



# Guide to Developing with the GCSS-AF Integration Framework Appendix A

Prepared for:  
**Department of the Air Force**  
**Headquarters Standard Systems Group (HQ SSG)**  
**Maxwell Air Force Base - Gunter Annex**  
**Montgomery, Alabama**

Contract Number: **F01620-96-D-0004**

Document Number: **GCSS-REPORT-1997-0011 Appendix A**  
Version: **3.8**  
Date: **08/10/01**



-*Lockheed Martin Systems Integration-Owego*  
1801 Route 17C  
Owego, NY 13827-3998

This Page Intentionally Left Blank.

DOCUMENT CHANGE HISTORY					
VERSION			CHANGE DESCRIPTION (REQUIRED FOR CHANGES AFFECTING TPM, REQUIREMENTS, CONFIGURATION, COST & SCHEDULE)		
NO.	APPROVAL	DATE	CHANGE DOC.	SECTION	NARRATIVE (OF ITEMS AFFECTED)
0.5		12/11/00			INITIAL DRAFT
1.0		1/12/01			RE-WRITE
1.5		02/21/01			RE-WRITE
2.0		03/09/01			DRAFT
2.5		03/16/01			DRAFT
3.0		03/23/01			DRAFT
3.8		08/08/01			IF 2.3 UPDATES Major updates to sections: 3.2.6, 3.3, 3.4, 3.5.3.

This Page Intentionally Left Blank

## TABLE OF CONTENTS

1.	APPENDIX A - UTILITY/HELPER CLASS AND API DESCRIPTIONS.....	1
1.1	INTRODUCTION.....	1
1.1.1	Purpose of this Document .....	1
1.1.2	Objectives.....	1
1.1.3	Scope.....	1
2.	CLASS AND API DESCRIPTIONS.....	4
2.1	TMESSAGE .....	4
2.2	COM.LMFS.FRAMEWORK.AMITHelpers.* .....	10
2.3	COM.LMFS.FRAMEWORK.BOD.BOD (OAG BOD BASE CLASS).....	25
2.4	COM.LMFS.FRAMEWORK.PUBSUB.MQRFH - PUBLISH/SUBSCRIBE HELPERS.....	36
2.5	COM.LMFS.FRAMEWORK.SERVLET.IFSERVLET .....	47
2.6	COM.LMFS.FRAMEWORK.* - SERVLET UTILITY HELPERS.....	51
3.	SECURITY HELPERS .....	52
3.1	COM.LMFS.FRAMEWORK.LDAPHELPER.* .....	52
3.1.1	com.lmfs.framework.LDAPHelper.LDAPHelper .....	53
3.1.2	com.lmfs.framework.LDAPHelper.LDAPResultsHelper .....	54
3.1.3	com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator.....	56
3.1.4	com.lmfs.framework.LDAPHelper.LDAPTests .....	57
3.2	COM.LMFS.FRAMEWORK.SECURITY.* .....	57
3.2.1	com.lmfs.framework.security.IFSecurityException.....	58
3.2.2	com.lmfs.framework.security.IFMenuPOSCreator.....	58
3.2.3	com.lmfs.framework.security.PDCheckParser.....	59
3.2.4	com.lmfs.framework.security.PDCheckEnvironment.....	62
3.2.5	com.lmfs.framework.security.AuthnInfo .....	63
com.lmfs.framework.security.aznAPIInitialize		
3.3	COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.* .....	64
3.3.1	com.lmfs.framework.security.PDAuthn.PDAuthnInfo .....	65
3.3.2	com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl .....	66
3.3.3	com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl.....	6869
3.3.4	com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl .....	70
3.3.5	com.lmfs.framework.security.PDAuthn.EJB.PDEJBAuthnInfoImpl.....	70
3.4	COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.* .....	71
3.4.1	com.lmfs.framework.security.PDAuthz.PDCheckAuthz .....	72
3.4.2	com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException .....	73
com.lmfs.framework.security.PDAuthz.Application		
3.4.3	com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl .....	74
3.4.4	com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl .....	77
3.4.5	com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl .....	78
3.4.6	com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl .....	78
3.5	COM.LMFS.FRAMEWORK.UTIL.* .....	81
3.5.1	com.lmfs.framework.util.StringExt .....	81
3.5.2	com.lmfs.framework.util.ArrayExt .....	82
3.5.3	com.lmfs.framework.util.PropertiesExt .....	83
3.6	SESSIONINFO STRUCTURES .....	8485
3.7	IFCBSECURITYINFO.....	86
3.7.1.1.1	Authentication.....	8889

