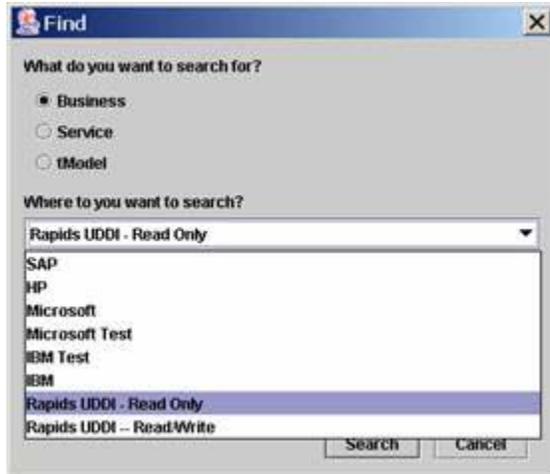
The new entries appear on the list of managed registries.



### To find UDDI objects:

These instructions show how to test the browser against your local registry, by searching for various objects.

1. Click **Search** in the browser window.
2. Select the **tModel** option.
3. Select the jUDDI registry entry you just created from the list (here, named NESI UDDI) and click **Search**.



A screen like the following one appears. The data that appear here are the predefined data that you entered when you executed the *predefined MySQL script* to create UDDI tables.



The registry is now ready for use.



---

# Glossary

## #

**.NET:** To address the confusing maze of computer languages, libraries, tools, and toolkits that were necessary for creating multi-tier applications, Microsoft developed the .NET Framework and integrated it into Microsoft Windows as a component. It supports building and running multi-tier and service-oriented architectures, including web services and client and server applications. It simplifies the process of designing, developing, and testing software, allowing individual developers to focus on core, application-specific code.

## A

**ACAT:** Acquisition Category. DoD acquisition program categories that facilitate decentralized decision making, execution, and compliance with statutorily imposed requirements. The categories determine the level of review, decision authority, and applicable procedures. (Source: [http://www.dau.mil/pubs/glossary/11th Glossary 2003.pdf](http://www.dau.mil/pubs/glossary/11th%20Glossary%202003.pdf))

**access control:** The methods by which interactions with resources are limited to collections of users or programs for the purpose of enforcing integrity, confidentiality, or availability constraints. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**ACID:** Atomicity, Consistency, Isolation, Durability. The acronym for the four properties guaranteed by transactions. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**ActiveX:** An ActiveX control is similar to a Java applet. However, ActiveX controls have full access to the Windows OS. This gives them much more power than Java applets, plus a risk that the applet may damage software or data on your machine. To control this risk, Microsoft developed a registration system so that browsers can identify and authenticate an ActiveX control before downloading it. Another difference between Java applets and ActiveX controls is that Java applets can be written to run on all platforms, whereas ActiveX controls are currently limited to Windows environments.

**adapter:** An intermediary that translates between incompatible component interfaces, allowing them to communicate.

**adapter pattern:** A generalized API that provides a common set of function calls across different applications. It enables classes with incompatible interfaces to work together. It is sometimes called a wrapper because an adapter class wraps the implementation of another class in the desired interface. This pattern makes heavy use of delegation, where the delegator is the adapter (or wrapper) and the delegate is the class being adapted.

**AGI:** Americal Geological Institute

**air warfare:** Air defense against airborne weapons including theater ballistic missiles. Operations include surveillance, offensive counter air, defensive counter air, and electronic warfare.

**anonymous access:** Accessing a resource without authentication. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**ANSI:** American National Standards Institute. Administrator and coordinator of the United States private-sector voluntary standardization system. ANSI facilitates the development of American National Standards (ANS) by accrediting the procedures of standards-developing organizations. The Institute remains a private, nonprofit membership organization supported by a diverse constituency of private and public sector organizations. (Source:<http://web.ansi.org/>)

**AoA:** Analysis of Alternatives. Provides analysis and suggestions for performance characteristics. Assesses the advantages and disadvantages of alternatives, including the sensitivity of each alternative to possible changes in key assumptions or variables. (Source: [http://www.dau.mil/pubs/glossary/11th Glossary 2003.pdf](http://www.dau.mil/pubs/glossary/11th%20Glossary%202003.pdf))

**Apache Ant:** A Java-based build tool that automates the build process using XML descriptor files to capture the build process.

**APB:** Acquisition Program Baseline. Establishes program threshold and objective values for the minimum number of cost, schedule, and performance attributes that describe the program over its life cycle. (Source: [http://www.dtic.mil/cjcs\\_directives/cdata/unlimit/3170\\_01.pdf](http://www.dtic.mil/cjcs_directives/cdata/unlimit/3170_01.pdf))

**API:** Application Programming Interface. A special type of interface that specifies the calling conventions with which one component may access the resources and services provided by another component. APIs are defined by sets of procedures or function-invocation specifications. An API is a special case of an interface.

**API adapter:** A generalized API that provides a common set of function calls across different applications.

**applet:** A J2EE component that typically executes in a web browser. Applets can also execute in a variety of other applications or devices that support the applet programming model. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**application diversity:** A situation where users can pull multiple apps to access the same data or choose the same app (e.g., for collaboration).

**application server:** A platform for developing and deploying multi-tier distributed enterprise applications.

**architecture:** (1) The structure of components, their relationships, and the principles and guidelines governing their design and evolution over time. (2) A high-level design that provides decisions about the problem(s) that the product will solve, component descriptions, relationships between components, and dynamic operation description. (3) A framework or structure that portrays relationships among all the elements of the subject force, system, or activity. Also, the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. The organizational structure of a system or component, their relationships, and the principles and guidelines governing their design and evolution over time. (Source: IEEE 610.12)

**architecture views, software:** Conceptual Architecture. The purpose of the conceptual architecture is to direct attention at an appropriate decomposition of the system without delving into details. Moreover, it provides a useful vehicle for communicating the architecture to non-technical audiences, such as management, marketing, and users. It consists of the Architecture Diagram (without interfaces) and an informal component specification (which we call CRC-R cards) for each component.

- architecture, functional:** The hierarchical arrangement of functions, their internal and external (to the aggregate itself) functional interfaces and external physical interfaces, their respective functional and performance requirements, and design constraints.
- architecture, software:** (1) The software architecture of a program or computing system is the structure or structures of the system, which comprise (a) software components, (b) the externally visible properties of those components, and (c) the relationships among them. (2) The structure and relationships among the components of a computer program. The software architecture may also include the program's interface with its operations environment.
- architecture, system:** (1) A logical, physical structure that specifies interfaces and services provided by the system components necessary to accomplish system functionality. (2) The structure and relationship among the components of a system. The system architecture may also include the system's interface with the operational environment.
- AS:** Acquisition Strategy. High-level business and technical management approach designed to achieve program objectives within specified resource constraints. Framework for planning, organizing, staffing, controlling, and leading a program. (Source: [http://www.dau.mil/pubs/glossary/11th Glossary 2003.pdf](http://www.dau.mil/pubs/glossary/11th%20Glossary%202003.pdf))
- ASD (NII):** Assistant Secretary of Defense for Networks and Information Integration. (Source:<http://www.dod.mil/nii/>)
- ASP:** Active Server Page. A script that is executed by Microsoft Internet Information Services. The output is returned to the user as HTML. Typically, an ASP script generates a customized web page on the fly before sending it to the user. ASPs are specific to Microsoft, only run on IIS or PWS, can contain HTML, JScript, and VBScript, and can access COM components.
- asset:** Any sensor, weapon, aircraft, boat, unmanned air vehicle (UAV), etc., directly controlled by own ship.
- Associated Measurement Report (AMR):** A sensor measurement that has been processed by the originating sensor for clutter rejection and meets defined signal-to-noise parameters, and has been associated with either a local sensor track or a global composite track.
- association:** (1) The automatic or manual establishment of a relationship between two or more tracks when the information on them is deemed to pertain to the same contact. (2) The process of identifying and linking data sets that may correspond to the same object while retaining each track as an individual entity.
- assured sharing:** Trusted accessibility to net resources such as data, services, apps, people, and collaborative environments.
- ATAM:** Architecture Tradeoff Analysis Method (SEI). A risk mitigation method that can occur early in the software development life cycle when it is relatively inexpensive to change architectural decisions. (Source: [http://www.sei.cmu.edu/ata/ata\\_init2.html](http://www.sei.cmu.edu/ata/ata_init2.html))
- ATO:** Authority to Operate. An ATO or IATO is required prior to conducting operational tests on a deployable system. An ATO or IATO is granted only after the bulk of certification and accreditation activities are concluded, and the Designated Approving Authority (DAA) is satisfied with the residual risk to the system. (Source: [http://akss.dau.mil/dag/Guidebook/IG\\_c9.9.2.2.asp](http://akss.dau.mil/dag/Guidebook/IG_c9.9.2.2.asp))
- attribute:** A distinct characteristic of an object. Real-world object attributes are often specified in terms of their physical traits, such as size, shape, weight, and color. Cyberspace object

attributes might describe size, type of encoding, and network address.

(Source:<http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)

**attribute data:** Any non-kinematic data provided by a sensor for a track. Examples include IFF mode codes, INTEL data (e.g., imagery), EW data (e.g., parametric data), non-cooperative target recognition (NCTR) data, etc.

**authentication:** The process that verifies the identity of a user, device, or other entity in a computer system, usually as a prerequisite to allowing access to resources in a system. The Java servlet specification requires three types of authentication (basic, form-based, and mutual) and supports digest authentication. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**authorization:** The process by which access to a method or resource is determined. Authorization depends on the determination of whether the principal associated with a request through authentication is in a given security role. A security role is a logical grouping of users defined by the person who assembles the application. A deployer maps security roles to security identities. Security identities may be principals or groups in the operational environment. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**AWT:** Abstract Window Toolkit. The AWT is part of the Java Foundation Classes (JFC) -- the standard API for providing graphical user interfaces (GUIs) for Java programs.

## B

**B2B:** Business-to-Business integration

**baseline, allocated:** The initially approved documentation describing a system's functional, performance, interoperability, and interface requirements that are allocated from those of the system or higher level subsystem; interface requirements with interfacing subsystems; design constraints; derived functional and performance requirements; and verification requirements and methods to demonstrate the achievement of those requirements and constraints.

**baseline, functional:** The initially approved documentation describing a system's or configuration item's functional performance, interoperability, and interface requirements. Also, the verification required to demonstrate the achievement of those specified requirements.

**battle force:** A standing operational naval task force organization of carriers, surface combatants, and submarines assigned to numbered fleets. A battle force is subdivided into battle groups.

**BCD:** Binary Coded Decimal. Binary-coded decimal (BCD) is, after character encodings, the most common way of encoding decimal digits in computing and in electronic systems. In BCD, a digit is usually represented by four (binary) bits, of which the leftmost (written conventionally) has value 8, and the remaining three have values 4, 2, and 1. Only the combinations of these bits that, when summed, have values in the range 0-9 are valid. Other combinations are sometimes used for sign or other indications. (Source: [http://en.wikipedia.org/wiki/Binary\\_Coded\\_Decimal](http://en.wikipedia.org/wiki/Binary_Coded_Decimal))

**BCI:** Business Community Integration

**black box:** Provides a specified function or functions at a specified level of performance for an agreed-upon cost.

**BPEL:** Business Process Execution Language. BPEL is emerging as the standard for assembling a set of discrete services into an end-to-end process flow, radically reducing the cost and complexity of process integration initiatives. (Source: <http://www.oracle.com/technology/products/ias/bpel/index.html>)

**business logic:** The code that implements the functionality of an application. In the Enterprise JavaBeans architecture, this logic is implemented by the methods of an enterprise bean. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**business method:** A method of an enterprise bean that implements the business logic or rules of an application. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

## C

**C2:** Command and Control. The exercise of authority and direction by a properly designated commander over assigned forces in the accomplishment of the mission. C2 functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures. A commander employs these when planning, directing, coordinating, and controlling forces and operations in the accomplishment of the mission.

**C2ERA:** Command and Control Enterprise Reference Architecture. A technical concept of operations for building information systems better suited to the NCW environment. C2ERA prescribed the technical architecture mandated by the Designated Acquisition Commander for C4ISR Enterprise Integration in the U.S. Air Force. C2ERA is one of two NCW projects that merged to form NESI. The other project was RAPIDS.

**C2I:** Command, Control, and Intelligence / Command and Control Integration

**C2IEDM:** Command and Control Information Exchange Data Model. A data model that is managed by the Multilateral Interoperability Programme (MIP). It originated with experts from various NATO partners and from the Partnership-for-Peace nations. This data model is in the process of being submitted to OMG for consideration as the standard for information exchange. It falls under the shared operational picture exchange service. (Source: [http://www.mip-site.org/MIP\\_DMWG.htm](http://www.mip-site.org/MIP_DMWG.htm))

**C2W:** Command & Control Warfare

**C3:** Command, Control, & Communications

**C4I:** Command, Control, Communications, Computers, and Intelligence

**C4ISR:** Command, Control, Communications, Computers, and Intelligence, Surveillance, and Reconnaissance

**CA:** Certificate Authority. A trusted organization that issues public key certificates and provides identification to the bearer. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**capability on demand:** Delivery of and/or access to capabilities (data, applications, connectivity) incrementally and as needed, on demand, and controlled by user clearance.

**cascade delete:** A deletion that triggers another deletion. A cascade delete can be specified for an entity bean that has container-managed persistence. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**CCB:** Configuration Control Board. Also Change Control Board. Duties include reviewing change requests, making decisions, and communicating decisions made to affected groups and individuals. Represents the interests of program and project management by

- ensuring that a structured process is used to consider proposed changes and incorporate them into a specified release of a product.
- CCM:** CORBA Component Model. Part of the CORBA 3.0 Specification, CCM extends the CORBA object model and enforces composition rather than inheritance. Similar to a CORBA version of EJB that can be used with any language on any platform. (Source: <http://www.omg.org/technology/documents/formal/components.htm>)
- CDATA:** Character Data. A predefined XML tag for character data that means “don't interpret these characters,” as opposed to parsed character data (PCDATA), in which the normal rules of XML syntax apply. CDATA sections are typically used to show examples of XML syntax. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- CDD:** Capabilities Development Document. Provides operational performance attributes, including supportability, for the acquisition community to design the proposed system. Includes key performance parameters (KPP) and other parameters that guide the development, demonstration, and testing of the current increment. Outlines the overall strategy for developing full capability. (Source: <http://www.dau.mil/pubs/glossary/11thGlossary2003.pdf>)
- CDRL:** Contract Data Requirements List. A list of contract data requirements that are authorized for a specific acquisition and made a part of the contract. (Source: <http://www.dau.mil/pubs/glossary/11thGlossary2003.pdf>)
- CDS:** Cross Domain Security. User authentication across multiple application spaces.
- CES:** Core Enterprise Services. Generic information services that apply to any COI, provide the basic ability to search the enterprise for desired information, and then establish a connection to the desired service. (Source: [http://www.defenselink.mil/nii/org/cio/doc/GIG\\_ES\\_Core\\_Enterprise\\_Services\\_Strategy\\_V1-1a.pdf](http://www.defenselink.mil/nii/org/cio/doc/GIG_ES_Core_Enterprise_Services_Strategy_V1-1a.pdf))
- CGI script:** Common Gateway Interface. CGI is a standard for interfacing external applications with information servers, such as HTTP or web servers. A plain HTML document that the web daemon retrieves is static, which means it exists in a constant state: a text file that doesn't change. A CGI program, on the other hand, is executed in real time, so it can output dynamic information.
- CIM:** Common Information Model
- CJCS directive:** Chairman of the Joint Chiefs of Staff Directives. Instructions, Manuals, Notices, Guides, and other policy and procedures published by the Chairman, Joint Chiefs of Staff. (Source: [http://www.dtic.mil/cjcs\\_directives/index.htm](http://www.dtic.mil/cjcs_directives/index.htm))
- CJCSI:** Chairman of the Joint Chiefs of Staff Instruction
- class-based design:** Any design that incorporates objects and classes. Contrast with object-oriented design and objects-based design.
- class-based programming language:** A programming language that enables programmers to define and use objects and classes; for example, CLU. Contrast with object-based programming languages and object-oriented programming languages.
- CLI:** Command Line Interface. A method of interacting with a computer by giving it lines of textual commands (that is, a sequence of characters) either from keyboard input or from a script. (Source: [http://en.wikipedia.org/wiki/Command\\_line\\_interface](http://en.wikipedia.org/wiki/Command_line_interface))