3.7.1.1.2 Providing Access Control Checks in MA Software.....	90
---	----

## TABLE OF FIGURES

FIGURE 1: EXAMPLE OF CODE USING TMOUTBOUNDCOPYHELPER .....	5
FIGURE 2: EXAMPLE OF CODE USING THE TMINBOUND MESSAGE INTERFACE .....	6
FIGURE 3: CALLING ORDER OF SENDANDFORGET.JAVA .....	11
FIGURE 4: EXAMPLE CODE FROM THE FILEWRAPPER COMPONENT FOR SENDANDFORGET.JAVA ..	11
FIGURE 5: CALLING ORDER OF REQUESTRESPONSE.JAVA .....	12
FIGURE 6: EXAMPLE CODE EXCERPTED FROM THE FILEWRAPPER COMPONENT FOR REQUESTRESPONSE.JAVA .....	12
FIGURE 7: CALLING ORDER OF THE RECIEVER.JAVA .....	13
FIGURE 8: EXAMPLE CODE OF RECIEVER.JAVA .....	13
FIGURE 9: CALLING ORDER FOR PUBLISH.JAVA .....	14
FIGURE 10: EXAMPLE CODE FOR PUBLISH.JAVA .....	14
FIGURE 11: CALLING ORDER FOR SUBSCRIBE.JAVA .....	14
FIGURE 12: EXAMPLE CODE FOR SUBSCRIBE.JAVA .....	15
FIGURE 13: BUSINESS OBJECT DOCUMENT (BOD) ARCHITECTURE .....	26
<b>FIGURE 14: AN EXAMPLE OF AN APPLICATION USING THE BASE BOD CLASS TO DETERMINE HOW TO HANDLE AN INCOMING UNSOLICITED MESSAGE:</b> .....	28
FIGURE 15: EXAMPLE OF THE BOD CLASS USING APPLICATION PROVIDED EXTENSIONS .....	32
FIGURE 16: EXAMPLE OF PUBLISH/SUBSCRIBE HELPER CODE .....	39
<b>FIGURE 17: EXAMPLE THAT SHOWS USE OF PUBLISH/SUBSCRIBE JAVA BINDINGS</b> .....	40
FIGURE 18 IFSERVLET EXTENDED CLASS DIAGRAM .....	48
FIGURE 19: LDAPHELPER EXAMPLE FROM MENU.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT .....	53
FIGURE 20: LDAPRESULTSHelper CODE EXAMPLE FROM THE MENU.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT .....	55
FIGURE 21: EXAMPLE OF PDCKECKPARSER CODE .....	60
FIGURE 22: EXAMPLE OF <IVCHECK> HTML TAG .....	60
FIGURE 23: EXAMPLES OF GETCLIENTIDENT, GETGROUPS, AND GETPAC FROM THE IFSERVLET GETSESSIONINFOSTRUCT METHOD .....	69
FIGURE 24: EXAMPLE OF CONSTRUCTING AND INITIALIZING A PDSERVLETAUTHZIMPL OBJECT TAKEN FROM THE MENUSERVLET.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT .....	79
FIGURE 25: EXAMPLE OF PDCKECKAUTHORIZATIONPAC TAKEN FROM THE MENU.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT .....	80
FIGURE 26: EXAMPLE OF PROPERTIESEXT FROM THE MENUSERVLET.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT .....	83
FIGURE 27: EXAMPLE OF USERSECINFOSTRUCT CODE .....	85
FIGURE 28: EXAMPLE OF APPSESSIONINFOSTRUCT CODE .....	85
FIGURE 29: EXAMPLE OF USERSSESSIONINFOSTRUCT CODE .....	86
FIGURE 30: EXAMPLE OF SESSIONINFOSTRUCT CODE .....	86
FIGURE 31: IFCBSECURITYINFOSTRUCT .....	90
FIGURE 32: EXAMPLE IDL FROM PDSESSIONMODULE OF THE PDC TEST COMPONENT .....	91
FIGURE 33: EXAMPLE IDL FROM COM_LMFS_FRAMEWORK_TESTCOMPONENTS_REQUSITION_ORDERINGTIE.JAVA OF THE REQUISITION COMPONENT TEST COMPONENT .....	92
FIGURE 34: EXAMPLE OF OBTAINING PARAMETERS REQUIRED BY THE GCSSDECISIONACCESSALLOWED METHOD .....	94
FIGURE 35: CODE EXAMPLE OF FINDING IFCBSECINFO HOME .....	95
FIGURE 36: CODE EXAMPLE OF CREATING IFCBSECURITYINFO INSTANCE .....	96

FIGURE 37: CODE EXAMPLE OF INVOKING THE GCSAFDECISIONACCESSALLOWED METHOD .....97

## TABLE OF TABLES

<b>TABLE 1: INTEGRATION FRAMEWORK PROVIDED JAR FILE CONTENTS .....</b>	2
<b>TABLE 2: TMMESSAGE .....</b>	7
<b>TABLE 3: COM.LMFS.FRAMEWORK.AMIHELPERS.* .....</b>	15
<b>TABLE 4: COM.LMFA.FRAMEWORK.BOD.BOD (BASE CLASS).....</b>	33
<b>TABLE 5: PUBSUBHELPERS (BINDINGS FOR C APPLICATIONS) .....</b>	41
<b>TABLE 6: COM.LMFS.FRAMEWORK.SERVLET.IFSERVLET .....</b>	49
<b>TABLE 7: COM.LMFS.FRAMEWORK.* - SERVLET UTILITY HELPERS .....</b>	51
TABLE 8: COM.LMFS.FRAMEWORK.LDAPHELPER.LDAPHELPER .....	53
TABLE 9: COM.LMFS.FRAMEWORK.LDAPHELPER.LDAPRESULTSHelper .....	55
<b>TABLE 10: COM.LMFS.FRAMEWORK.LDAPHELPER.LDAPMENUENTRYCOMPARATOR .....</b>	56
<b>TABLE 11: COM.LMFS.FRAMEWORK.LDAPHELPER.LDAPTESTS .....</b>	57
TABLE 12: COM.LMFS.FRAMEWORK.SECURITY.IFSECURITYEXCEPTION .....	58
TABLE 13: COM.LMFS.FRAMEWORK.SECURITY.IFMENUPOSCREATOR .....	59
TABLE 14: COM.LMFS.FRAMEWORK.SECURITY.PDCHECKPARSER .....	61
TABLE 15: COM.LMFS.FRAMEWORK.SECURITY.PDCHECKENVIRONMENT .....	62
TABLE 16: COM.LMFS.FRAMEWORK.SECURITY.AUTHNINFO .....	64
TABLE 17: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.PDAUTHNINFO .....	65
TABLE 18: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.PDAUTHNINFOIMPL .....	66
TABLE 19: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.SERVLET.PDSERVLETAUTHNINFOIMPL .....	69
TABLE 20: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.APPLICATION.PDAPPAuthNINFOIMPL .....	70
TABLE 21: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.EJB.PDEJBAUTHNINFOIMPL .....	70
TABLE 22: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.PDCHECKAUTHZ .....	72
TABLE 23: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.PDAUTHZUNAUTHORIZEDEXCEPTION ..	73
TABLE 24: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.PDCHECKAUTHZIMPL .....	74
TABLE 25: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.CB.PDCBAUTHZIMPL .....	77
TABLE 26: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.EJB.PDEJBAUTHZIMPL .....	78
TABLE 27: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.SERVLET.PDSERVLETAUTHZIMPL .....	80
TABLE 28: COM.LMFS.FRAMEWORK.UTIL CLASS .....	81
TABLE 29: COM.LMFS.FRAMEWORK.UTIL.STRINGEXT .....	81
TABLE 30: COM.LMFS.FRAMEWORK.UTIL.ARRAYEXT .....	82
TABLE 31: COM.LMFS.FRAMEWORK.UTIL.PROPERTIESEXT .....	83
TABLE 32: IFCBSECURITYINFO .....	86
TABLE 33: MA AUTHENTICATION SCENARIOS .....	89
TABLE 34: PARAMETERS OF THE GCSAF/DECISIONACCESSALLOWED MET HOD .....	92