- client:** A system entity that accesses a web service. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)
- client-certificate authentication:** An authentication mechanism that uses HTTP over SSL, in which the server and (optionally) the client authenticate each other with a public key certificate that conforms to a standard that is defined by X.509 Public Key Infrastructure. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- CLR:** Common Language Runtime. The CLR is the basis for everything in the Microsoft .NET Framework. It provides a standard implementation by providing a common set of datatypes (integers, strings, classes, interfaces), specifications for how inheritance works, and a common set of semantics for languages built on it. (Source: [http://en.wikipedia.org/wiki/Common\\_Language\\_Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime))
- CMM:** Capability Maturity Model
- cohesion:** The manner and degree to which the tasks performed by a single software module are related to one another. Types include coincidental, communicational, functional, logical, procedural, sequential, and temporal. Synonym: module strength. Contrast with coupling. In a well-designed, highly modular software design, the modules will have high cohesion; that is, each will have a clearly defined set of functions that have a close relationship to each other. This facilitates changes to modules since the changes will affect only the closely-related functions. In contrast, modules that contain multiple, unrelated functions blur the integrity of the software's design since the unrelated functions are bound into a single module, thereby creating dependencies that inhibit the ability to easily make changes. (Source: IEEE Std 610.12-1990)
- COI:** Community of Interest. A collection of people who exchange information using a common vocabulary in support of shared missions, business processes, and objectives. The community is made up of the users/operators who participate in the information exchange, the system builders who develop computer systems for these users, and the functional proponents who define requirements and acquire systems on behalf of the users.
- collaboration:** Portal members can communicate synchronously through chat or messaging, or asynchronously through threaded discussion, blogs, and email digests (forums).
- COM:** Component Object Model. A Microsoft software architecture for building component-based applications. COM objects are discrete components, each with a unique identity, which expose interfaces that allow applications and other components to access their features. COM objects are more versatile than Win32 DLLs because they are completely language-independent, have built-in interprocess communications capability, and easily fit into an object-oriented program design. COM was first released in 1993 with OLE2, largely to replace the interprocess communication mechanism DDE used by the initial release of OLE. ActiveX is based on COM.
- combat identification (CID):** CID is the process of attaining an accurate characterization of detected objects in the joint battlespace to the extent that high confidence and timely application of military options and weapons resources can occur. Depending on the situation, this characterization may be limited to "friend," "enemy," or "neutral." In other situations, other characterizations may be required, including, but not limited to, class, type, nationality, and mission configuration.
- commercial software:** "Commercial software is software developed by businesses which aim to make money from its use. Most commercial software is proprietary, but there is

commercial free software, and there is non-commercial non-free software.” (Source: GNU.org: Categories of Free and Non-Free Software: <http://www.gnu.org/philosophy/categories.html>)

**commit:** The point in a transaction when all updates to any resources involved in the transaction are made permanent. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**common language runtime:** The common language runtime (CLR) is a high-performance engine for running applications built using the .NET Framework. Code that targets the runtime and whose execution is managed by the runtime is referred to as managed code. Responsibility for tasks such as creating objects, making method calls, and so on is delegated to the CLR, which enables it to provide additional services to the code as it executes. While the component is running, the CLR provides services—such as memory management (including garbage collection), process management, thread management, and security enforcement—and satisfies any dependencies that the component may have on other components. (Source: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbd/html/guidenetapp.asp>)

**Community of Interest (COI) service:** A service that may be offered to the enterprise, but is owned and operated by a Community of Interest to provide or support a well-defined set of mission functions and associated information.

**compiler:** “A computer program that translates programs expressed in a high-order language into their machine language equivalent.” (Source: IEEE Std 610.12-1990)

**component:** “One of the parts that make up a system. A component may be hardware or software and may be subdivided into other components. Note the terms 'module,' 'component,' and 'unit' are often used interchangeably or defined to be subelements of one another in different ways depending on the context. The relationship of these terms is not yet standardized.” (Source: IEEE Std 610.12-1990) “A product that is not subject to decomposition from the perspective of a specific application.”(Source: ISO 10303-1)

**component-based software:** Mission applications that are architected as components integrated within a component framework

**component, system:** A basic part of a system. System components may be personnel, hardware, software, facilities, data, material, services, and/or techniques that satisfy one or more requirements in the lowest levels of the functional architecture. System components may be subsystems and/or configuration items.

**composite/collaborative track:** A representation of an entity that is formed by combining individual instances of measurement data or a collection of measurements from one or more sensors into a single composite/collaborative track state vector and combined attribute information.

**conceptual model:** Captures the concepts of the relational database and can help enforce the first three normalization rules

**condition:** A variable of the operational environment or situation in which a unit, system, or individual is expected to operate that may affect performance.

**connection pooling:** A technique for establishing a pool of resource connections that applications can share on an application server.

**connector:** A portable service API to external resources.

**CONOPS:** Concept of Operations

- consumer:** A system entity invoking producers in a manner conforming to a specification. For example, a portal aggregating content from portlets accessed using the WSRP protocol is a type of consumer. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)
- container:** A standard extension mechanism for containers that provides connectivity to enterprise information systems. A connector is specific to an enterprise information system. It consists of a resource adapter and application development tools for enterprise information system connectivity. The resource adapter is plugged in to a container through its support for system-level contracts defined in the Connector architecture. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- content and document management:** Services that support the full lifecycle of document creation and provide mechanisms for authoring, approval, version control, scheduled publishing, indexing, and searching.
- copyright:** The U.S. Copyright Act, 17 U.S.C. §§ 101 – 810 (17 U.S.C. §§ 101 - 810), is Federal legislation enacted by Congress under its Constitutional grant of authority to protect the writings of authors. The Copyright Act covers architectural design, software, the graphic arts, motion pictures, and sound recordings (§ 106). A copyright gives the owner the exclusive right to reproduce, distribute, perform, display, or license his work. The owner also receives the exclusive right to produce or license derivatives of his or her work (§ 201(d)). Limited exceptions to this exclusivity exist for types of “fair use,” such as book reviews (§ 107). To be covered by copyright a work must be original and in a concrete “medium of expression” (§ 102). Under current law, works are covered whether or not a copyright notice is attached and whether or not the work is registered.
- CORBA®:** Common Object Request Broker Architecture. CORBA “wraps” code written in another language into a bundle containing additional information on the capabilities of the code inside, and explaining how to call it. The resulting wrapped objects can then be called from other programs (or CORBA objects) over the network. The CORBA specification defines APIs, communication protocol, and object/service information models to enable heterogeneous applications written in various languages running on various platforms to interoperate. (Source: <http://en.wikipedia.org/wiki/CORBA>)
- core enterprise service:** A ubiquitous, common solution service that provides capabilities essential to the operation of the enterprise.
- correlation:** (1) The determination that a locally derived track represents the same object or point as another track, and/or the process of combining two such tracks/data under one track number. (Logicon) (2) The process of identifying tracks believed to represent the same object and replacing them with a single track, combining the data from the duplicate tracks as appropriate.
- CoS:** Class of Service. A queuing discipline. The algorithm compares fields of packets or CoS tags to classify packets in different priority queues. CoS does not ensure network performance or certain priority in delivering packets. See also Quality of Service(QoS). (Source: [http://en.wikipedia.org/wiki/Class\\_of\\_service](http://en.wikipedia.org/wiki/Class_of_service))
- COTS:** Commercial Off-The-Shelf. A term for systems that are manufactured commercially, and may be tailored for specific uses. (Source: [http://en.wikipedia.org/wiki/Commercial\\_off-the-shelf](http://en.wikipedia.org/wiki/Commercial_off-the-shelf))
- coupling:** The manner and degree of interdependence between software modules. Types include common-environment coupling, content coupling, control coupling, data coupling, hybrid

- coupling, and pathological coupling. Contrast with cohesion. In a well-designed, highly modular software design, the coupling between modules will be minimized. This facilitates changing and replacing modules with minimal effect on other modules within the system. (Source: IEEE Std 610.12-1990)
- CPD:** Capabilities Production Document. Addresses the production attributes and quantities specific to a single increment of an acquisition program. Supersedes threshold and objective performance values of the CDD. (Source: [http://www.dau.mil/pubs/glossary/11thGlossary\\_2003.pdf](http://www.dau.mil/pubs/glossary/11thGlossary_2003.pdf))
- CRD:** Capstone Requirements Document. A document containing capabilities-based requirements that facilitates the development of individual Capability Development Documents (CDDs) by providing a common framework and operational concept to guide their development. CRDs that have been approved by the Joint Requirements Oversight Council (JROC) continue to be valid until absorbed into appropriate integrated architectures as required by CJCSI 3170.01C and retired. The JROC retains the authority to specifically direct the development of new CRDs, as necessary. The CRD format is contained in CJCSM 3170.01.(CJCSI 3170.01C and CJCSM 3170.01). (Source: [http://www.dau.mil/pubs/glossary/11th\\_Glossary\\_2003.pdf](http://www.dau.mil/pubs/glossary/11th_Glossary_2003.pdf))
- credentials:** The information describing the security attributes of a principal. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- cross compiler:** “A compiler that executes on one computer but generates machine code for a different computer.” (Source: IEEE Std 610.12-1990)
- CSM:** Component and Service Management. A set of management capabilities for monitoring and controlling deployed applications, their components, and web services. CSM collects data, analyzes it, and makes system management recommendations to operators. CSM also provides the ability to manage version configuration information and a scheduler to run batch jobs at a predetermined schedule. Other CSM capabilities include configuration management, end-to-end performance monitoring and analysis, service desk support, software distribution, service life-cycle management, and quality-of-service management.
- CSS:** Cascading Style Sheet. A stylesheet used with HTML and XML documents to add a style to all elements marked with a particular tag, for the direction of browsers or other presentation mechanisms. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>) -OR- Closed Source Software. Software in which the source code is not open and available, e.g., COTS (commercial off-the-shelf) software. COTS is usually distributed in a binary form. With COTS licenses, the purchaser is not allowed to take apart or reverse-engineer the product, or modify the product for any purpose. Other forms of CSS include shareware and royalty-free libraries (e.g., runtime libraries for compilers). CSS may come with source code, but the associated licenses forbid the creation and distribution of any derived works. (Source: [http://en.wikipedia.org/wiki/Closed\\_source](http://en.wikipedia.org/wiki/Closed_source))
- customized application:** An application that can be tailored on a continuing basis to meet current Rules of Engagement (ROE) and readjusted to meet tomorrow’s needs.
- customized delivery:** Smart push-and-pull of data reduces overload and provides the requested data to operators when they need it. Tailored discovery, publish, and subscribe capabilities allow operators to register for specific data and services in specific timeframes.

## D

- DAC:** Discretionary Access Control. Defines basic access control policies to objects in a file system. Generally, these are done at the discretion of the object owner: file/directory permissions and user/group ownership. (Source: [http://en.wikipedia.org/wiki/Discretionary\\_access\\_control](http://en.wikipedia.org/wiki/Discretionary_access_control))
- data-centric:** Data separated from applications; apps talk to each other by posting data.
- data asset:** Any entity that involves data.
- data exchange:** Operators can move data between applications easily and without losing data or capabilities. Data may carry security labels allowing for its exchange with partners operating at coalition or multinational releasable security levels.
- data modeling:** Modeling is an essential step in understanding the data that will comprise a system. The end products of data modeling can be XML schemas or RDBMS schema definitions. Many COIs create their own data models, such as C2IEDM for the C2 community.
- DBMS:** Database Management System. A system, usually automated and computerized, for managing any collection of compatible, and ideally normalized, data. (Source: <http://en.wikipedia.org/wiki/DBMS>)
- DCID:** Director of Central Intelligence Directive. CIA publications that provide timely, coordinated, and clear guidance and direction to the Intelligence Community.
- DCTS:** Defense Collaboration Tool Suite. A flexible, integrated set of applications providing interoperable, synchronous, and asynchronous collaboration capability to the Department of Defense's (DoD) agencies, Combatant Commands, and military services. (Source:<http://www.disa.mil/main/prodsol/dcts.html>)
- DDMS:** DoD Discovery Metadata Specification. A NCES metadata initiative. DDMS defines discovery metadata elements for resources posted to community and organizational shared spaces. Sometimes (incorrectly) referred to as DoD Discovery Metadata Standard. (Source:<http://diides.ncr.disa.mil/mdreg/user/DDMS.cfm>)
- decorrelation:** The determination that locally held track data for a given track number does not represent the same object or point as track data being received in a remote track report for the same track number.
- deployment:** The process whereby software is installed into an operational environment. (Source:<http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- deployment descriptor:** An XML file provided with each module and J2EE application that describes how they should be deployed. The deployment descriptor directs a deployment tool to deploy a module or application with specific container options and describes specific configuration requirements that a deployer must resolve. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- design architecture:** An arrangement of design elements that provides the design solution for a product or life cycle process intended to satisfy the functional architecture and the requirements baseline. (Source: IEEE 1220)
- design requirement:** "A requirement that specifies or constrains the design of a system component." (Source:IEEE Std 610.12-1990)
- DFARS:** Defense Federal Acquisition Regulation Supplement
- DHTML:** Dynamic HTML. Designates a technique of creating interactive web sites by using a combination of the static markup language HTML, a client-side scripting language such

as JavaScript, and the style definition language Cascading Style Sheets.  
(Source:[http://en.wikipedia.org/wiki/Dynamic\\_web\\_page](http://en.wikipedia.org/wiki/Dynamic_web_page))

**DIACAP:** DoD Information Assurance Certification and Accreditation Program

**DiffServ:** Differentiated Services

**DII:** Dynamic Invocation Interface

**DISA:** Defense Information Systems Agency

**disconnected application:** An application that may not be available at all times. Not all applications within the enterprise will have a 24/7 connection to the other machines in the network. For example, consider a submarine that surfaces several times a day to obtain mission information. A message-base system can store the messages in a queue until the submarine surfaces. Disconnected applications allow the receiving application to process messages at any time. As a result, the sender and receiver are not as dependent on each other.

**DISN:** Defense Information System Network

**DISR:** Defense Information Technology Standards & Profiles Registry

**distributed application:** An application made up of distinct components running in separate runtime environments, usually on different platforms connected via a network. Typical distributed applications are two-tier (client-server), three-tier (client-middleware-server), and multitier (client-multiple middleware-multiple servers). (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**DITSCAP:** Defense Information Technology Security Certification and Accreditation Process

**DoD:** Department of Defense

**DoDAF:** DoD Architecture Framework

**DoDD:** Department of Defense Directive

**DoDI:** Department of Defense Instruction

**DoDIIS:** Department of Defense Intelligence Information System

**DOM:** Document Object Model. An API for accessing and manipulating XML documents as tree structures. DOM provides platform-neutral, language-neutral interfaces that enable programs and scripts to dynamically access and modify content and structure in XML documents. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**domain:** A group of related items within a certain area of interest.

**domain analysis:** The process of identifying the types of information that the data model uses. A good data model captures descriptive information about each of the types.

**DON:** Department of the Navy

**DOS:** Data-Oriented Services. A software component that receives a request and optionally returns an XML data response to a UFS or another DOS. A DOS has no visual or presentation component.

**DOTMLPF:** Doctrine, Organization, Training, Materiel, Leadership, Personnel, Facilities

**DRR:** Design Readiness Review

**DSI:** Data Source Interface

**DTD:** Document Type Definition. An optional part of the XML document prolog, as specified by the XML standard. The DTD specifies constraints on the tags and tag sequences that can be in the document. The DTD has a number of shortcomings, however, and this has led to various schema proposals. For example, the DTD entry `<!ELEMENT username(#PCDATA)>` says that the XML element called “username” contains parsed character data; that is, text alone, with no other structural elements under it. The DTD includes both the local subset, defined in the current file, and the external subset, which consists of the definitions contained in external DTD files that are referenced in the local subset using a parameter entity. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**dual stacking:** Incorporating both IPv4 and IPv6 support in routers and computers.