# 1. Appendix A - Utility/Helper Class and API Descriptions

## 1.1 Introduction

GCSS-AF provides a component-based Reference Architecture Framework that serves as the Integration and Application Framework Layers for GCSS-AF functional capabilities consistent with the Defense Information Infrastructure Common Operating Environment (DII COE), the Joint Technical Architecture - Air Force (JTA-AF), and based on commercial open standards. The GCSS-AF Reference Architecture Framework also provides common interfaces for those functions that either directly or indirectly support Command and Control (C2) or share information with C2 Systems.

It is assumed that the reader is familiar with the GCSS-AF Integration Framework and has at least a high level understanding of the services and facilities that it provides. This document is to be used as a reference regarding how to use those services and facilities.

### 1.1.1 Purpose of this Document

The purpose of this Appendix is to provide Application Developers with a description and examples of intended use of the helper and utility classes that are provided by the GCSS-AF Integration Framework (IF). This Appendix is also intended to direct developers to additional information required to develop applications employing the IF capabilities.

### 1.1.2 Objectives

The primary objective of this document is to provide the application developer with the information required in order to take advantage of the services that are provided by the GCSS-AF Integration Framework. Once familiar with the design guidance and application development approaches identified in the main body the [Guide to Developing with the GCSS-AF Integration Framework](#), an application developer should be able to use this appendix as an API dictionary and reference.

### 1.1.3 Scope

This document contains two main sections:

#### Section 2

This section contains the description of the utility and helper classes that are generally to be used during the development of Presentation and Business layer application components.

#### Section 3

This section contains the descriptions of the *utility* and *helper* classes that allow applications to make use of the security services provided by the IF.

[Table 1: Integration Framework Provided jar File Contents](#) [Table 1: Integration Framework Provided jar File Contents](#) identifies each of the utility and helper classes that are provided by the

IF, the jar file that each is contained in, and a short description of the intended purpose. Refer to the appropriate section of this appendix for additional details.

**Table 1: Integration Framework Provided jar File Contents**

Utility/Helper Package	Contained in.....	Short Description
TMMessages	N/A	Provides MQSeries communication support from within a business application deployed in the component broker.
com.lmfs.framework.amiHelpers.*	IFSServices.jar	A set of APIs that wrapper the OAMAS Application Messing Interface. These wrappers are used by applications executing outside the application server.
<u><a href="#">com.lmfs.framework.BOD.BOD (OAG BOD Base Class)</a></u>  <u><a href="#">The Open Application Group's (OAG) Business Object Document (BOD) is the architecture used to communicate messages or business documents between software applications or components. Each BOD includes supporting details to enable the destination Business application to accomplish the action.</a></u>  <u><a href="#">The BOD consists of two areas as shown in Figure 13.</a></u>  <ul style="list-style-type: none"> <li>• <u><a href="#">The Control Area</a></u></li> <li>• <u><a href="#">The Business Data Area.</a></u></li> </ul> <u><a href="#">com.lmfs.framework.BOD.BOD (OAG BOD Base Class)</a></u>  <u><a href="#">The Open Application Group's (OAG) Business Object Document (BOD) is the architecture used to communicate messages or business documents between software applications or components. Each BOD includes supporting details to enable the destination Business application to accomplish the action.</a></u>  <u><a href="#">The BOD consists of two areas as shown in Figure 13.</a></u>  <u><a href="#">? The Control Area</a></u> <u><a href="#">? The Business Data Area.</a></u>	IFSServices.jar	Base class provided by the Integration Framework that supports the building and parsing of OAG compliant BODs
	IFSServices.jar	Provides an API that encapsulates the