**dynamic web page:** See DHTML.

## E

**EAI:** Enterprise Application Integration

**EAR:** Enterprise Application Archive. A JAR archive that contains a J2EE application. It contains all the JAR, WAR, and RAR archives for an enterprise application, plus an XML descriptor. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**ebXML:** Electronic Business XML. A group of specifications designed to enable enterprises to conduct business through the exchange of XML-based messages. It is sponsored by OASIS and the United Nations Centre for the Facilitation of Procedures and Practices in Administration, Commerce and Transport (U.N./CEFACT). (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**EIP:** Enterprise Information Portals. An EIP is an intranet portal, usually under the control of a single domain such as the DoD. Some of the features of an EIP are single touch point, collaboration, content and document management, personalization, and integration. (Source: [http://en.wikipedia.org/wiki/web\\_portal](http://en.wikipedia.org/wiki/web_portal))

**EJB:** Enterprise Java Bean. A server-side component architecture for the development and deployment of object-oriented, distributed, enterprise-level applications. Applications written using the Enterprise JavaBeans architecture are scalable, transactional, and secure. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**embedded style sheet:** A style sheet in the heading of an HTML document. They override linked style sheets.

**end user:** A human user of information. This is distinct from those who develop or support the automated systems that provide the information. -OR- A person who uses a device-specific user agent to access a web site. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)

**endpoint:** The URL or location of the web service on the internet.

**enterprise:** An organization considered as an entity or system that includes interdependent resources (e.g., people, organizations, and technology) that must coordinate functions and share information in support of a common mission or a set of related missions.

**enterprise guidelines:** Rules that govern the choice/implementation of COI Enterprise Services.

**enterprise service:** A service that provides capabilities to the enterprise. See also Core Enterprise Service and Community of Interest Service.

**entity bean:** An enterprise bean that represents persistent data maintained in a database. An entity bean can manage its own persistence or can delegate this function to its container. An entity bean is identified by a primary key. If the container in which an entity bean is hosted crashes, the entity bean, its primary key, and any remote references survive the crash. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**ESC:** Electronic Systems Center

**ESM:** Enterprise Service Management

**Ethernet:** A network communication system developed and standardized by DEC, Intel, and Xerox, using baseband transmission, CSMA/CD access, logical bus topology, and coaxial cable. The successor IEEE 802.3 standard provides for integration into the OSI model. It extends the physical layer and media with repeaters and implementations that operate on fiber, thin coax, and twisted-pair cable. (Source: <http://www.sun.com/products-n-solutions/hardware/docs/html/817-6210-10/glossary.html>)

**EUC:** Extended User Community

**event-driven:** An application that responds to events.

**event-driven application:** An application that responds to events. For example, a weather-reporting application may respond to weather sensor events. Since message-base systems are inherently asynchronous, synchronization is not an issue for application development. An application can simply put the message in the queue and not have to wait for a response. This decoupling allows applications to be more responsive and operate independently of time constraints.

**execution architecture:** An execution architecture is created for distributed or concurrent systems. The process view shows the mapping of components onto the processes of the physical system. The deployment view shows the mapping of (physical) components in the executing system onto the nodes of the physical system.

**external style sheet:** See linked style sheet.

**external time source:** Synchronizes internal clocks across BF platforms and represents the source of UTC time for the above system time.

## F

**façade:** Provides a unified interface to a set of interfaces in a subsystem. Façade defines a higher-level interface that makes the subsystem easier to use. This can simplify a number of complicated object interactions into a single interface.

**FAR:** Federal Acquisition Regulation

**FCS:** Future Combat Systems

**feel aspect:** One of the traditional aspects of a graphical user interface. The "feel" covers the behavior of dynamic elements such as buttons, boxes, and menus

**FMS:** Foreign Military Sales

**force:** (1) An aggregation of military personnel, weapon systems, vehicles, and necessary support, or combination thereof. (2) A major subdivision of a fleet.

**FORCEnet:** An operational construct and architectural framework that integrates the SEAPOW21 concepts of Sea Strike, Sea Shield, and Sea Basing by connecting

warriors; sensors, networks; command and control; platforms and weapons; providing accelerated speed and accuracy of decision; and integrating knowledge to dominate the battlespace. FORCENet provides the following capabilities: expeditionary, multi-tiered, sensor and weapon grids; distributed, collaborative, command and control; dynamic, multi-path survivable networks; adaptive/automated decision aids; and human-centric integration.

**foreign key:** An attribute in a relation of a database that serves as the primary key of another relation in the same database.

**free software:** Free software is software whose license terms do not restrict the users in the ways that they can run, copy, distribute, study, change, and improve the software. By definition, free software is open-source. In this definition, “free” refers not to the cost of acquiring or using the software but rather to how the software can be used. (Source: GNU.org: <http://www.gnu.org/philosophy/free-sw.html>)

**freeware:** The term “freeware” has no single definition, but is commonly used to refer to software whose license terms permit redistribution but not modification. Usually, the source code for freeware is not available. (Source: GNU.org: <http://www.gnu.org/philosophy/categories.html>)

**FTP:** File Transfer Protocol. FTP transfers files to and from a remote network. The protocol includes the ftp command (local machine) and the in.ftpd daemon (remote machine). FTP enables a user to specify the name of the remote host and file transfer command options on the local host's command line. The in.ftpd daemon on the remote host then handles the requests from the local host. Unlike rcp, ftp works even when the remote computer does not run a UNIX-based operating system. A user must log in to the remote computer to make an ftp connection unless it has been set up to allow anonymous FTP. (Source: <http://www.sun.com/products-n-solutions/hardware/docs/html/817-6210-10/glossary.html>)

**functional analysis:** Examination of a defined function to identify all the subfunctions necessary to the accomplishment of that function. Identification of functional relationships and interfaces (internal and external) and the capture of these in a functional architecture. The flow down of upper-level performance requirements and the assignment of these requirements to lower-level sub-functions.

**functional architecture:** An arrangement of functions and their subfunctions and interfaces (internal and external) that defines the execution sequencing, conditions for control or data flow, and the performance requirements to satisfy the requirements baseline. (Source: IEEE 1220)

**functional requirements:** Specific actions that a system must be able to perform, without taking physical constraints into consideration. These are often best described in a use-case model and in use cases. Functional requirements specify the input and output behavior of a system.

## G

**GCCS:** Global Command and Control System

**GCCS-M:** Global Command and Control System – Maritime. GCCS-M [AN/USQ-119E(V)], previously the Joint Maritime Command Information System (JMCIS), is the Navy's primary fielded Command and Control System. It is a globally interconnected, end-to-end set of information capabilities, associated processes, and personnel. It collects,

processes, stores, disseminates, and manages information on demand to warfighters, policy makers, and support personnel. It uses this data to execute the full range of Navy missions (e.g., strategic deterrence, sea control, power projection, etc.) in near-real-time via external communication channels, local area networks (LANs), and direct interfaces with other systems.

**GCSS:** Global Combat Support System

**GIG:** Global Information Grid. Globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating, and managing information on demand to warfighters, policy makers, and support personnel. The GIG includes all owned and leased communications and computing systems and services, software (including applications), data, security services, and other associated services necessary to achieve Information Superiority. It also includes National Security Systems (NSS) as defined in section 5142 of the Clinger-Cohen Act of 1996. The GIG supports all DoD, National Security, and related Intelligence Community (IC) missions and functions (strategic, operational, tactical, and business) in war and in peace. The GIG provides capabilities from all operating locations (bases, posts, camps, stations, facilities, mobile platforms, and deployed sites). The GIG provides interfaces to coalition, allied, and non-DoD users and systems.

**GIG-BE:** Global Information Grid Bandwidth Expansion

**GIG-ES:** Global Information Grid Enterprise Services

**GIG enterprise service:** A service that provides capabilities for use in the DoD enterprise. GIG Enterprise Services are the combination of Core Enterprise Services and Community of Interest Services. Also referred to as Global Enterprise Services.

**GIOP:** General InterORB Protocol

**GO-1:** Geographic Objects Initiative, Phase 1. Interoperability initiative and specification from Open GIS Consortium on GIS APIs. (Source: <http://ip.opengis.org/go1/>)

**GOTS:** Government Off-The-Shelf

**GPL:** General Public License. A license that defines a specific set of distribution terms for free software. A GPL specifically does not let redistributors add any additional restrictions when they redistribute or modify the software. This means that every copy of the software, even if it has been modified, must be free software.

(Source: <http://www.gnu.org/copyleft/gpl.html>)

**group:** An authenticated set of users classified by common traits such as job title or customer profile. Groups are also associated with a set of roles, and every user that is a member of a group inherits all the roles assigned to that group. (Source:

<http://java.sun.com/j2ee/1.4/docs/glossary.html>) -OR- (1) A flexible administrative and tactical unit composed of either two or more battalions or two or more squadrons. The term also applies to combat support and combat service support units. (2) A number of ships and/or aircraft, normally a subdivision of a force, assigned for a specific purpose.

**GUI:** Graphical User Interface

## H

**HAIPE:** High Assurance Internet Protocol Encryptor

**hard real-time:** A system is said to be hard real-time if the correctness of an operation depends not only upon the logical correctness of the operation but also upon the time at which it is performed. An operation performed after the deadline is, by definition, incorrect, and usually has no value. In a soft real-time system the value of an operation declines steadily after the deadline expires. (Source: [http://en.wikipedia.org/wiki/Real\\_time](http://en.wikipedia.org/wiki/Real_time))

**heterogeneous replication:** Data transfer between the same or different RDBMS vendors: for example, Oracle to Oracle, or Oracle to Sybase to SQL Server to MySQL. Heterogeneous replication is proprietary to the heterogeneous vendor but reduces the dependency on a specific RDBMS vendor.

**hierarchical database:** A hierarchical database defines a set of parent-child relationships. Their use should be limited to integration of existing databases, such as IBM's Informational Management System (IMS). Hierarchical database systems require developers to predict all possible access patterns in advance and design the database accordingly. A database access pattern that is not included in the design becomes very difficult and inefficient.

**high-order language:** "Any programming language that requires little knowledge of the computer hardware on which a program will run, can be translated into several different machine languages, allows symbolic naming of operations and addresses, provides features designed to facilitate expression of data structures and program logic, and usually results in several machine instructions for each program statement. Examples include Ada, ALGOL, COBOL, FORTRAN, Pascal." (Source: IEEE Std 610.13-1993. IEEE Standard Glossary of Computer Languages)

**high availability:** Data tier availability can be affected by hardware failure, power outages, data errors, user errors, programmer errors, OS errors, and RDBMS errors. Various hardware and software methods help mitigate availability issues. The more reliable a system needs to be, the more it costs. Consequently, defining availability to meet requirements is essential to controlling costs.

**homogeneous replication:** Data transfer between two databases that are implemented using the same RDBMS provider: for example, between two Sybase or two Oracle RDBMSs.

**HTML:** Hypertext Markup Language. A markup language for hypertext documents on the Internet. HTML supports embedding images, sounds, video streams, form fields, references to other objects with URLs, and basic text formatting. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**HTTP:** Hypertext Transfer Protocol. The Internet protocol used to retrieve hypertext objects from remote hosts. HTTP messages consist of requests from client to server and responses from server to client. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**HTTPS:** In XML documents, a piece of text that describes a unit of data or an element. The tag is distinguishable as markup, as opposed to data, because it is surrounded by angle brackets (< and >). To treat such markup syntax as data, you use an entity reference or a CDATA section. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

## I

**IA:** Information Assurance. The set of measures taken to protect and defend information and information systems to ensure confidentiality, integrity, availability, and accountability, extended to restoration with protect, detect, monitor, and react capabilities.

**IABM:** Integrated Architecture Behavior Model

**IAI:** Internet Application Integration

**IAS:** Information Assurance and Security Measures

**IAW:** In accordance with

**IC:** Intelligence Community

**ICD:** Initial Capabilities Document

**ICSF:** Integrated C4I System Framework. Defines capability gaps in terms of functional area(s), relevant range of military ops, time, obstacles to overcome, and key attributes, with appropriate measures of effectiveness. Recommends materiel approach(es) based on cost analysis, efficacy, sustainability, environmental quality impacts, and associated risks.

**ID:** Identification. (1) Identification is the Identity, Category, Platform, Type, Activity, and Nationality/Alliance of the track. (2) The process of determining the friendly or hostile character of an unknown detected contact.

**IDE:** Integrated Development Environment

**identity:** Identity refers to the nature or attributes of the track: Friend, Assumed Friend, Neutral, Unknown, Pending, Suspect, or Hostile.

**IDL:** Interface Definition Language. A language used to define interfaces to remote CORBA objects. The interfaces are independent of operating systems and programming languages. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**IEC:** International Engineering Consortium

**IEEE:** International Electrical and Electronics Engineers

**IETF:** Internet Engineering Task Force

**IFR:** Interface Repository

**IIOP:** Internet InterORB Protocol. A protocol used for communication between CORBA object request brokers. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**IIS:** Internet Information Services. A set of Internet-based services for Windows machines. Originally supplied as part of the Option Pack for Windows NT, they were subsequently integrated with Windows 2000 and Windows Server 2003. The current (Windows 2003) version is IIS 6.0 and includes servers for FTP, SMTP, NNTP and HTTP/HTTPS. Earlier versions also included a Gopher server.

**implementation requirement:** "A requirement that specifies or constrains the coding or construction of a system or system component." See also requirements. (Source: IEEE Std 610.12-1990)

**incremental upgrade:** Certain capabilities can be modernized without impacting other capabilities.

**INE:** Inline Network Encryptor

**inline style sheet:** A style sheet in an individual HTML tag. It overrides linked and embedded style sheets.

**integration:** Integration is the action or process of combining elements so that they become a whole. Vertical integration acts within a system, whereas horizontal integration acts between or among systems. In the net-centric environment, integration creates links between computer systems, applications, services, or processes. The word is normally

used in the context of computing, but can apply to business processes as much as to the underlying process automation. In the past, computer integration such as enterprise application integration (EAI) has typically been tightly coupled, or “hard wired,” making it difficult to adapt to changing requirements. Thanks to the advent of web services and the evolution of service-oriented architectures, more agile, loosely coupled forms of integration are starting to emerge.

**INTEL-generated track:** Track based on INTEL data that is of sufficient quality for correlation/association with a system track.

**intellectual property** : The products resulting from intellectual effort and covered by a set of laws governing use of these products. These laws cover patents, copyrights, and trade secrets, and are conveyed by specific license terms and conditions describing allowable use. See also software licenses, software patents, copyrights.

**interface:** The functional and physical characteristics required to exist at a common boundary or connection between systems or items. (Source: DoD 4120.214-M)

**interface requirement:** “A requirement that specifies an external item with which a system or system component must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction.” (Source: IEEE Std 610.12-1990)

**interface standard:** A standard that specifies the physical, functional, and operational relationships between various hardware and software elements to permit interchangeability, interconnection, compatibility and/or communications.

**Internet:** The Internet, or simply the Net, is the publicly available worldwide system of interconnected computer networks that transmit data by packet switching using a standardized Internet Protocol (IP) and many other protocols. It is made up of thousands of smaller commercial, academic, and government networks. It carries various information and services, such as electronic mail, online chat and the interlinked web pages and other documents of the World Wide web. Because this is by far the largest, most extensive internet (with a small i) in the world, it is simply called the Internet (with a capital I). (Source: <http://en.wikipedia.org/wiki/Internet>)

**interoperability:** The ability of systems, units, or forces to (1) provide data, information, materiel, and services to, and accept the same from, other systems, units, or forces, and (2) to use the data, information, materiel, and services so exchanged to enable them to operate effectively together. IT and NSS interoperability includes both the technical exchange of information and the end-to-end operational effectiveness of that exchange of information as required for mission accomplishment. Interoperability is more than just information exchange. It includes systems, processes, procedures, organizations, and missions over the life cycle and must be balanced with information assurance. -OR- The ability for entities to work with each other. In the loosely coupled environment of a service-oriented architecture, separate resources don't need to know the details of how they each work, but they need to have enough common ground to reliably exchange messages without error or misunderstanding. Standardized specifications go a long way towards creating this common ground, but differences in implementation may still cause breakdowns in communication. Interoperability is when services can interact with each other without encountering such problems.

**intranet:** An intranet is a local area network (LAN) used internally in an organization to facilitate communication and access to information that is sometimes access-restricted. Sometimes the term refers only to the most visible service, the internal web site. The same concepts and technologies of the Internet such as clients and servers running on the

Internet protocol suite are used to build an intranet. HTTP and other internet protocols are commonly used as well, especially FTP and email. There is often an attempt to use internet technologies to provide new interfaces with corporate "legacy" data and information systems. (Source: <http://en.wikipedia.org/wiki/Intranet>)

**IP:** Internet Protocol. Data packets routed across network, not switched via dedicated circuits.

**IPv4:** Internet Protocol Version 4. Version 4 of the Internet Protocol (IP). It was the first version of the Internet Protocol to be widely deployed, and forms the basis for most of the current Internet (as of 2004). It is described in IETF RFC 791, which was first published in September, 1981. IPv4 uses 32-bit addresses, limiting it to 4,294,967,296 unique addresses, many of which are reserved for special purposes such as local networks or multicast addresses. This reduces the number of addresses that can be allocated as public Internet addresses. As the number of addresses available is consumed, an IPv4 address shortage appears to be inevitable in the long run. This limitation has helped stimulate the push towards IPv6, which is currently in the early stages of deployment, and may eventually replace IPv4. (Source: <http://en.wikipedia.org/wiki/IPv4>)

**IPv6:** Internet Protocol Version 6. Version 6 of the Internet Protocol; it was initially called IP Next Generation (IPng) when it was picked as the winner in the IETF's IPng selection process. IPv6 is intended to replace the previous standard, IPv4, which only supports up to about 4 billion ( $4 \times 10^9$ ) addresses. IPv6 supports up to about  $3.4 \times 10^{38}$  (340 undecillion) addresses. This is the equivalent of  $4.3 \times 10^{20}$  (430 quintillion) addresses per square inch ( $6.7 \times 10^{17}$  (670 quadrillion) addresses/mm<sup>2</sup>) of the Earth's surface. It is expected that IPv4 will be supported until at least 2025, to allow time for bugs and system errors to be corrected. (Source: <http://en.wikipedia.org/wiki/IPv6>)

**ISO:** International Standards Organization

**ISP:** Integrated Support Plans. Describes system dependencies and interface requirements. Includes system interface descriptions, infrastructure and support requirements, standards profiles, performance measures, and interoperability issues.

**ISR:** Intelligence, Surveillance, and Reconnaissance

**IT:** Information Technology

**ITIL:** Information Technology Infrastructure Library

**ITSM:** IT Service Management

**ITU:** International Telecommunication Union

## J

**J2A:** Java Connector Architecture

**J2EE:** Java 2 Enterprise Edition. The J2EE environment is the standard for developing component-based multi-tier enterprise applications. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, web-based applications. Features include web-services support and development tools. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**J2EE application:** Any deployable unit of J2EE functionality. This can be a single J2EE module or a group of modules packaged into an EAR file along with a J2EE application

deployment descriptor. J2EE applications are typically engineered to be distributed across multiple computing tiers. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**J2EE component:** A self-contained functional software unit that is supported by a container and is configurable at deployment time. The J2EE specification defines the following J2EE components. (1) Application clients and applets are components that run on the client. (2) Java servlet and JavaServer Pages (JSP) technology components, web components that run on the server. (3) Enterprise JavaBeans (EJB) components (enterprise beans), business components that run on the server. J2EE components are written in the Java programming language and are compiled in the same way as any program in the language. The difference between J2EE components and “standard” Java classes is that J2EE components are assembled into a J2EE application, verified to be well formed and in compliance with the J2EE specification, and deployed to production, where they are run and managed by the J2EE server or client container. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**J2EE module:** A software unit that consists of one or more J2EE components of the same container type and one deployment descriptor of that type. There are four types of modules: EJB, web, application client, and resource adapter. Modules can be deployed as standalone units or can be assembled into a J2EE application. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**J2EE server:** The runtime portion of a J2EE product. A J2EE server provides EJB or web containers or both. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**JAAS:** Java Authentication and Authorization Service

**JAD:** Joint Application Development

**JAR:** Java Archive. A platform-independent file format that enables you to bundle multiple files into a single archive file. JAR files are packaged with the ZIP file format, so you can use them for ZIP-like tasks such as lossless data compression, archiving, decompression, and archive unpacking. Typically JAR files contain the class files and auxiliary resources associated with applets and applications. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**Java Server Faces:** A framework for building user interfaces for web applications. It includes (1) A set of APIs for representing UI components and managing their state, handling events and input validation, defining page navigation, and supporting internationalization and accessibility; (2) A JavaServer Pages (JSP) custom tag library for expressing a JavaServer Faces interface within a JSP page.

**JavaBean:** A specification developed by Sun Microsystems that defines how Java objects interact and is similar to an ActiveX control. It can be used by any application that understands the JavaBean format.

**JavaMail:** A platform- and protocol-independent framework for building Java-based mail client applications.

**JavaScript:** The Netscape-developed object scripting language used in millions of web pages and server applications worldwide. Contrary to popular misconception, JavaScript is not "Interpretive Java." Rather, it is a dynamic scripting language that supports prototype-based object construction.

**JC2:** Joint Command and Control

**JCA:** J2EE Connector Architecture -OR- Java Cryptography Architecture

**JCIDS:** Joint Capabilities Integration and Development System

**JDBC:** Java Database Connection. An API that supports database and data-source access from Java applications.

**JDK:** Java Development Kits

**JEDI:** Joint Enterprise DoDIIS Infrastructure

**JFC:** Joint Forces Commander

**JIC:** Joint Intelligence Center

**JMS:** Java Message Service. An API for invoking operations on enterprise messaging systems. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**JMS client:** A Java-base application or object that produces and consumes messages, where messages are objects that contain the data being transferred between JMS clients.

**JMS connection class:** Once a connection factory is obtained, a connection to a JMS provider (MOM) can be created. A connection represents a communication link between the application and the messaging server. Depending on the connection type, connections allow users to create sessions for sending and receiving messages from a queue or topic.

**JMS connection factory class:** An administered object that a client uses to create a connection to the JMS provider (MOM). JMS clients access the connection factory through portable interfaces so the code does not need to be changed if the underlying implementation (MOM) changes. Administrators configure the connection factory in the Java Naming and Directory Interface (JNDI) namespace so that JMS clients can look them up. Depending on the type of message, users will use either a queue connection factory or topic connection factory.

**JMS destination class:** An administered object that encapsulates the identity of a message destination, which is where messages are delivered and consumed. It is either a queue or a topic. The JMS administrator creates these objects, and users discover them using JNDI. Like the connection factory, the administrator can create two types of destinations: queues for Point-to-Point and topics for Publish/Subscribe.

**JMS message consumer class:** An object created by a session. It receives messages sent to a destination. The consumer can receive messages synchronously (blocking) or asynchronously (non-blocking) for both queue and topic-type messaging.

**JMS message producer class:** An object created by a session that sends messages to a destination. The user can create a sender to a specific destination or create a generic sender that specifies the destination at the time the message is sent.

**JMS messages:** Objects that contain the data being transferred between JMS clients. Java base applications or objects that produce and consume messages, where messages are objects that contain the data being transferred between JMS clients.

**JMS messages class:** An object that is sent between consumers and producers; that is, from one application to another. A message has three main parts: (1) A message header (required): Contains operational settings to identify and route messages; (2) A set of message properties (optional): Contains additional properties to support compatibility with other providers or users. It can be used to create custom fields or filters (selectors). (3) A message body (optional): Allows users to create five types of messages (text message, map message, bytes message, stream message, and object message). The message

interface is extremely flexible and provides numerous ways to customize the contents of a message.

**JMS provider:** Represents a JMS interface to the MOM. It implements the JMS interface, which is a specification published by Sun. It is basically an adapter to the MOM.

**JMS session class:** Represents a single-threaded context for sending and receiving messages. A session is single-threaded so that messages are serialized, meaning that messages are received one-by-one in the order sent. The benefit of a session is that it supports transactions. If the user selects transaction support, the session context holds a group of messages until the transaction is committed, then delivers the messages. Before committing the transaction, the user can cancel the messages using a rollback operation. A session allows users to create message producers to send messages, and message consumers to receive messages.

**JMS/AMI:** Java Message Service / Application Messaging Interface

**JNDI:** Java Naming and Directory Interface. An API that provides naming and directory functionality. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**joint:** Connotes activities, operations, organizations, etc., in which elements of two or more military departments participate.

**joint composite tracking network (JCTN):** Generic title for a joint telecommunications network and processing capability to enable composite tracking among joint, heterogeneous mixes of sensors and to support appropriate levels of cooperative engagement of targets by weapons systems. It is envisioned as a real-time, sensor fusion system that distributes and fuses sensor measurement data into composite tracks that create a high-fidelity, coherent air picture. The JCTN is a concept rooted in the Navy's experience with Cooperative Engagement Capability (CEC). It includes common software and a communications element that allow participating units to share fused sensor data. The communications structure as currently envisioned includes wide-band line-of-sight communications, satellite links, and other communication systems.

**joint data network (JDN):** A collection of near-real-time communications and information systems used primarily at the coordination and execution level. It provides information exchange necessary to facilitate the Joint/Service Battle Manager's comprehension of the tactical situation, and also provides the means to exercise command and control beyond the range of organic sensors. The JDN carries near-real-time tracks, unit status information, engagement status and coordination data, and force orders. JDN information is used to cue radars as well. The backbone of the JDN is Link-16. However, other data links such as TADILA/B/C, Link-22, and VMF (Variable Message Format) will exchange information with the JDN through gateways at various platforms to ensure that disadvantaged users are included in the JDN. Satellites link geographically dispersed users in near real-time without consuming limited tactical bandwidth.

**joint force:** A general term applied to a force composed of significant elements, assigned or attached, of two or more military departments operating under a single joint force commander.

**joint planning network (JPN):** A collection of non-real-time and near real-time communication and information systems. JPN provides distributed collaborative planning capability, automated decision aids, and a means for distributing plans within theater. The core of the JPN is the GlobalCommand and Control System (GCCS) operating in the Defense Information Infrastructure Common Operating Environment (DII COE).

**joint task force:** A joint force that is constituted and so designated by the Secretary of Defense, a combatant commander, a sub-unified commander, or an existing joint task force commander.

**JPO:** Joint Program Office

**JScript:** Microsoft's extended implementation of ECMAScript (ECMA262), an international standard based on Netscape's JavaScript and Microsoft's JScript languages. JScript is implemented as a Windows Script engine. This means that you can plug it in to any application that supports Windows Script, such as Internet Explorer, Active Server Pages, and Windows Script Host. It also means that any application supporting Windows Script can use multiple languages: JScript, VBScript, Perl, and others.

**JSP:** Java Server Page. An extensible web technology that uses static data, JSP elements, and server-side Java objects to generate dynamic content for a client. Typically the static data is HTML or XML elements, and in many cases the client is a web browser. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**JSP page:** A text-based document containing static text and JSP elements that describes how to process a request to create a response. A JSP page is translated into and handles requests as a servlet. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**JSR:** Java Specification Request

**JSR 168:** JSR 168: Portlet Specification. To enable interoperability between portlets and portals, this specification defines a set of APIs for portal computing that address the areas of aggregation, personalization, presentation, and security. (Source: <http://www.jcp.org/en/jsr/detail?id=168>)

**JSSE:** Java Secure Socket Extension. A set of packages that enables secure Internet communications. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**JTA:** Java Transaction API. JTA specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications.

**JTAR:** Joint Tactical Air Request

**JTRS:** Joint Tactical Radio System

**Just-In-Time (JIT) compilation:** This is the primary method by which .NET executes MSIL. As the MSIL is executed, the code is compiled and optimized for the executing environment. JIT compilation provides environment optimization, runtime type safety, and assembly verification. To accomplish this, the JIT compiler examines the assembly metadata for any illegal accesses and handles violations appropriately.

**JVM:** Java Virtual Machine

**JWC:** A modular, loosely coupled, web-enabled, distributed, N-tier, service-oriented architecture written in Java, JavaScript, and HTML. It is platform-independent and designed to work in a heterogeneous environment.

## K

**keystore:** A file containing the keys and certificates used for authentication. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**kinematics:** Position, Velocity, and Acceleration.

**KIP:** Key Interface Profile

**KPP:** Key Performance Parameter

## L

**LAN:** Local Area Network. A group of interconnected computer and support devices.  
(Source:<http://www.sun.com/products-n-solutions/hardware/docs/html/817-6210-10/glossary.html>)

**layered software architecture:** Application software is separated into n tiers that separate concerns; minimally, client, presentation, middle, and data tiers

**LDAP:** Light Directory Access Protocol. A set of protocols for accessing information directories. LDAP is a simpler version of the X.500 standard. Unlike X.500, LDAP supports TCP/IP, which is necessary for Internet access. Because it's a simpler version of X.500, LDAP is sometimes called X.500-lite.

**least-common-denominator data access mechanism:** When one application is able to obtain data provided by another by removing arbitrary implementation barriers to data exchange.

**linked style sheet:** A style sheet in a separate text file that is saved in the root with a css file extension. The head section of the document contains a link to the file.

**logical architecture:** The logical architecture adds precision, providing a detailed "blueprint" from which component developers and component users can work in relative independence. It incorporates the detailed architecture diagram (with interfaces), component and interface specifications, and component collaboration diagrams, along with discussion of mechanisms, rationale, etc.

**look and feel:** Design aspects of a graphical user interface. It covers colors, shapes, layout, typefaces, and so on (the "look"); and, the behavior of dynamic elements such as buttons, boxes, and menus (the "feel"). It is used in reference to software and web sites. (Source: [http://en.wikipedia.org/wiki/Look\\_and\\_feel](http://en.wikipedia.org/wiki/Look_and_feel))

**look aspect:** One of the traditional aspects of a graphical user interface. The "look" covers such things as colors, shapes, layout, and typefaces.

**loosely coupled:** A computing model where application elements require a simple level of coordination and allow for flexible reconfiguration. Interconnection is often asynchronous and message-based.

## M

**MAIS:** Major Automated Information System

**manual track:** A track that is entered and updated by an operator. It may represent an object not seen by current sensors or provide a different representation of an entity than is currently being depicted by the sensors. In addition to system track correlation, the operator has the ability to associate or correlate this track with other tracks.

**MCP:** Mission Capability Package

**MDA:** Milestone Decision Authority

**MDD:** Model Driven Development. A general class of software development processes and techniques that emphasizes the use of models as a key element in the development. MDA™ is an example of one approach to MDD.

**MEA:** Multi-Element Array

**measurement:** A sensor-derived detection, contact, hit, or observation at a given point in time.

**measurement report:** A detection from a single sensor which has not yet been subjected to an association process.

**message bean:** An enterprise bean that provides asynchronous message support and clearly separates message and business processing.

**MHTML:** Mime HTML

**MIME:** Multi-purpose Internet Mail Extensions

**mission:** The task, together with the purpose, that clearly indicates the action to be taken and the reason for that action.

**mission-essential task (MET):** A task selected by a force commander from the Universal Navy Task List (UNTL) deemed essential to mission accomplishment.

**mission-essential task list (METL):** A list of tasks considered essential to the accomplishment of assigned or anticipated missions. A METL includes associated conditions and standards and may identify command-linked and supporting tasks.