Utility/Helper Package	Contained in.....	Short Description
<a href="#">com.lmfs.framework.pubsub.MQRFH-Publish/Subscribe Helpers</a> <a href="#">com.lmfs.framework.pubsub.MQRFH-Publish/Subscribe Helpers</a>		MQSeries publish and subscribe message constructs and formats.
com.lmfs.framework.servlet.IFServlet	com_lmfs_framework_servlet.jar	A Servlet base class that provides the implementation of a set of services required by Servlets built using the Integration Framework, particularly security. This base class is extended by application developers to create application specific Servlets.
com.lmfs.framework.* - Servlet Utility Helpers	com_lmfs_framework.jar	Provides a set of classes to be used by applications for logging into the WAS-EE environment and for handling property file.
com.lmfs.framework.LDAPHelper.*	IFSServices.jar	Provides a set of classes that help simplify connecting to and interfacing with an LDAP repository.
com.lmfs.framework.security.*	IFSServices.jar	Provides a set of classes that may be used by applications to perform authorization checks and filtering of HTML documents.
<a href="#">com.lmfs.framework.security.aznAPIInitialize</a>  <u><a href="#">aznAPIInitialize</a> is an interface that describes the standard set of routines for initializing the azn API used within the PDAuthnInfoImpl or PDCheckAuthzImpl classes.</u>	IFSServices.jar	Provides an interface for retrieving authentication information.
<b>Table 17:</b> <a href="#">com.lmfs.framework.security.aznAPIInitialize</a>		
<a href="#">com.lmfs.framework.security.aznAPIInitialize</a>		
<a href="#">public init (IV.Auth.AZNAPI)</a>		
• <a href="#">In apiObj-A valid aznAPI to set to the aznAPI.</a>	<a href="#">Calls initApi to initialize the aznAPI (sets the azn object using an already-initialized azn object).</a>	We assume that the sent aznAPI object is initialized.
<a href="#">public init (java.util.Properties)</a>		
• <a href="#">Prp java.util.Properties-Properties to initialize the aznAPI with.</a>		<a href="#">Initialize the aznAPI using the properties passed.</a>
<a href="#">public init (com.lmfs.framework.util.PropertiesExt)</a>		
• <a href="#">In com.lmfs.framework.util.PropertiesExt prp-PropertiesExt containing the properties to initialize the aznAPI with.</a>		<a href="#">Initialize the aznAPI using the propertiesExt passed.</a>
<a href="#">public init (java.util.ResourceBundle)</a>		
• <a href="#">In java.util.ResourceBundle rb- Resource Bundle containing the properties to initialize the aznAPI with.</a>		<a href="#">Initialize the aznAPI using the ResourceBundle passed.</a>
<a href="#">com.lmfs.framework.security.PDAuthn.*</a>		

Utility/Helper Package	Contained in.....	Short Description
<code>com.lmfs.framework.security.aznAPIInitialize</code>		
<code>aznAPIInitialize</code> is an interface that describes the standard set of routines for initializing the azn API used within the <code>PDAuthnInfoImpl</code> or <code>PDCheckAuthzImpl</code> classes.		
<b>Table 17:</b> <code>com.lmfs.framework.security.aznAPIInitialize</code>		
<code>com.lmfs.framework.security.aznAPIInitialize</code>		
<code>public init(IV.Auth.AZNAPI)</code>		
? In <code>apiObj</code> A valid aznAPI to set to the aznAPI.	Calls <code>initApi</code> to initialize the aznAPI (sets the azn object) using an already initialized azn object). We assume that the sent aznAPI object is initialized.	
<code>public init(java.util.Properties)</code>		
? Prp <code>java.util.Properties</code> Properties to initialize the aznAPI with.	Initialize the aznAPI using the properties passed.	
<code>public init(com.lmfs.framework.util.PropertiesExt)</code>		
? In <code>PropertiesExt</code> com.lmfs.framework.util.PropertiesExt prp- PropertiesExt containing the properties to initialize the aznAPI with.	Initialize the aznAPI using the propertiesExt passed.	
<code>public init(java.util.ResourceBundle)</code>		
? In <code>java.util.ResourceBundle</code> rb- Resource Bundle containing the properties to initialize the aznAPI with.	Initialize the aznAPI using the ResourceBundle passed.	
<code>com.lmfs.framework.security.PDAuthn.*</code>		
<code>com.lmfs.framework.security.PDAuthz.*</code> <code>com.lmfs.framework.security.PDAuthz.*</code>	IFSServices.jar	Provides an interface for authorization checks. This allows for an abstract implementation of the actual authorization engine to allow for ease of use within an application or Servlet
<code>com.lmfs.framework.util.*</code>	IFSServices.jar	A set of utility classes that extend the functionality of several standard classes provided by Java.
IFCBSecurityInfo	N/A	Provides application authentication and authorization support to EJB and CORBA components.

## 2. Class and API Descriptions

### 2.1 TMMessage

The **TextMessage** Component Broker application, also known at **TMMessage**, contains two classes, **TMOutbound** and **TMInbound**, that provide access to the attributes of messages (such as priority, encoding, and format) which are to be sent or received using MQSeries from within a Component Broker component. Note that in order to use this component in your server, your

server in Component Broker must be configured in SMUI to use the **TextMessageApp**, **iDXAAAServices** and **iMQAAServices** applications.

When sending messages from an application the **TMOOutboundCopyHelper** sets the desired attributes of the outgoing message. You would use the **TMOOutboundCopyHelper**, for example, to set the queue to which you want to put the message, the message's priority, or the encoding of the message. The *copy* helper is then converted to a string and ultimately passed to the *Put* method of the WebSphere Enterprise Edition MQSeries Application Adapter (MQAA). The MQAA ensures that the message is placed on the specified queue.

The following is an example of Java code to send a message using this interface:

```
// copy for TMOOutbound
TextMessageCopy.TMOOutboundCopy outCopy = TextMessageCopy.TMOOutboundCopyHelper._create() ;

// set destination queueName
outCopy.queueName( destQueue ) ;

// set message data
outCopy.messageData( msgByteArray ) ;

// set encoding
outCopy.encoding( MQC.MQENC_INTEGER_NORMAL | MQC.MQENC_DECIMAL_NORMAL |
MQC.MQENC_FLOAT_IEEE_NORMAL ) ;

// set format to be a simple character string message
outCopy.format(MQC.MQFMT_STRING) ;

// convert the copy helper to a byte array
theCopyOrKeyStr = outCopy._toString() ;

// put the message to the output queue.
getOutboundMsgHome().put(theCopyOrKeyStr) ;
```

**Figure 1: Example of Code Using TMOutboundCopyHelper**

When receiving messages, the **TMInbound** interface contains the characteristics of the message as specified by the sender of the message. The receiver may require some of these attributes in order to respond to a request contained in the message or to confirm the receipt of the message.