**MLPP:** Multi-Level Priority and Preemption

**MMU:** Memory Management Unit

**model-driven architecture (MDA™):** Model-driven architecture™ is a trademarked term denoting a specific approach to the development of software using models as the basis. The MDA™ specifies system functionality separately from the implementation of that functionality on a specific technology platform. To accomplish this goal, the MDA™ defines an architecture that provides a set of guidelines for structuring specifications expressed as models. The MDA™ model architecture relates multiple standards, including Unified Modeling Language™ (UML™), the Meta Object Facility™ (MOF™), the XML Metadata interchange (XMI™), and the Common Warehouse Metamodel (CWM™). Note that the term “architecture” in MM does not refer to the architecture of the system being modeled, but rather to the architecture of the various standards and model forms that serve as the technology basis for MDA™.

**Model 4:** TADIL A Taxonomy (Link-11)

**Model 5:** TADIL J Taxonomy (Link-16)

**modular design:** Characterized by (1) Functional partitioning into discrete scalable, reusable modules consisting of isolated, self-contained functional elements; (2) Rigorous use of welldefined modular interfaces, including object-oriented descriptions of module functionality; (3) Ease of change to achieve technology transparency and, to the extent possible, make use of industry standards for key interfaces.

**module:** “(1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine. (2) A logically separable part of a program. Note: The terms ‘module,’ ‘component,’ and ‘unit’ are often used interchangeably or defined to be sub-elements of one another in different ways depending

upon the context. The relationship of these terms is not yet standardized.” See also component. (Source: IEEE Std 610.12-1990)

**MOJE:** Map Objects Java Edition

**MOM:** Message-Oriented Middleware

**MOP:** Maintenance Operation Protocol

**MOSA:** Modular Open Systems Approach

**multi-sensor correlated track:** A representation of an entity that is formed by correlating track reports using various methods based upon time latency of the given tracks. These multiple tracks are correlated to form one representation of the track.

**multi-user access:** A system where multiple users can simultaneously access data stores, use applications, and analyze and direct operations.

**mutual authentication:** An authentication mechanism employed by two parties for the purpose of proving each other's identity to one another. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

## N

**namespace:** A standard that lets you specify a unique label for the set of element names defined by a DTD or XSD. A document using that DTD or XSD can be included in any other document without causing a conflict between element names. The elements defined in a particular DTD are uniquely identified so that, for example, the parser can tell when an element <name> should be interpreted according to the particular DTD rather than using the definition for an element <name> in a different DTD. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**NAS:** Network Attached Storage

**native XML database:** Defines a logical model for an XML document (as opposed to the data in that document) and stores and retrieves documents according to that model. These databases are accessed via programming interfaces such as SAX, DOM, or JDOM. There is a trend away from pure XML storage because all the leading relational database vendors are introducing advanced XML capabilities.

**Navy Tactical Task List (NTTL):** The comprehensive list of Navy and Coast Guard (DoD-related missions) tasks at the tactical level of war.

**NCES:** Net-Centric Enterprise Services. The NCES program provides enterprise-level Information Technology (IT) services and infrastructure components, also called Core Enterprise Services, for the Department of Defense (DoD) Global Information Grid (GIG).

**NCOIC:** Network Centric Operations Industry Consortium

**NCOW:** Net-Centric Operations Warfare

**NCOW RM:** Net-Centric Operations Warfare Reference Model. An information-enabled concept of operations that generates increased combat power by networking sensors, decision makers, and shooters. This enables shared awareness, increased speed of command, higher tempo of operations, greater lethality, increased survivability, and a degree of self-synchronization. In essence, network-centric warfare translates information superiority into combat power by effectively linking knowledgeable entities in the battlespace.

**NCW:** Net-Centric Warfare

**near-real-time (Tracks):** (1) Near-real-time tracks are generated by real-time sensors on remote units, whose delivery latencies are sufficiently large that while they can be used to help decide to engage on the target, they cannot be used to fire on the target. The data is primarily used for situational awareness. (2) The timelines of the data or information have been delayed by the time required for electronic communications and automatic data processing. (Source: 7P1 SS)

**NESI:** Net-Centric Enterprise Solutions for Interoperability. A joint effort between the U.S. Navy's Program Executive Office for C4I & Space and the U.S. Air Force's Electronic Systems Center. It provides a reference architecture, implementation guidance, and a set of reusable software components. These facilitate the design, development, maintenance, evolution, and use of information systems for the Net-Centric Operations and Warfare (NCOW) environment.

**net:** A globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for data and information exchange.

**net-centric:** A net-centric environment is one in which users and local applications depend upon common services for functionality and data. Users can access applications and data through web services. This provides an information environment that comprises interoperable computing and communication components. A net-centric environment exploits advancing technology to move from an application-centric to a data-centric paradigm.

**net-centric information environment:** A net-centric information environment uses emerging standards and technologies to optimize assured information sharing among users. It results from implementing GIG component architectures in accordance with the NCOW RM. A net-centric information environment includes core and COI enterprise services, and a data-sharing strategy that emphasizes metadata concepts, shared information spaces, and the TPPU paradigm.

**net-centricity:** The realization of a robust, globally interconnected, network environment (including infrastructure, systems, processes, people) in which data is shared seamlessly in a timely manner among users, applications, and platforms. By securely interconnecting people and systems, independent of time or location, net-centricity substantially improves military situational awareness and significantly shortens decision-making cycles. Users can better protect assets; exploit information more effectively; use resources more efficiently; and unify our forces by supporting extended, collaborative communities to focus on the mission.

**NETOPS:** Network Operations. An organizational, procedural, and technological construct for ensuring information and decision superiority at the strategic, operational, and tactical levels of warfare as well as within DoD business operations. NetOps is an operational approach, which addresses the interdependency and integration of IA/CND, S&NM, and CS capabilities. NetOps consists of the organizations, tactics, techniques, procedures, functionalities, and technologies required to plan, administer, and monitor use of the GIG infrastructure and the end-to-end information flows of the GIG; and to respond to threats, outages, and other operational impact. NetOps ensures mission requirements are properly considered in GIG operational decision-making. NetOps enables the GIG to provide its users with information they need, when and where they need it, with appropriate protection. NetOps is essential for successful execution of net-centric warfare and other net-centric operations in support of national security objectives.

**network:** A system of computers, terminals, databases, cables, satellites, and other elements that enable digital communications.

**NGEN compilation:** Native Image Generator compilation. NGEN enables you to produce a native binary image of MSIL code for the current environment. This improves the performance of the .NET application by eliminating the JIT overhead associated with the execution. Once you run NGEN against an assembly, the resulting native image is placed in the Global Assembly Cache for use by all other .NET assemblies.

**niche database:** Created in response to shortcomings in relational databases. Market domination by large vendors has made it hard for small vendors to break into the market, so niche database vendors mainly provide supporting tools.

**NIS:** Node Information Services

**NMCI:** Navy Marine Corps Intranet

**node:** A set of information systems acquired and managed as a single element in the net-centric enterprise. In NESI, these entities are designed to support distributed services for a collection of systems, applications, data, and components that share a common set of mission functions on a common infrastructure.

**node manager:** The organization responsible for integrated planning, acquisition, and delivery of integrated, tested, certified C2 Node systems, sub-systems, components, and services.

**node platform infrastructure:** A set of information systems and technologies, based on a commercial product stack, that provides an integrated common software component execution framework and infrastructure.

**non-functional requirements:** Address issues such as reliability, performance, supportability, constraints, and physical matters. Many requirements are non-functional, and describe only attributes of the system or attributes of the system environment. Although some of these may be captured in use cases, those that cannot may be specified in supplementary specifications.

**non-real time (Tracks):** (1) Non-real-time tracks have latencies that nominally range from 15 seconds to days. (2) The timelines of the data or information have been delayed such that the data or information has questionable utility beyond situational awareness. (Source: 7P1SS)

**normalization:** Normalization avoids duplication of data, insert anomalies, delete anomalies, and update anomalies. A relation is in first normal form (1NF) if and only if all underlying simple domains contain atomic values only. A relation is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key. A relation is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. Data models should follow the three forms unless there is overriding justification not to. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**NPI:** Node Platform Infrastructure

**NR KPP:** Net-Ready Key Performance Parameter. Measures the net-centricity of a new program or major upgrade.

## O

**OASIS:** Organization for the Advancement of Structured Information Standards. A nonprofit, international consortium that promotes the adoption of product-independent standards for information formats such as SGML, XML, and HTML. Its web site is <http://www.oasis-open.org/>. The DTD repository it sponsors is at <http://www.XML.org>. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**object-based design:** Any design that incorporates objects. Contrast with object-oriented design and class-based design.

**object-based programming language:** A programming language that provides the ability for the programmer to define and use objects; for example, Ada 83.

**object-oriented databases (OODBMS):** Object-oriented databases are based on the object model, and use the same conceptual models as object-oriented analysis and design. Since OODBMSs have portability issues, only a limited number of people and outside resources support them.

**object-oriented design:** Any design that incorporates objects, classes, and inheritance. Contrast with object-based design and class-based design.

**object-oriented programming language:** A programming language that enables programmers to define and use objects, classes, and inheritance; for example, C++, Ada 95.

**object type:** During the Object-Oriented (OO) boom there was a push for all programming efforts to completely support the OO paradigm. Many of the DBMS vendors responded by providing support for User-Defined Types (UDT) and Objects.

**OBV:** Objects by Value

**ODBC:** Open Database Connectivity

**OGC:** Open Geospatial Consortium, Inc. Data posted by authoritative sources and visible, available, and usable to accelerate decision making. (Source: <http://www.opengeospatial.org/>)

**OHIO:** Only Handle Information Once

**OLA:** Operational Level Agreements

**OMG:** Object Management Group. A open-membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. Its web site is <http://www.omg.org/>. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**OO:** Object Oriented

**open-systems approach:** An integrated business and technical strategy that employs a modular design and, where appropriate, defines key interfaces using widely supported, consensus-based standards that are published and maintained by a recognized industry-standards organization.

**open architecture:** An architecture that supports public access to some of its parts. An open architecture is the design of an open system. It also refers to a set of design patterns and principles by which open systems are developed. An architecture that employs open standards for key interfaces within a system.

**open source:** Generically, “open source” refers to a program in which the source code is available to the general public for use and/or modification from its original design free of charge. Open-source code is typically created as a collaborative effort in which

programmers improve upon the code and share the changes within the community. Open source sprouted in the technological community as a response to proprietary software owned by corporations.

**open standards:** Standards that are widely used, consensus-based, published, and maintained by recognized industry-standards organizations.

**open system:** “An open system is a collection of interacting software, hardware, and human components designed to satisfy stated needs, with interface specifications of its components that are fully defined, available to the public, and maintained according to group consensus. In the collection, the implementations of the components conform to the interface specifications.” (SEI)

**ORB:** Object Request Broker. A library that enables CORBA objects to locate and communicate with one another. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**OS file system:** Stores and retrieves data, acting as a data tier. Advocates cite performance and simplicity, but the loss of DBMS-inherent capabilities such as ad-hoc queries and the ability to upgrade to faster machines is a deterrent. File-system-based data tiers often result in proprietary solutions that are hard to maintain and port.

**OSI:** Open Systems Interconnect

**OSJTF:** Open Systems Joint Task Force

**OSS:** Open Source Software. (References: Scott Hissam, Charles B. Weinstock, Daniel Plakosh, Jayathirtha Asundi Perspectives on Open Source Software. November 2001. Technical Report CMU/SEI-2001-TR-019.) “The term open source software at the most basic level simply means software for which the source code is open and available. Open and available is meant to convey two concepts: Open—The source code for the software can be read (seen) and written (modified). Further, this term is meant to promote the creation and distribution of derivative works of the software. Available—The source code can be acquired either free of charge or for a nominal fee (e.g., media and shipping charges or online connection charges).”

**OTN:** Oracle Technology Network

**OUSD:** Office of the Under Secretary of Defense

**OWS:** OGC Web Services

## P

**Pacq:** Probability of Acquisition

**parser:** A module that reads in XML data from an input source and breaks it into chunks so that your program knows when it is working with a tag, an attribute, or element data. A nonvalidating parser ensures that the XML data is well formed but does not verify that it is valid. See also validating parser. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**PART:** Program Assessment Rating Tool

**PCP:** Program Change Proposals

**PDA:** Personal Digital Assistant

**PEO:** Program Executive Office

**performance requirement:** “A requirement that imposes conditions on a functional requirement; for example, a requirement that specifies the speed, accuracy, or memory usage with which a given function must be performed.” (Source: IEEE Std 610.12-1990)

**personalization:** The ability for portal members to subscribe to specific types of content and services. Users can customize the look and feel of their environment.

**physical model:** Translates the conceptual model to a particular RDBMS implementation.

**PKI:** Public Key Infrastructure

**POA:** Portable Object Adapter. A CORBA standard for building server-side applications that are portable across heterogeneous ORBs. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**point-to-point messaging system:** A messaging system built on the concept of message queues. Each message is addressed to a specific queue. Clients extract messages from the queues established to hold their messages. These messages are normally persistent and a client can retrieve messages at any time, similar to email. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**pop-up window:** A window that suddenly appears (pops up) when you select an option with a mouse or press a special function key. Usually, the pop-up window contains a menu of commands and stays on the screen only until you select one of the commands. Also, a type of window that appears over the browser window of a web site when visited by a user. Pop-up windows are used extensively in advertising on the web, though advertising is not the only application for pop-up windows. Turn these off when using UNCG / DCL online courses. A special kind of pop-up window is a pull-down menu, which appears just below the item you selected, as if you had pulled it down. (Source: <http://web.uncg.edu/dcl/icampus/access/glossary.asp>)

**POR:** Program of Record

**portability:** “The ease with which a system or component can be transferred from hardware or software environment to another.” (Source: IEEE Std 610.12-1990) The level of software portability of any specific product depends on two factors: the design of the product itself, and the characteristics of the source and target execution environments. Software products are rarely if ever 100% portable. Generally, the level of portability depends on the target platform. Software that is highly portable to one class of platform might be not portable to other classes.

**portal:** A web portal is a web site that provides a starting point, gateway, or portal to other resources on the Internet or an intranet. Intranet portals are also known as "enterprise information portals" (EIP). Examples of existing portals are Yahoo, Excite, Lycos, Altavista, infoseek, and Hotbot. (Source: [http://en.wikipedia.org/wiki/web\\_portal](http://en.wikipedia.org/wiki/web_portal))

**portal page:** A complete document rendered by a portal. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)

**portlet:** A reusable web component that displays relevant information to portal users. Examples for portlets include email, weather, discussion forums, and news. The purpose of the Web Services for Remote Portlets interface is to provide a web services standard that allows for the "plug-n-play" of portals, other intermediary web applications that aggregate content, and applications from disparate sources. The portlet specification enables interoperability between portlets and portals. This specification defines a set of APIs for portal computing that addresses the areas of aggregation, personalization, presentation, and security. (Source: <http://en.wikipedia.org/wiki/Portlets>)

**portlet container:** A portlet container provides a runtime environment for portlets implemented according to the portlet API. In this environment portlets can be instantiated, used, and finally destroyed. The portlet container is not a standalone container like the servlet container; instead it is implemented as a thin layer on top of the servlet container and reuses the functionality provided by the servlet container. (Source: [http://66.102.7.104/search?q=cache:2wl3P7hXCWAJ:portals.apache.org/pluto/+what+is+a+portlet+container"&hl=en](http://66.102.7.104/search?q=cache:2wl3P7hXCWAJ:portals.apache.org/pluto/+what+is+a+portlet+container))

**POSIX®:** Portable Operating System Interface for Computing Environments

**primary key:** An object that uniquely identifies a row within a table.

**procedural language support:** Procedural languages are optimized to efficiently manipulate large quantities of data. Unfortunately, they are not very portable. Java is a useful, portable alternate to these proprietary languages.

**producer:** A web service conforming to the WSRP specification. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)

**proprietary software:** Proprietary software is software for which an individual or company holds the exclusive copyright, and for which the license rights deny others access to the software's source code and the right to copy, modify, and study the software. (Source: [http://en.wikipedia.org/wiki/Proprietary\\_software](http://en.wikipedia.org/wiki/Proprietary_software))

**proprietary standard:** A standard that is exclusively owned by an individual or organization, the use of which generally would require a license and/or fee.

**proxy pattern:** Provides a surrogate or placeholder for another object to control access to it.

**public domain:** The term “public domain” describes publications, software, and other resources which are not protected by copyrights or patents.

**public key certificate:** Used in client-certificate authentication to enable the server, and optionally the client, to authenticate each other. The public key certificate is the digital equivalent of a passport. It is issued by a trusted organization, called a certificate authority, and provides identification for the bearer. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**publish/subscribe messaging system:** A messaging system in which clients address messages to a specific node in a content hierarchy, called a topic. Publishers and subscribers are generally anonymous and can dynamically publish or subscribe to the content hierarchy. The system takes care of distributing the messages arriving from a node's multiple publishers to its multiple subscribers. Messages are generally not persistent and will only be received by subscribers who are listening at the time the message is sent. A special case known as a “durable subscription” allows subscribers to receive messages sent while the subscribers are not active. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**PWS:** Personal Web Server. A web server program for PC users who want to share web pages and other files from their hard drive. PWS is a scaled-down version of Microsoft's more robust web server, Internet Information Server IIS. PWS can be used with a full-time Internet connection to serve web pages for a web site with limited traffic. It can also be used for testing a web site offline or from a "staging" site before putting it on a main web site that is exposed to more traffic.

## Q

**QoS:** Quality of Service. Data timeliness, accuracy, completeness, integrity, and ease of use. Refers to the probability of the network meeting a given traffic contract. In many cases is used informally to refer to the probability of a packet passing between two points in the network. (Source: [http://en.wikipedia.org/wiki/Quality\\_of\\_service](http://en.wikipedia.org/wiki/Quality_of_service)) -OR- A defined level of performance that adapts to the environment in which it is operating. QoS may be requested by the user of the information. The level of QoS provided is based on the request, the available capabilities of the provider, and the priority of the user.

## R

**RAPIDS:** Reusable Applications Integration and Development Standards. Established with the objective of developing a common set of software standards and implementing a set of processes designed to build portable and reusable software. The intent was to reduce both the time and cost of developing software for Navy C4I systems. This NCW effort was merged with the Air Force's C2ERA to form NESI.

**RAR:** Resource Adaptor Archive. A J2EE component that implements the J2EE Connector architecture for a specific Enterprise Information System (EIS). J2EE applications communicate with an EIS through the resource adapter. You can deploy RARs on any J2EE server. A RAR file may be independent or contained in an EAR file.

**RBAC:** Role-Based Access Control. An approach to restricting system access to authorized users. It is a newer and alternative approach to discretionary access control and mandatory access control. It assigns permissions to specific operations with meaning in the organization, rather than to low-level data objects. (Source: <http://en.wikipedia.org/wiki/RBAC>)

**RDBMS:** Relational Database Management System. A database management system (DBMS) that is based on the relational model or that presents the data to the user as relations. A collection of tables, each table consisting of a set of rows and columns, can satisfy this property. RDBMSs also provide relational operators to manipulate the data in tabular form. (Source: <http://en.wikipedia.org/wiki/RDBMS>)

**real-time:** An operation within a larger dynamic system is called a real-time operation if the combined reaction- and operation-time of a task is shorter than the maximum delay that is allowed, in view of circumstances outside the operation. The task must also occur before the system to be controlled becomes unstable. A real-time operation is not necessarily fast, as slow systems can allow slow real-time operations. This applies for all types of dynamically changing systems. The polar opposite of a real-time operation is a batch job with interactive timesharing falling somewhere in-between the two extremes. (Source: [http://en.wikipedia.org/wiki/Real\\_time](http://en.wikipedia.org/wiki/Real_time))

**real-time (tracks):** (1) Real-time tracks are generated by sensors whose delivery latencies are sufficiently small to use in anti-air warfare (AAW). They form composite tracks for situational awareness and are of sufficient quality to engage and fire on the target. "Quality" is a weapon-dependent term. The key issue is the latency of the arrival and subsequent usage of the track data. Periodicity is also a component of track quality. (2) Pertaining to a system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process.

**real-time system:** A system in which the correctness of system behavior depends on both the logical correctness of the computation and the time at which the result is produced. For a real-time system, the system fails if its timing constraints are not met. "Real time" is not

- necessarily synonymous with “fast.” The latency of the response might not be an issue, and it could be on the order of seconds or minutes. But the bounded latency that is sufficient to solve the problem at hand is guaranteed by the system. “Bounded” means that the response is neither too early nor too late. In real-time systems, early can be as bad as late.
- reference model:** A structure that allows the modules and interfaces of a system to be described in a consistent manner.
- referential integrity:** A feature provided by RDBMSs that prevents users or applications from entering inconsistent data. Most RDBMSs have various referential integrity rules that you can apply when you create a relationship between two tables.
- relational databases (RDBMS):** A collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.
- replication:** Replication is the process of copying data from one DBMS to another DBMS. As data are added to or modified in a database, replication adds or modifies the data in another, physically separated, database.
- request-response messaging system:** A system of messaging that includes blocking until a response is received. (Source:<http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- request for information (RFI):** Any specific time-sensitive ad-hoc requirement for intelligence information or products. RFIs support ongoing crises or operations not necessarily related to standing requirements or scheduled intelligence production. A RFI can be initiated to respond to operation requirements and will be validated in accordance with the theater command’s procedures.
- requirement:** A condition or capability to which a system must conform. Requirements may be derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document. A desired feature, property, or behavior of a system. A capability that the system must deliver.
- resource adapter:** A system-level software driver that a Java application uses to connect to an Enterprise Information System (EIS).
- RMI:** Remote Method Invocation. A technology that allows an object running in one Java virtual machine to invoke methods on an object running in a different Java virtual machine. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- RMI/IIOP:** Remote Method Invocation / Internet Inter-Orb Protocol. A version of RMI implemented to use the CORBA IIOP protocol. RMI over IIOP provides interoperability with CORBA objects implemented in any language if all the remote interfaces are originally defined as RMI interfaces. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- ROE:** Rules of Engagement
- role mapping:** The process of associating groups, principals, or both, recognized by the container with security roles specified in the deployment descriptor. Security roles must be mapped by the deployer before a component is installed in the server. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- rollback:** The point in a transaction when all updates to any resources involved in the transaction are reversed. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**RPC:** Remove Procedure Call. An alternative to sockets that abstracts the communication interface to the level of a procedure call. The programmer has the illusion of calling a local procedure, but in fact the arguments of the call are packaged and sent to the remove target of the cell. RPC systems encode arguments and return values using an external data representation such as XDR. RPC does not translate well into distributed object systems, which require communication between program-level objects in different address spaces. To match the semantics of object invocation, distributed object systems require RMI. A local surrogate (stub) object manages the invocation on a remote object.

**RTOS:** Real-time Operation System. An operating system that has been developed for real-time applications. Typically used for embedded applications. This type of operating system does not necessarily have high throughput — the specialized scheduling algorithm and a high clock-interrupt rate can both interfere with throughput. (Source: <http://en.wikipedia.org/wiki/RTOS>)

## S

**SAML:** Security Assertion Markup Language. An XML standard for exchanging authentication and authorization data between security domains; that is, between an identity provider and a service provider. SAML is a product of the OASIS Security Services Technical Committee. (Source: <http://en.wikipedia.org/wiki/SAML>)

**SAN:** Storage Area Network. A network designed to attach computer storage devices such as disk array controllers and tape libraries to servers. (Source: <http://en.wikipedia.org/wiki/SAN>)

**SAP:** Service Access Point. SAP provides all of the information necessary for a user to access and consume a service. Includes the logical and physical location of the service on the net.

**SCA:** System Communications Architecture

**SCDR:** Shared Cross-Domain Resource

**SDF:** Service Definition Framework. SDF provides service users, customers, developers, providers, and managers with a common frame of reference. Its structure and methodology enable you to fully define the Service Access Points (SAPs) for the service.

**SDK:** Software Developer's Kits. A set of development tools that allows a software engineer to create applications for a certain software package, software framework, hardware platform, computer system, operating system, and so on. It may be as simple as an application programming interface in the form of some files to interface to a particular programming language, or as complex as sophisticated hardware to communicate with a certain embedded system. Common tools include debugging aids and other utilities. SDKs frequently include sample code, technical notes, and other supporting documentation to clarify points from the primary reference material. (Source: <http://en.wikipedia.org/wiki/SDK>)

**Sea Basing:** Projecting Joint Operational Independence through the extended reach of networked weapons and sensors. Capabilities include: (1) enhanced afloat positioning of joint assets; (2) offensive and defensive power projection; (3) command and control; (4) integrated joint logistics; and (5) accelerated deployment and employment timelines.

- Sea Shield:** Takes naval defense beyond unit- and task-force defense to provide the nation with sea-based theater and strategic defense. Capabilities include: (1) homeland defense; (2) sea and littoral superiority; (3) theater air missile defense; and (4) force entry enabling.
- Sea Strike:** Describes the capabilities of naval forces to project decisive and persistent offensive power anywhere in the world. Capabilities include: (1) Persistent intelligence, surveillance, and reconnaissance; (2) time-sensitive strikes; (3) electronic warfare/ and information operations; (4) ship-to-objective maneuvers; and (5) covert strikes.
- security role:** An abstract logical grouping of users that is defined by the application assembler. When an application is deployed, the roles are mapped to security identities, such as principals or groups, in the operational environment. In the J2EE server authentication service, a role is an abstract name for permission to access a particular set of resources. A role can be compared to a key that can open a lock. Many people might have a copy of the key; the lock doesn't care who you are, only that you have the right key. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- separation of implementation and interface:** Services expose mission capabilities through well-defined interfaces and provide reliable and scalable components
- service:** A service is any function that has a clearly defined interface accessed through well-defined public access points.
- service availability:** The name and location of the organization responsible for the day-to-day operational management of the service. Include operational point of contact information, trouble-reporting procedures, and applicable POCs, telephone numbers, email addresses, etc.
- service consumer:** The person, organization, or automated asset that makes use of a service.
- service description:** A short descriptive name of the service. Include a human-readable description and the XML Qualified Name (QName) for the service.
- service performance specification:** The percentage of time that the service shall be available over a specified period of time (typically one year). Agreed-upon maintenance or other scheduled downtime does not count against total availability.
- service provider:** The person, organization, or automated asset that implements and operates a service.
- service registry:** Provides descriptive information about a service, enabling the lookup and discovery of services.
- service response time:** The planned performance levels of the service (e.g., throughput, capacity, or other applicable measure) expressed as a function of work units processed per unit of time.
- servlet:** A Java program that extends the functionality of a web server, generating dynamic content and interacting with web applications using a request-response paradigm. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)
- servlet container:** A container that provides the network services over which requests and responses are sent, decodes requests, and formats responses. All servlet containers must support HTTP as a protocol for requests and responses but can also support additional request-response protocols, such as HTTPS. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**servlet context:** An object that contains a servlet's view of the web application within which the servlet is running. Using the context, a servlet can log events, obtain URL references to resources, and set and store attributes that other servlets in the context can use. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**servlet session:** An object used by a servlet to track a user's interaction with a web application across multiple HTTP requests. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**session:** An interaction between system entities of finite duration, often involving a user, typified by the maintenance of some state of the interaction for the duration of the interaction. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)

**session bean:** An enterprise bean that is created by a client and that usually exists only for the duration of a single client-server session. A session bean performs operations, such as calculations or database access, for the client. Although a session bean can be transactional, it is not recoverable should a system crash occur. Session bean objects can be stateless or can maintain conversational state across methods and transactions. If a session bean maintains state, then the EJB container manages this state if the object must be removed from memory. However, the session bean object itself must manage its own persistent data. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**SGML:** Standard Generalized Markup Language. The parent of both HTML and XML. Although HTML shares SGML's propensity for embedding presentation information in the markup, XML is a standard that allows information content to be totally separated from the mechanisms for rendering that content. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**shareware:** Shareware is software whose license terms permit free redistribution, but also require that anyone who continues to use a copy must pay a license fee. (Source: <http://www.gnu.org/philosophy/categories.html>)

**SIAP:** Single Integrated Air Picture. The SIAP is the product of fused, common, continuous, unambiguous tracks of all airborne objects in the surveillance area. Each object within the SIAP has one, and only one, track number and set of associated characteristics. The SIAP is developed from near-real time and real time data, and is scalable and filterable to support situation awareness, battle management, and target engagements. JTAMDO Battle Management Concept.

**SIMPLE:** Session Initiation Protocol for Instant Messaging

**single touch point:** The portal becomes the delivery mechanism for all business information services.

**SIP:** Simple Initiation Protocol. The SIP standard concerns simple call placement and is designed to be easily expandable.

**SLA:** Service Level Agreements. A contractual vehicle between a service provider and a service consumer. It specifies performance requirements, measures of effectiveness, reporting, cost, and recourse. It usually defines repair turnaround times for users.

**smart pull:** Applications that encourage discovery; users can pull data directly from the net or use value-added discovery services.

**SMTP:** Simple Mail Transfer Protocol

**SNMP:** Simple Network Management Protocol. A design style for building flexible, adaptable, distributed-computing environments. SOA is the unifying structure in which the

components of a computer, computer system, or system of systems are integrated. All of its inter-component functions are defined as services. Service-oriented design is fundamentally about sharing and reusing functionality across diverse applications.

**SOA:** Service-Oriented Architecture. Services enable access to data and application functionality through public interfaces exposed to the enterprise

**SOAP:** Simple Object Access Protocol. SOAP is a lightweight XML-based messaging protocol used to encode the information in web-service request-and-response messages before sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols, including SMTP, MIME, and HTTP.

**soft real-time:** In a soft real-time system, the value of an operation declines steadily after the deadline expires.

**software license:** A software license sets out the terms under which the software may be used, and serves as an agreement between the producer and the users of the program. A set of terms and conditions which the owner of the copyright on a piece of software conveys to users of the software. Licenses take many different forms.

**software patent:** Patents grant an inventor the right to exclude others from producing or using the inventor's discovery or invention for a limited period of time. In order to be patented an invention must be novel, useful, and not of an obvious nature (see §§ 101 - 103 of Title 35). The Federal agency charged with administering patent laws is the Patent and Trademark Office. See §§ 1-26 of Title 35. Its regulations, pertaining to Patents, are found in Parts 2 - 6 of Title 37 of the Code of Federal Regulations. Each patent application for an alleged new invention is reviewed by an examiner to determine if it is entitled to a patent. See § 1.104 of Part 1 of Title 37 (C.F.R.). While historically a model was required as part of a patent application, in most cases today, only a detailed specification is necessary. See §§ 112 - 114 of Title 35. Software may be patented. There are currently more than 4,000 software patents in effect.

**software unit:** (1) A separately testable element specified in the design or a computer software component. (2) A logically separable part of a computer program. (3) A software component that is not subdivided into other components. (4) (IEEE Std 1008-1987 [10]) Note: The terms "module," "component," and "unit" are often used interchangeably or as subelements of one another in different ways depending upon the context. The relationship of these terms is not yet standardized. In common usage, the term generally denotes the smallest compilable software component, in the context of (3) from the IEEE definition. That is, it can be compiled, and it does not contain any other software components. See also test unit. (Source: IEEE Std 610.12-1990)

**SOO:** Statement of Objectives

**SOW:** Statement of Work

**SPAWAR:** Space and Naval Warfare Systems Command

**SQL:** Structured Query Language. The standardized relational database language for defining database objects and manipulating data. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**SQL-92:** Structured Query Language 1992. The SQL-92 and SQL:1999 standards are very detailed and specific. At the current time, no RDBMS vendors fully support the entire standard. Vendors that claim they are SQL-92-compliant or SQL:1999-compliant are actually only compliant to a certain level. The SQL-92 standard defines the following

levels, which also apply to SQL:1999: (1) Notational; (2) Transitional level SQL92; (3) Intermediate level SQL92; (4) .Full SQL92. (Source: <http://dbs.uni-leipzig.de/en/lokal/standards.pdf>; [http://developer.mimer.com/documentation/html\\_82/Mimer\\_SQL\\_Reference\\_Manual/Intro\\_SQL\\_Stds3.html](http://developer.mimer.com/documentation/html_82/Mimer_SQL_Reference_Manual/Intro_SQL_Stds3.html))

**SQL/J:** Structured Query Language for Java. A set of standards that includes (1) specifications for embedding SQL statements in methods in the Java programming language and (2) specifications for calling Java static methods as SQL stored procedures and user-defined functions. An SQL checker can detect errors in static SQL statements at program development time, rather than at execution time as with a JDBC driver. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**SQL:1999:** Structured Query Language 1999. See SQL-92.