The following is an example of code using the **TMInbound** message interface.

```
// string version of message text
String msgString = null ;
// inbound message object
TextMessage.TMInbound inMsg = null ;

// byte array for key or copy helper use
byte theCopyOrKeyStr[] = null ;

// key for TMInbound
TextMessageKey.TMInboundKey inKey =
TextMessageKey.TMInboundKeyHelper._create() ;
// set queue name to access for message
inKey.queueName(queueToGetFrom) ;

// get the Key in byte[]
theCopyOrKeyStr = inKey._toString() ;

// get the ExtendedInboundMessageQueue
IExtendedMessagingModule.ExtendedInboundMessageQueue inHome =
IExtendedMessagingModule.ExtendedInboundMessageQueueHelper.narrow(getInboundMsgHome()) ;

try {

    // begin transaction
    currentTransaction.begin();

    // get the message
    obj = inHome.getUsingKeyString(theCopyOrKeyStr) ;
    // No message in the queue
    currentTransaction.commit(true) ;
} // end catch IMessageNotFound
catch (Exception exc) {
    // commit the transaction
    currentTransaction.commit(true) ;
    // No message in the queue
    currentTransaction.commit(true) ;
} // end catch IMessageNotFound
catch (Exception exc) {
    // commit the transaction
    currentTransaction.commit(true) ;
} // end catch Exception
```

Figure 2: Example of Code Using the TMInbound Message Interface

**Table 2: TMMessage**

<b>TMMessages</b>		
TMOOutbound		
replyToQ	• N/A	The string that specifies the name of the queue, which the recipient of the message is to send the requested reply. If there is no expected response to the message, this field remains blank.
replyToQMgr	• N/A	The string that specifies the name of the queue manager where the <i>reply</i> queue exists. If this field remains blank, the queue manager shall fill it in before placing the message on the specified queue. The name used shall be the same as the queue manager the sending application is using.
queueName	• N/A	The string that specifies the name of the queue to which the message is to be put.
messageData	• N/A	The bytestring that contains the message itself. This is the sent data.
disposition	• N/A	The long that contains the disposition option, which can be specified as part of the report options of the MQMD structure. It can be set to either <b>MQRO_DEAD_LETTER_Q</b> (the default) or <b>MQRO_DISCARD_MSG</b> . <b>Further information on this attribute can be obtained from the Description of the MQMD structure in the MQSeries Application Programming Reference. (*MQSAPR)<sup>1</sup></b>
msgType	• N/A	The long that contains the message type field of the MQMD structure. <b>*MQSAPR</b>
expiry	• N/A	The long that specifies the length of time, which the message be allowed to remain on the recipients' queue. Once this time-period has expired the message shall be erased or moved to the <b>deadletter Queue</b> .
encoding	• N/A	The long that identifies the representation used for numeric values in the application message data; this applies to binary integer data, packed-decimal integer data, and floating-point data.
codedCharSetId	• N/A	The long that specifies the <i>coded character set identifier</i> of character data in the application message data.
format	• N/A	The string that specifies the format of the message data. Two examples of the field content are <b>MQC.MQFMT_STRING</b> for <i>string</i> type messages and <b>MQFMT_RF_HEADER</b> for <i>published</i> messages.
priority	• N/A	The long that specifies the priority of the message.
persistence	• N/A	The long that specifies the persistence of the message. <b>*MQSAPR</b>

<sup>1</sup> Further information on this attribute can be obtained from the Description of the MQMD structure in the *MQSeries Application Programming Reference. (\*MQSAPR)*

<b>TMMessages</b>		
msgId	• N/A	The string that is used to distinguish one message from another. No two messages <i>should</i> have the same message identifier, although this is allowed by the queue manager. The message identifier is a permanent property of the message, and shall remain after restarts of the queue manager.
correlId	• N/A	The <b>StringCorrelationID</b> . This field correlates a response to a particular request. This field set to the value contained in the incoming request, from the <b>TMInbound</b> message value.
putApplType	• N/A	The <b>long</b> that is the type of application, which <b>put</b> the message. This is part of the <b>origin context</b> of the message. <b>*MQSAPG</b>
putApplName	• N/A	The string that is part of the <b>origin context</b> of the message. The format of the name depends on the <b>PutApplType</b> . <b>*MQSAPG</b>
putDate	• N/A	The string that is the date when the message was <b>put</b> . This is part of the <b>origin context</b> of the message. <b>*MQSAPG</b>
putTime	• N/A	The string that is the time when the message was <b>put</b> . This is part of the <b>origin context</b> of the message. <b>*MQSAPG</b>
applOriginData	• N/A	The string that is <i>Application data</i> relating to origin. This is part of the <b>origin context</b> of the message. <b>*MQSAPG</b>
<b>TMInbound</b>		
replyToQ	• N/A	The string that specifies the name of the queue where the <i>response</i> message is placed. This field remains empty unless the incoming message is a request or the sender expects confirmation of receipt.
replyToQMgr	• N/A	The string that specifies the name of the queue manager where the <i>reply</i> queue is hosted.
queueName	• N/A	The string that specifies the name of the queue to which the message is to be read.
messageData	• N/A	The <b>bytestring</b> that contains the message data, which was retrieved from the MQSeries queue.
msgType	• N/A	The <b>long</b> that contains the message type field of the MQMD structure. <b>*MQSAPR</b>
expiry	• N/A	The <b>long</b> that specifies the length of time, which the message shall be allowed to remain on the recipients' queue. Once this time -period has expired the message shall be erased or moved to the <b>deadletter</b> queue.
encoding	• N/A	The <b>long</b> that identifies the representation used for numeric values in the application message data;

TMMessages		
		this applies to binary integer data, packed-decimal integer data, and floating-point data.
codedCharsetId	• N/A	The <b>long</b> that specifies the coded character set identifier of character data in the application message data.
priority	• N/A	The <b>long</b> that specifies the priority of the message.
persistence	• N/A	The <b>long</b> that specifies the persistence of the message. <b>*MQSAPR</b>
putApplType	• N/A	The <b>long</b> that is the type of application, which <b>put</b> the message. This is part of the <b>origin context</b> of the message. <b>*MQSAPG</b>
msgId	• N/A	The string that is used to distinguish one message from another. No two messages <i>should</i> have the same message identifier, although this is allowed by the queue manager. The message identifier is a permanent property of the message, and shall remain after restarts of the queue manager.
correlId	• N/A	The string that is the <i>correlation ID</i> . This field correlates a response to a particular request.
qMgrName	• N/A	The string that specifies the name of the queue manager to which the message shall be put.
putApplName	• N/A	The string that is part of the <b>origin context</b> of the message. The format of the name depends on the <b>PutApplType</b> . <b>*MQSAPG</b>
putDate	• N/A	The string that is the date when the message was put. This is part of the <b>origin context</b> of the message. <b>*MQSAPG</b>
putTime	• N/A	The string that is the time when the message was put. This is part of the <b>origin context</b> of the message. <b>*MQSAPG</b>
applOriginData	• N/A	The string that is Application data relating to origin. This is part of the <b>origin context</b> of the message. <b>*MQSAPG</b> .
waitInterval	• N/A	The <b>long</b> that specifies how long to wait for an incoming message before timing out.
backoutCount	• N/A	The <b>long</b> attribute that specifies the number of times, which a message was retrieved from the queue from within a transaction that was rolled back. This value determines the number of attempted retries on any given message.
feedback	• N/A	The <b>long</b> that is a feedback or reason code. This works with a message type <b>MQMT_REPORT</b> to indicate the nature of the report, and is only meaningful with that type of message. <b>*MQSAPR</b>
userIdentity	• N/A	The string that is part of the <b>identity context</b> of

TMMessages		
		the message; it identifies the user who originated the message. The queue manager treats this information as character data, but does not define the format of it. <b>*MQSAPR</b>
accountingToken	• N/A	The string that is part of the <b>identity context</b> of the message; it allows an application to cause work done as a result of the message to be appropriately charged.
applIdentityData	• N/A	The string that is application data relating to identity. This is part of the <b>identity context</b> of the message; it is information that the application suite defines. It provides additional information about the message or its originator.
format	• N/A	The string that specifies the format of the incoming message data. <b>MQC.MQFMT_STRING</b> for "string-type" messages and <b>MQFMT_RF_HEADER</b> for published messages are examples of the contents of this field.
stripRFHeader	• None	A method that returns a <b>bytestring</b> , which is the message data without the <b>publish/subscribe</b> header attached. If the application does not need the information contained in the <b>publish/subscribe</b> header, this method is used. The message shall be treated like a <i>string</i> type message.

## 2.2 com.lmfs.framework.amiHelpers.\*

The **OAMAS API** is the specification for the messaging interface between the business components to which the framework adheres. It provides separation of *application business functions* from the *infrastructure support*. The Integration Framework AMI helper and template classes encapsulate the Vendor's *OAMAS API* implementation. These are API wrappers on top of the Message-Oriented Middleware Application Messaging Interface. These *helper* classes constitute the interface, which application developer's use for applications meant to execute outside the application server. This makes the task of the application developer simpler and it removes the specifics of the product from the interface.

The AMI helper classes maintain an internal state that includes connections with one or more queue managers. This internal state is initialized with the **create()** call and connection(s) to MQSeries queue manager is initialized with the **open()** call. The **close()** call disconnects from the queue manager and releases any resources that were allocated by **open()** or **create()**. AMI uses policies to control many aspects of the connection to MQSeries and the processing of **puts** and **gets** to queues. Please see the AMI documentation for a thorough description of policies and their use.

There are five AMI helper classes. These classes support the various messaging styles for the sender and receiver roles. Messaging systems support multiple styles of communication between applications that include *send and forget*, *request/reply*, and *publish/subscribe*. To obtain a complete *OAMAS API* specification and messaging style descriptions, visit the **Open Applications Group** website at: <http://www.openapplications.org/oamas/loadform.htm>.

An application using the *send and forget* style sends a message to another application or list of applications but does not expect any reply. An example is a *Data Replication Application*.

Applications using *request/reply* are like client server applications where a client is making a request from a server and expecting a reply.

### **Publish/Subscribe**

*Publish/subscribe* is similar to *send and forget* except that the sending application does not know who the recipients are. Instead, the sender sends the message to a broker who manages the subscriptions of applications that requested to *receive* messages. The broker decides which subscribing applications should receive a published message by matching the subscriptions with either the message topic information or the content of the message.

### **SendAndForget.java**

This class simplifies the sending of messages using the *send and forget* messaging style.

The methods of this class are called in the following order:

```
create(...)  
open()  
send(...)  
  
.  
  
.  
send(...)  
close()
```

**Figure 3: Calling Order of SendAndForget.java**

Example code from the **Filewrapper** component:

```
private static com.lmfs.framework.amiHelpers.SendAndForget sender;  
    sender = new com.lmfs.framework.amiHelpers.SendAndForget();  
    sender.create("GCSS.IF.IFSTC3FILEWRAPPER.SESSION",  
"GCSS.IF.DEFAULT.POLICY", args[5],  
        "GCSS.IF.IFSTC3FILEWRAPPER.SEND.MESSAGE");  
    sender.open();  
    sender.setRetryCount(3);  
    sender.send(theMessage);  
    (sender.close() not performed because file wrapper loops forever and reuses sender.)
```

**Figure 4: Example Code from the Filewrapper Component for SendAndForget.java**

### **RequestResponse.java**

This class simplifies the sending and receipt of messages using the *request/reply* messaging style.