**SSL:** Secure Socket Layer. A technology that allows web browsers and web servers to communicate over a secured connection. The protocol runs above TCP/IP and below application protocols. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**SSO:** Single Sign-On

**stakeholder:** An enterprise, organization, or individual having an interest or a stake in the outcome of the engineering of a system. (Source: EIA-632, Annex A)

**standard:** A document that establishes engineering and technical requirements for products, processes, procedures, practices, and methods that have been decreed by authority or adopted by consensus. (Source: EIA-632, Annex A)

**stateful session bean:** A session bean with no conversational state. All instances of a stateless session bean are identical. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**stateless session bean:** A session bean with no conversational state. All instances of a stateless session bean are identical. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**stored procedure:** A unit or module of code that executes in a database and implement some bit of application logic or business rule. Often written in proprietary language such as Oracle's PL/SQL or Sybase's Transact-SQL.

**STR:** Software Trouble Report

**style sheet:** A specification of formatting instructions that, when applied to structured information, provides a particular rendering of that information (for example, online or printed). Different style sheets may be applied to the same piece of structured information to produce different presentations of the information. (Source: IBM WebSphere Glossary)

**subsystem:** A group of items that performs a set of functions within a particular end product. (Source: EIA-632, Annex A)

**supporting source track:** A composite/collaborative track, a multi-sensor correlated track, a manual track, or an INTEL-generated track that is the basis for declaring the existence of a system track.

**supporting task:** Specific activities that contribute to the accomplishment of a joint-mission-essential task. Supporting tasks are accomplished at the same command level or by subordinate elements of a joint force (e.g., joint staff, functional components, etc.).

**Swing (JFC):** Java Foundation Classes. The Java Foundation Classes are a set of Java class libraries provided as part of the Java 2 Platform, Standard Edition (J2SE) to support

- building graphical user interfaces (GUI) and graphics functionality for client applications that will run on popular platforms such as Microsoft Windows, Linux, and Mac OSX.
- system:** Two or more interrelated pieces of equipment (or sets) arranged in a package to perform an operational function or to satisfy a requirement. (Source: Defense Acquisition Glossary of Terms, Jan 2001)
- system architecture:** The composite of the design architectures for products and their life cycle processes. (Source: IEEE 1220-1998)
- system time:** Represents the time standard used within the combat system, including the local source of Universal Coordinated Time (UTC), a system-wide monotonically increasing reference time, as well as other representations of the system-wide reference time.
- system track:** A platform-specific representation of an individual entity, identified by a unique system track number, containing one or more track state vectors and uncertainties, as well as associated attributes, attribute uncertainties, and data valid time.

## T

- tactical data, other:** Data of a non-kinematic, non-sensor-processed nature including intelligence, imagery, voice, context information (e.g., commercial air and shipping lanes, political boundaries).
- task:** A discrete event or action, not specific to a single unit, weapon system, or individual, that enables a mission or function to be accomplished.
- TCP:** Transmission Control Protocol. One of the core protocols of the Internet protocol suite. Using TCP, programs on networked computers can create connections to one another, over which they can send data. The protocol guarantees that data sent by one endpoint will be received in the same order by the other, without any pieces missing. It also distinguishes data for different applications (such as a web server and an email server) on the same computer. (Source: [http://en.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](http://en.wikipedia.org/wiki/Transmission_Control_Protocol))
- TCP/IP:** Transmission Control Protocol/Internet Protocol. A suite of communications protocols used to connect hosts on the Internet. TCP/IP uses several protocols, the two main ones being TCP and IP. TCP/IP is built into the UNIX operating system and is used by the Internet, making it the de facto standard for transmitting data over networks. Even network operating systems that have their own protocols, such as Netware, also support TCP/IP.
- TDL:** Technical Direction Letter
- TDS:** Technology Development Strategy. Rationale and description of how the program will be divided into technology spirals and development increments, specific performance goals, and exit criteria for moving beyond prototype limitations. Program strategy for the total R&D program. Specific cost, schedule, performance goals, and test plan for first technology spiral development.
- telnet:** The Telnet protocol enables terminals and terminal-oriented processes to communicate on a network running TCP/IP. (Source: <http://www.sun.com/products-n-solutions/hardware/docs/html/817-6210-10/glossary.html>)

**TEMP:** Test and Evaluation Master Plan. Describes all planned testing, including measures to evaluate the performance of the system during test periods, an integrated test schedule, and resource requirements.

**tenet:** Net-centric design precept.

**time-out:** A period of time after which some condition becomes true if some event has not occurred. -OR- The action of so doing. For example, a session that is terminated because its state has been inactive for a specified period of time is said to “time out”. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)

**TLS:** Transport Level Security

**TPED:** Task, Process, Exploit, Disseminate

**TPPU:** Task, Post, Process, Use

**TQ:** Track Quality. A numerical value assigned to a track that represents the accuracy of the track position. It is computed from data related to the past tracking performance.

**track:** (1) A set of detections, contacts, hits, or observations, generated by the same real object in the environment. It is identified by a track number, and has intrinsic and derived attributes associated with it. (2) A series of related contacts displayed on a data display console or other display device. (3) To display or record the successive positions of a moving object.

**track kinematics:** A track state vector that represents the best understanding of the entity’s position and movement at a defined point in time with the objective of predicting the entity’s future position if it maintains a consistent direction of movement.

**track number:** The unique or alphanumeric identifier associated with a specific set of track data, representing a vehicular object, point, line of bearing, fix, or area of probability.

**track quality (TQ):** A numerical value assigned to a track that represents the accuracy of the track position, computed from data related to past tracking performance.

**track state:** Smoothed position and velocity representation of an individual object, which minimizes the RMS errors in estimates of the closest point of approach and time of closest point of approach.

**track, local:** A track established within a unit based on sensor measurements derived from the local platform sensors.

**track, remote:** A track established by a remote unit, or group of units, and supplied to the local platform.

**trade secret:** A trade secret is any formula, pattern, device, or compilation of information used in a business that gives an advantage over competitors who do not know it or use it.

**trademark:** Trademarks are generally distinctive symbols, pictures, or words that sellers use to distinguish and identify the origin of their products. Trademark status may also be granted to distinctive and unique packaging, color combinations, building designs, product styles, and overall presentations. It is possible to receive trademark status for identification that is not obviously distinct or unique, but which has developed a secondary meaning over time that identifies it with the product or seller. The owner of a trademark has exclusive right to use it on the product it was intended to identify, and often on related products. Service marks receive the same legal protection as trademarks but are meant to distinguish services rather than products. A trademark registered under

the Lanham Act has nationwide protection. See § 1115 of the Act. Under the Lanham Act, a seller applies to register a trademark with the Patent and Trademark Office. The mark can already be in use or be one that will be used in the future.

**transaction:** A set of input data that triggers execution of a specific processor job. Usually manipulates data that may need to be rolled back to the original values if any part of the transaction fails. Transactions enable multiple users to access the same data concurrently. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**transport infrastructure:** The foundation for net-centric transformation in DoD.

**TRD:** Technical Requirements Document

**trigger:** In a DBMS, a trigger is a SQL procedure that initiates (fires) an action when an event (INSERT, DELETE, or UPDATE) occurs. Since triggers are event-driven specialized procedures, the DBMS stores and manages them. A trigger cannot be called or executed; the DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers maintain the referential integrity of data by changing the data in a systematic fashion.

**trusted path:** A communications path where: (1) There is reasonable confidence that there has not been any malicious alteration of the information; (2) The data are timely, meaning they originated within a small preceding period of time.

**TTP:** Tactics, Techniques, and Procedures

**tunneling:** Transporting IPv6 traffic through IPv4 networks by encapsulating IPv6 packet in IPv4 and vice-versa.

## U

**UCS:** Universal Multiple-Octet Coded Character Set

**UDDI:** Universal Description, Discovery, and Integration. An industry initiative to create a platform-independent, open framework for describing services, discovering businesses, and integrating business services using the Internet, as well as a registry. It is being developed by a vendor consortium. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**UDOP:** User-Defined Operation Picture

**UDP:** User Datagram Protocol

**UFS:** User-Facing Services. A software component that receives a UFS request from the portal. It returns a UFS response that formats the content for display, usually in a markup language such as HTML or WML, and produces visual output in a portlet.

**UML:** Unified Modeling Language. A standard notation for modeling real-world objects as a first step in developing an object-oriented design methodology. UML is defined by the Object Management Group (OMG). (Source: <http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html>)

**unassociated measurement report (UMR):** (1) A sensor measurement that has been processed by the originating sensor for clutter rejection and meets defined signal-to-noise parameters, but has not been associated with a track. (2) A measurement report from a

single sensor that has not been successfully associated with an existing composite or single-sensor track and which may be the initial detection of a new entity.

**Unicode:** A standard defined by the Unicode Consortium. Unicode uses a 16-bit code page that maps digits to characters in languages around the world. Because 16 bits covers 32,768 codes, Unicode is large enough to include all the world's languages, with the exception of ideographic languages that have a different character for every concept, such as Chinese. For more information, see <http://www.unicode.org/>. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**Universal Joint Task List (UJTL):** The comprehensive list of tasks at the Strategic and Operational levels of war. A menu of capabilities (mission-derived tasks with associated conditions and standards, i.e., the tools) that a joint force commander may select to accomplish the assigned mission. Once identified as essential to mission accomplishment, the tasks are reflected within the command joint mission essential task list.

**Universal Navy Task List (UNTL):** UNTL = UJTL + NTTL

**UNK:** Unknown (contact)

**URI:** Uniform Resource Identifier. An encoded address that represents any web resource, such as an HTML document, image, video clip, or program. As opposed to a URL or aURN, which are concrete entities, a URI is an abstract superclass. (Source: <http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html>)

**URL:** Uniform Resource Locator. A sequence of characters that represents information resources on a computer or in a network such as the Internet. This sequence of characters includes (1) the abbreviated name of the protocol used to access the information resource and (2) the information used by the protocol to locate the information resource. (Source: <http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html>)

**URN:** Uniform Resource Name. A name that uniquely identifies a web service to a client. (Source: <http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html>)

**USC:** Universal Multiple-Octet Coded Character Set

**use-case model:** A model that describes a system's functional requirements in terms of use cases. Consists of all the actors of the system and all the various use cases by which the actor interact with the system, thereby describing the total functional behavior of the system.

**use-case survey:** A list of names and perhaps brief descriptions of use cases associated with a system, component, or other logical or physical entity.

**use case:** A sequence of actions, performed by a system, that yields a result of value to a user. A set of actions, including variants, that a system performs that yields an observable result of value to a particular actor. (UML)

**user (security):** An individual or application program identity that has been authenticated. A user can have a set of roles associated with that identity, which entitles the user to access all resources protected by those roles.

**user agent:** A system entity that is used by an end user to access a web site. A user agent provides a runtime environment for distributed application components on the client

device. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)

**UTC:** Universal Time. Similar to GMT or Zulu times.

## V

**VBScript:** A programming language developed by Microsoft that is similar to JavaScript. It is used to embed code into HTML pages. It is actually a subset of Microsoft's Visual Basic.

**vendor:** Any person, organization, or automated asset that interfaces with the information environment as a service consumer or service provider.

**VoiceXML:** VoiceXML (VXML) is the W3C's standard XML format for specifying interactive voice dialogues between a human and a computer. It is fully analogous to HTML, and brings the same advantages of web application development and deployment to voice applications that HTML brings to visual applications. Just as HTML documents are interpreted by a visual web browser, VoiceXML documents are interpreted by a voice browser. A common architecture is to deploy banks of voice browsers attached to the public switched telephone network (PSTN) so that users can simply pick up a phone to interact with voice applications. VoiceXML has tags that instruct the voice browser to provide speech synthesis, automatic speech recognition, dialog management, and soundfile playback.

**VoIP:** Voice over Internet Protocol. a set of standards and technologies that allow voice to be transmitted over IP networks.

**VPF:** Vector Product Format

**VPN:** Virtual Private Network

**VTC:** Video TeleConferencing. A meeting among persons where telephony and closed-circuit television technologies are used simultaneously. Video teleconference communication is multi-way and synchronous, as it would be if all parties were in the same room. (Source: [http://en.wikipedia.org/wiki/Video\\_teleconference](http://en.wikipedia.org/wiki/Video_teleconference))

## W

**W2W:** Web-to-Web

**W3C:** World Wide Web Consortium. The international body that governs Internet standards. Its web site is <http://www.w3.org/>.

**WAP:** Wireless Application Protocol. WAP is an open international standard for applications that use wireless communication, such as Internet access from a mobile phone. WAP provides services equivalent to a web browser with some mobile-specific additions. It is specifically designed to address the limitations of very small portable devices. During its first years of existence WAP suffered from considerable negative media attention and has been criticised heavily for its design choices and limitations. (Source: <http://en.wikipedia.org/wiki/WAP>)

**WAR:** Web Application Archive. A JAR archive that contains a web module. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**warfare system:** All shipboard tactical systems and tactical mission-support systems, such as weapons, sensors, command and control, navigation, aviation support systems, mission

planning, intelligence, surveillance and reconnaissance, interior and exterior communications, topside design, and warfare system networks. (Source: N00178-04-R-2010, AircraftCarrier Warfare Systems Support)

**WCS:** Web Coverage Services or Web Coverage Server

**Web-DAV:** Web Distributed Authoring and Versioning

**web application:** A collection of components that can be bundled together and run in multiple containers from multiple vendors. -OR- An application written for the Internet, including those built with Java technologies such as Java Server Pages and servlets, and those built with non-Java technologies such as CGI and Perl. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**web browser:** A client program that initiates requests to a web server and displays the information that the server returns. (Source: <http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html>)

**web container:** A container that implements the web-component contract of the J2EE architecture. This contract specifies a runtime environment for web components that includes security, concurrency, life-cycle management, transaction, deployment, and other services. A web container provides the same services as a JSP container as well as a federated view of the J2EE platform APIs. A web container is provided by a web or J2EE server. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**web module:** A deployable unit that consists of one or more web components, other resources, and a web application deployment descriptor. The web module is contained in a hierarchy of directories and files in a standard web application format. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**web page:** A document created with HTML (HyperText Markup Language) that is part of a group of hypertext documents or resources available on the World Wide Web. Collectively, these documents and resources form what is known as a web site. You can read HTML documents that reside somewhere on the Internet or on your local hard drive with software called a web browser. Web pages can contain hypertext links to other places within the same document, to other documents at the same web site, or to documents at other web sites.

**web server:** Software that provides services to access the Internet, an intranet, or an extranet. A web server hosts web sites, provides support for HTTP and other protocols, and executes server-side programs (such as CGI scripts or servlets) that perform certain functions. In the J2EE architecture, a web server provides services to a web container. For example, a web container typically relies on a web server to provide HTTP message handling. The J2EE architecture assumes that a web container is hosted by a web server from the same vendor, so it does not specify the contract between these two entities. A web server can host one or more web containers. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**web server provider:** A vendor that supplies a web server. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**web service:** An application that exists in a distributed environment, such as the Internet. A web service accepts a request, performs its function based on the request, and returns a response. The request and the response can be part of the same operation, or they can occur separately, in which case the consumer does not need to wait for a response. Both

the request and the response usually take the form of XML, a portable data-interchange format, and are delivered over a wire protocol, such as HTTP. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>) -OR- A web service is a software application or component that is identified by a URI and can be accessed over the Internet. It uses a vendor/platform/language-neutral data interchange format to invoke the service and supply the response. Web services use a message exchange pattern that is sufficiently well defined to be processed by a software application. Its interfaces and binding are capable of being defined, described, and discovered by XML artifacts. It supports direct interactions with other software applications using XML-based messages via Internet-based protocols.