The methods of this class are called in one of the following order:

```
create(...)  
open()  
receiveRequest()  
sendReply(...)  
  
.  
  
.  
  
receiveRequest()  
sendReply(...)  
close()  
  
Or  
  
create(...)  
open()  
sendRequest()  
receiveReply(...)  
  
.  
  
sendRequest()  
receiveReply(...)  
close()
```

**Figure 5: Calling Order of RequestResponse.java**

Example code excerpted from the **Filewrapper** component:

```
private static com.lmfs.framework.amiHelpers.RequestResponse sendAndReceiver;  
    sendAndReceiver = new com.lmfs.framework.amiHelpers.RequestResponse();  
    sendAndReceiver.create("GCSS.IF.IFSTC3FILEWRAPPER.SESSION",  
"GCSS.IF.DEFAULT.POLICY",  
        args[5], args[6],  
"GCSS.IF.IFSTC3FILEWRAPPER.SEND.MESSAGE",  
        "GCSS.IF.IFSTC3FILEWRAPPER.RECEIVE.MESSAGE");  
    sendAndReceiver.open();  
    sendAndReceiver.setRetryCount(3);  
    // send the request  
    sendAndReceiver.sendRequest(theMessage);  
    ...  
    sendAndReceiver.setWaitTime(60000);  
    theReply = sendAndReceiver.receiveReply();  
(sendAndReceiver.close() not performed because file wrapper loops forever and reuses  
sendAndReceiver.)
```

**Figure 6: Example Code Excerpted from the Filewrapper Component for RequestResponse.java**

### Receiver.java

This class simplifies the receipt of messages. The **Receiver.java**, along with the *send and forget* class, can be used to implement the *send and forget* messaging style or *the request/reply* messaging style. The *request/response* class is also provided.

The methods of this class are called in the following order:

```
private static com.lmfs.framework.amiHelpers.RequestResponse sendAndReceiver;
    sendAndReceiver = new com.lmfs.framework.amiHelpers.RequestResponse();
    sendAndReceiver.create("GCSS.IF.IFSTC3FILEWRAPPER SESSION",
"GCSS.IF.DEFAULT.POLICY",
                args[5], args[6],
"GCSS.IF.IFSTC3FILEWRAPPER SEND.MESSAGE",
                "GCSS.IF.IFSTC3FILEWRAPPER RECEIVE.MESSAGE");
    sendAndReceiver.open();
    sendAndReceiver.setRetryCount(3);
// send the request
    sendAndReceiver.sendRequest(theMessage);
...
    sendAndReceiver.setWaitTime(60000);
    theReply = sendAndReceiver.receiveReply();
(sendAndReceiver.close() not performed because file wrapper loops forever and reuses
sendAndReceiver.)
```

**Figure 7: Calling Order of the Reciever.java**

Example code:

```
Receiver r = new Receiver();
    r.create("AMT.SAMPLE.SESSION", "AMT.SAMPLE.POLICY",
"AMT.SAMPLE.RECEIVER",
                "AMT.SAMPLE.MESSAGE.NAME");
r.open();
String msg = r.receive();
System.out.println("received message: " + msg);
r.close();
```

**Figure 8: Example Code of Reciever.java**

### Publisher.java

This class simplifies the publication of string format messages on a given topic. **Publisher.java**, along with the *subscribe* class, implements the *publish/subscribe* messaging style.

This class creates the objects needed to *publish* a message, opens them for use, publishes a message, and receives the confirm message from the broker.

The methods are called in this order:

```
create(...)  
open()  
publish(...)  
. . .  
publish(...)  
close()
```

**Figure 9: Calling Order for Publish.java**

Example code:

```
Publisher p = new Publisher();  
p.create("AMT.SAMPLE.SESSION", "AMT.SAMPLE.POLICY",  
"AMT.SAMPLE.PUBLISHER",  
        "AMT.SAMPLE.RECEIVER", "AMT.SAMPLE.MESSAGE1.NAME",  
        "AMT.SAMPLE.MESSAGE1.NAME");  
p.open();  
p.publish("topic we are publishing on", "message we are publishing");  
p.close();
```

**Figure 10: Example Code for Publish.java**

### **Subscriber.java**

This class simplifies the management of subscriptions and the receipt of messages that are published to topics that have been subscribed to. **Subscriber.java** along with the *publish* class, can be used to implement the *publish/subscribe* messaging style.

The methods of this class are called in the following order:

```
create(...)  
open()  
subscribe(...)  
receive()  
. . .  
receive()  
unsubscribe(...)  
close()
```

**Figure 11: Calling Order for Subscribe.java**

Example code:

```

Subscriber s = new Subscriber();
s.create("AMT.SAMPLE.SESSION", "AMT.SAMPLE.POLICY",
"AMT.SAMPLE.SUBSCRIBER",
        "AMT.SAMPLE.SENDMESSAGE.NAME", "AMT.SAMPLE.GETMESSAGE.NAME");
s.open();

s.subscribe("topic we are subscribing to");
String msg = s.receive();
System.out.println("received message: " + msg);
s.close();

```

**Figure 12: Example Code for Subscribe.java**

**Table 3: com.lmfs.framework.amiHelpers.\***

<b>Com.lmfs.framework.amiHelpers.*</b>	
<b>Class AMIHelperObject</b>	
Public void setRetryCount(int value)	
• value - (int) Specifies the number of retries attempted for a given action. (publish, send, receive, etc.)	This method allows the number of retries for sending and receiving messages to be set to a given value.
Public int getRetryCount()	
• Returns int - The current value of the retry count.	This method allows an application to retrieve the current value of the retry count.
Public void enableWarnings(boolean arg)	
• arg - (boolean) <b>true</b> enable AMI warning exceptions; <b>false</b> disables them.	Allows and application to enable ( <b>arg=true</b> ) or disable ( <b>arg=false</b> ) AMI warning exceptions.
Public void clearErrorCodes() throws AmErrorException, AmWarningException	
• Throws AmErrorException –A generic MQSeries AMI error exception.	Clears error codes in the AMI policy object.
• Throws AmWarningException –A generic MQSeries AMI warning error exception.	
Public AmStatus getLastErrorHandler() throws AmErrorException, AmWarningException	
• Returns AmStatus – The status of the last error condition from the AMI policy object.	Returns the <b>AmStatus</b> of the last error condition from the AMI policy object.
• Throws AmErrorException – A generic MQSeries AMI error exception.	
• Throws AmWarningException –A generic MQSeries AMI warning error exception.	
Public void create(String sessionName, String policyName) throws AmErrorException, AmWarningException	
• SessionName - (String) The caller-selected name given to the session.	Common code for the <b>create()</b> operation: This method creates the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AMI helper object. This method is called by the <b>create()</b> methods of all the AMI helper classes.
• PolicyName - (String) The name of the AMI policy to use.	
• Throws AmErrorException –A generic MQSeries AMI error exception.	
• Throws AmWarningException –A generic MQSeries AMI warning error exception.	