**web site:** A web site, website, or WWW site (often shortened to just "site") is a collection of web pages: i.e., HTML/XHTML documents accessible via HTTP on the Internet. All publicly accessible web sites in existence comprise the World Wide Web. The pages of a web site are accessed from a common root URL, the homepage, and usually reside on the same physical server. The URLs of the pages organize them into a hierarchy, although the hyperlinks between them control how the reader perceives the overall structure and how the traffic flows between the different parts of the site. (Source: [http://en.wikipedia.org/wiki/web\\_site](http://en.wikipedia.org/wiki/web_site))

**well-formed:** An XML document that is syntactically correct. It does not have any angle brackets that are not part of tags, all tags have an ending tag or are themselves self-ending, and all tags are fully nested. Knowing that a document is well formed makes it possible to process it. However, a well-formed document may not be valid. To determine that, you need a validating parser and a DTD. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**WfMC:** Workflow Management Coalition

**WFS:** Web Feature Services or Web Feature Server

**WML:** Wireless Markup Language. WML is the primary content format for devices that implement the WAP (Wireless Application Protocol) specification based on XML, such as mobile phones. (Source: [http://en.wikipedia.org/wiki/Wireless\\_Markup\\_Language](http://en.wikipedia.org/wiki/Wireless_Markup_Language))

**WMS:** Web Mapping Service

**WNS:** Web Notification Services

**workflow application:** One where various applications and components must process data to complete a task. For example, consider a purchase order that moves through various departments for authorization and eventual purchase. The orders may be treated as messages, which are put into various queues for processing. A workflow process involves constant change and update. You can introduce new components into the operation without changing any code.

**WS-I:** Web Services Interoperability

**WSDL:** Web Services Description Language. An XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.

**WSRM:** Web Services Reliable Messaging. XACML supports exchange of access control information using XML.

**WSRP:** Web Services for Remote Portlets. The WSRP specification defines a web-service interface for interacting with interactive presentation-oriented web services. It has been produced through the joint efforts of the Web Services for Interactive Applications (WSIA) and Web Services for Remote Portals (WSRP) OASIS Technical Committees. Scenarios that motivate WSRP/WSIA functionality include: (1) portal servers providing portlets as presentation-oriented web services that can be used by aggregation engines; (2) portal servers consuming presentation-oriented web services provided by portal or non-portal content providers and integrating them into a portal framework. (Source: <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>)

**WSRP service:** Presentation-oriented, interactive web services that can be aggregated by consuming applications. (Source: OASIS WSRP Specification 1.0 Glossary)

**WWW:** World Wide Web. The World Wide Web ("WWW," or simply "web") is an information space in which items of interest, referred to as resources, are identified by global identifiers called Uniform Resource Identifiers (URI). The term is often mistakenly used as a synonym for the Internet, but the web is actually a service that operates over the Internet. (Source: [http://en.wikipedia.org/wiki/World\\_Wide\\_web](http://en.wikipedia.org/wiki/World_Wide_web))

## X

**XACML:** eXtensible Access Control Markup Language. An OGC-compliant interface layer that runs on both C2PC and COE 4.x.

**Xalan-Java:** A XSLT processor made by Apache.

**Xalan processor:** An XSLT processor that is part of the Apache project. (Source: <http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html>)

**XIL:** XIS Integration Layer

**XKMS:** XML Key Management Specification

**XML:** eXtensible Markup Language. A markup language that allows you to define tags (markup) to identify the content, data, and text in XML documents. It differs from HTML, the markup language most often used to present information on the Internet. HTML has fixed tags that deal mainly with style or presentation. An XML document must undergo a transformation into a language with style tags under the control of a style sheet before it can be presented by a browser or other presentation mechanism. Two types of style sheets used with XML are CSS and XSL. Typically, XML is transformed into HTML for presentation. Although tags can be defined as needed in the generation of an XML document, you can use a document type definition (DTD) to define the elements allowed in a particular type of document. A document can be compared by using the rules in the DTD to determine its validity and to locate particular elements in the document. A web services application's J2EE deployment descriptors are expressed in XML with schemas defining allowed elements. Programs for processing XML documents use SAX or DOM APIs. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**XML schema:** A database-inspired method for specifying constraints on XML documents using an XML-based language. Schemas address deficiencies in DTDs, such as the inability to constrain the kinds of data that can occur in a particular field. Because schemas are founded on XML, they are hierarchical. Thus it is easier to create an unambiguous

specification, and it is possible to determine the scope over which a comment is meant to apply. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**XMPP:** eXtensible Messaging Presence Protocol

**XPath:** XML Path. An XSL sublanguage designed to uniquely identify or address parts of a source XML document, for use with XSLT. XPath also provides basic facilities for manipulation of strings, numbers, and Booleans. (Source: <http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html>)

**XSD:** XML Schema Definition. The W3C specification for defining the structure, content, and semantics of XML documents. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**XSL:** eXtensible Stylesheet Language. A standard that lets you do the following: (1) Specify an addressing mechanism, so that you can identify the parts of an XML document that a transformation applies to (XPath). (2) Specify tag conversions, so that you can convert XML data into different formats (XSLT). (3) Specify display characteristics, such as page sizes, margins, font heights and widths, and the flow objects on each page. Information fills in one area of a page and then automatically flows to the next object when that area fills up. That allows you to wrap text around pictures, or continue a newsletter article on a different page (XSL-FO). (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**XSL-FO:** eXtensible Stylesheet Language – Formatting Objects. XSL-FO is a language that specifies the physical layout, coloring, and typography of XML documents for screen, print, and other media. In this sense it is similar to CSS, but it is more powerful and flexible, particularly with regard to pagination and scrolling. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**XSLT:** eXtensible Style Language Transformations. An XML document that controls the transformation of an XML document into another XML document or HTML. The target document often has presentation-related tags dictating how it will be rendered by a browser or other presentation mechanism. XSLT was formerly a part of XSL, which also included a tag language of style flow objects. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**XSLTC:** eXtensible Style Language Transformations Compiler. A compiling version of XSLT. (Source: <http://java.sun.com/j2ee/1.4/docs/glossary.html>)

**XTCF:** eXtensible Tactical C4I Framework



---

# Index

## A

Apache Ant .....	281, 299
Apache Axis.....	282, 299
Apache Xalan-Java .....	284
APIs	
Geobject .....	89, 91
GO-1 .....	89
guidance .....	5
JNDI.....	45
UDDI4J.....	297
applications	
application servers.....	108
applications services.....	130
namespace management.....	262
architecture	
C/JMTK .....	104, 106
C/JMTK toolkit.....	108
JWC .....	128
OGC web services distributed.....	85
thick clients .....	144
thin clients.....	143
ArcIMS .....	108, 111, 130
ArcObjects .....	108
ArcSDE.....	108, 131
ATLAS .....	123
automated testing tools .....	262

## B

build lists.....	267
build process .....	281

## C

C/JMTK	
architecture.....	104, 106
C2PC to C/JMTK WFS example .....	126
components .....	108
ESRI ArcIMS and web client in Tomcat example .....	111
ESRI WFS in Tomcat example .....	116
Extended User Community .....	105
Foreign Military Sales.....	105
JMTK vs. C/JMTK .....	106
licenses.....	105
references .....	104
C/JMTK .....	104
C/JMTK .....	107
C2IEDM .....	31
C2PC	
C2PC to C/JMTK WFS example .....	126
C2PC to NOSWC WFS example .....	124
C2PC.....	123
catalina_home environment variable .....	284
catalina_opts environment variable.....	284
cell phones .....	146
CIIL .....	123

Codewarrior IDE.....	147
COE-M build lists .....	267
componentizing .....	6
COMPOSE software list .....	277
conceptual models .....	31

## D

data exposing community.....	133
data servers.....	108
Data Source Interfaces (DSIs)	
GO-1/Geobject API in NOSWC example .....	134
Members DSI .....	141
NOSWC .....	94
updating attributes .....	143
Data Source Interfaces (DSIs).....	134
data tier	
data modeling .....	31
implementations .....	30
normalization.....	31
OS file systems.....	30
XML.....	33
data tier.....	29
databases	
decoupling from applications .....	30
hierarchical .....	30
native XML .....	30
niche .....	30
object-oriented.....	30
relational.....	30
development communities.....	133
differentiated services .....	46
directories	
databases, differences between.....	40
JNDI .....	45
LDAP .....	41
directories.....	40
disclaimer .....	3
discovery	
Java JNDI to LDAP example .....	41
references .....	46
UDDI.....	45
discovery .....	40
Document Literal style .....	19, 134
domain analysis.....	31

## E

enterprise services .....	39
examples	
C2PC to C/JMTK WFS .....	126
ESRI ArcIMS and web client in Tomcat .....	111
ESRI WFS in Tomcat example.....	116
ESRI WFS to NOSWC WFS client.....	122
GO-1/Geobject APIs in NOSWC .....	134
Java JNDI to LDAP.....	41
jUDDI registry.....	287
SCA-compliant software component.....	56

vendor neutrality .....	2
WFS to NOSWC in JBoss.....	94
WFS to NOSWC in Tomcat.....	101
<b>F</b>	
feature store service .....	129
<b>G</b>	
Geobject	
sample DSI.....	134
Geobject.....	89, 90, 91
Geography Markup Language .....	87
GIS	
architecture.....	133
ATLAS.....	123
C/JMTK .....	104
examples .....	93
Geobject .....	90
NESI goals .....	83
OGC WS distributed architecture.....	85
thick clients .....	144
Web Coverage Service (WCS).....	89
Web Feature Service .....	86
GIS.....	83
GIS.....	83
GO-1 APIs .....	89
guaranteed services .....	46
<b>I</b>	
image processing.....	107
<b>J</b>	
Java	
Apache Ant .....	281
Java Community Process .....	266
Java JNDI to LDAP example.....	41
Java Standard Secure Socket Edition (JSSE) .....	299
Java.net .....	266
JavaMail.....	299
JNDI.....	45
jUDDI .....	287
Sun Developer's Network.....	266
Xalan-Java.....	284
Xerxes2 Java Parser .....	285
JMTK.....	106
JNDI	
guidance .....	45
JNDI .....	45
JTRS	
components .....	53
radio composition.....	51
references .....	79
SCA-compliant software component example .....	56
software components.....	54
JTRS .....	49, 50
jUDDI	
configuring.....	288
creating a UDDI publisher .....	297
database.....	290
installing.....	287
testing.....	290, 297
web server configuration.....	295
jUDDI .....	287

<b>JWC</b>	
application services .....	130
architecture .....	128
best practices .....	132
client.....	131
components.....	131
feature store service.....	129
mapping service.....	130
mediator service .....	131
references .....	132
security .....	131
JWC.....	127
<b>L</b>	
LDAP	
Java JNDI to LDAP example .....	41
LDAP .....	41
legacy wrapping .....	6
log4j.properties.....	288
logical models .....	31
<b>M</b>	
mapping service .....	130
mediator service .....	131
middle tier .....	9, 10
mobile code.....	265
mobile devices	
best practices .....	145
PalmOS.....	147
mobile devices.....	145
mobile devices.....	146
multi-tier architectures .....	279
<b>N</b>	
namespaces.....	262
Navy	
Navy Enterprise Portal (NEP) Architecture.....	279
Navy Open Source WebCOP (NOSWC).....	94
OGC architecture initiatives .....	85
thick clients, migrating to OGC.....	144
thin clients, migrating to OGC.....	143
Navy .....	267
NESI	
correspondence with other initiatives .....	279
documentation disclaimer.....	3
documentation structure .....	3
references .....	257
releasability statement .....	2
vendor neutrality disclaimer .....	2
NESI.....	1
network security .....	279
networks.....	39
normalization .....	31
NOSWC	
C2PC to NOSWC WFS example .....	124
ESRI WFS to NOSWC WFS client example .....	122
GO-1/Geobject APIs in NOSWC example.....	134
WFS to NOSWC in JBoss example.....	94
WFS to NOSWC in Tomcat example.....	101
NOSWC .....	94
<b>O</b>	
open-source tools	

- Apache Ant ..... 281
- Apache Axis ..... 282
- jUDDI ..... 287
- Tomcat ..... 284
- Xalan-Java ..... 284
- Xerxes2 Java Parser ..... 285
- open-source tools ..... 281
- P**
- PalmOS ..... 145, 147
- parsing XML ..... 36
- physical models ..... 31
- POSE emulator ..... 147
- PRC-TOOLS GCC ..... 147
- Q**
- QoS
  - implementations ..... 46
  - references ..... 47
- QoS ..... 46
- R**
- reference implementations ..... 83
- references ..... 257
- releasability statement ..... 2
- rendering control community ..... 133
- RPC style ..... 19
- S**
- security
  - JNDI ..... 45
  - JWC ..... 131
  - LDAP ..... 41
  - network security guidance ..... 279
  - testing ..... 262
- sensor modeling ..... 107
- service registries ..... 40, 45
- SOAP ..... 19
- software ..... 267, 277
- Software Communication Architecture (SCA)
  - device interface ..... 50
  - hardware configuration ..... 55
  - JTRS software ..... 51
  - member variables ..... 50
  - resource interface ..... 50
  - software component example ..... 56
- Software Communication Architecture (SCA) ..... 49
- Software Communication Architecture (SCA) ..... 50
- Software Communication Architecture (SCA) ..... 51
- Software Communication Architecture (SCA) ..... 55
- Solaris ..... 267
- Sun Developer's Network ..... 266
- T**
- testing ..... 262
- thick clients
  - C/JMTK components ..... 108
  - guidance ..... 144
  - migrating to OGC ..... 144
- thin clients
  - C/JMTK components ..... 108
  - migrating to OGC ..... 143
- tiers
- JWC architecture ..... 128
- transport ..... 39
- tiers ..... 9, 10, 29
- Tomcat ..... 101, 111, 116, 284
- U**
- UDDI
  - jUDDI ..... 287
  - scripts for database tables ..... 290
  - UDDI browser ..... 298, 299
  - UDDI4J ..... 297
- UDDI ..... 45
- UDDI4J ..... 297, 299
- user registries ..... 40
- V**
- vendors
  - vendor neutrality ..... 2
  - Web Feature Service (WFS) ..... 86
- W**
- Web Coverage Service (WCS) ..... 86, 89
- web services
  - .NET ..... 18, 27
  - componentizing ..... 6
  - feature store service ..... 129
  - SOAP ..... 19
- web services ..... 15
- web services ..... 24
- web.xml ..... 288
- webCOPs
  - clients ..... 131
  - overlays ..... 142
- webCOPs ..... 128
- WFS
  - C2PC to C/JMTK WFS example ..... 126
  - C2PC to NOSWC WFS example ..... 124
  - communication models ..... 87
  - ESRI WFS in Tomcat example ..... 116
  - ESRI WFS to NOSWC WFS client example ..... 122
  - feature store service ..... 129
  - GML ..... 87
  - public interfaces ..... 87
  - vendors ..... 86
  - WFS to NOSWC in JBoss example ..... 94
  - WFS to NOSWC in Tomcat example ..... 101
- WFS ..... 86
- WFS ..... 86
- wireless devices ..... 145, 146
- wrapper classes ..... 33
- X**
- Xalan-Java ..... 284, 299
- Xerxes2 Java Parser ..... 285, 299
- XIL ..... 123, 126
- XML
  - parsers ..... 33
  - parsing ..... 36
  - schema processor ..... 285
  - web services ..... 15, 24
  - Xalan-Java ..... 284
  - Xerxes2 Java Parser ..... 285
- XML ..... 33

---