Com.lmfs.framework.amiHelpers.*	
Public void open() throws AmWarningException, AmErrorException, AMIHelperExceptionQueueManagerNotRunning, AMIHelperExceptionQueueDoesNotExist	Common code for the <b>open()</b> operation: Open session with retries. This method initializes the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AMI helper object. This method is called by the <b>open()</b> methods of all the AMI helper classes.
<ul style="list-style-type: none"> <li>Throws AmErrorException – A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException – A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning – An exception that indicates the queue manager may not be running.</li> <li>Throws AMIHelperExceptionQueueDoesNotExist – An exception that indicates the target queue may not exist.</li> </ul>	
Public void close() throws AmErrorException, AmWarningException	Common code for close operation: Close session. This method closes the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AMI helper object. This method is called by the <b>close()</b> methods of all the AMI helper classes.
Public void setWaitTime(int waitTime) throws AmErrorException, AmWarningException	<i>Setter</i> for wait time in the policy of this application.
<ul style="list-style-type: none"> <li>WaitTime - (int) new wait time (in milliseconds).</li> <li>Throws AmErrorException – A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException – A generic MQSeries AMI warning error exception.</li> </ul>	
Public int getWaitTime() throws AmErrorException, AmWarningException	<i>Getter</i> for wait time from the policy for this application.
<ul style="list-style-type: none"> <li>Returns int – The current value of the wait time from the Policy in use.</li> <li>Throws AmErrorException – A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException – A generic MQSeries AMI warning error exception.</li> </ul>	
Public void beginTransaction() throws AmErrorException, AmWarningException	This is the method you would call to begin a MQSeries unit of work, or “transaction.” You would also need to check the “Syncpoint” box in the “General” tab of the policy you plan to use. You would check the box inside the AMI Administration tool.
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	
Public void commitTransaction() throws AmErrorException, AmWarningException	This is the method you would call to commit a MQSeries unit of work, or “transaction.” You would also need to check the “Syncpoint” box in the “General” tab of the policy you plan to use. You would check the box inside the AMI Administration tool.
Public void rollbackTransaction() throws AmErrorException, AmWarningException	This is the method you would call to rollback a MQSeries unit of work, or “transaction.” You would also need to
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception</li> </ul>	

<b>Com.lmfs.framework.amiHelpers.*</b>	
<ul style="list-style-type: none"> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	check the “Syncpoint” box in the “General” tab of the policy you plan to use. You would check the box inside the AMI Administration tool.
<b>Class Publisher</b>	
	Public void create(String sessionName, String policyName, String publisherName, String responceName, String messageName, String respName) throws AmErrorException, AmWarningException
<ul style="list-style-type: none"> <li>SessionName - (String) AMI session name chosen by the application developer.</li> <li>PolicyName - (String) AMI policy name defined through the AMI administration console.</li> <li>PublisherName - (String) AMI publisher service point name defined through the AMI administration console.</li> <li>ResponseName - (String) AMI receiver service point name for the <i>confirm message</i> which the broker shall send to the application after a <i>publish()</i>.</li> <li>MessageName - (String) AMI message name for the message to publish This is a unique name chosen by the application.</li> <li>RespName - (String) AMI message name for the <i>confirm message</i> the application shall receive from the broker.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	Creates all of the internal objects of the AMI helper class needed to <i>publish</i> a message and receive the subsequent <i>confirm message</i> from the broker.
Public void open() throws AMIHelperExceptionQueueManagerNotRunning, AmErrorException, AmWarningException, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning –An exception that indicates the queue manager may not be running.</li> <li>Throws AMIHelperExceptionQueueuDoesNotExist –An exception that indicates the target queue may not exist.</li> </ul>	Opens all of the objects needed. This method initializes the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AMI helper object.
Public void publish(String topic, String message) throws AmErrorException, AmWarningException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueIsPutInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> <li>Topic - (String) The topic on which to publish the message.</li> <li>Message - (String) The message to publish.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws</li> </ul>	Publishes the contents of a Java string using the specified topic. Also receives and discards the confirmation message from the broker.

<b>Com.lmfs.framework.amiHelpers.*</b>	
<p>AMIHelperExceptionNoMessageOnQueue –  <i>Indicates the no confirm message could be found on the receiver service point.</i></p> <ul style="list-style-type: none"> <li>• Throws AMIHelperExceptionQueueIsGetInhibited –  <i>Indicates that the underlying queue for the confirm message is get inhibited.</i></li> <li>• Throws AMIHelperExceptionQueueIsPutInhibited –  <i>Indicates that the underlying queue for the publisher is put inhibited.</i></li> <li>• Throws AMIHelperExceptionQueueManagerNotRunning –  <i>An exception that indicates the queue manager may not be running.</i></li> </ul>	
public void close() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> <li>• Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>• Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	Closes all of the AMI MQSeries objects used internally by this class
<b>class Receiver</b>	
public void create(String sessionName, String policyName, String servicePointName, String messageName) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> <li>• SessionName - (String) AMI session name.</li> <li>• PolicyName - (String) AMI policy name.</li> <li>• ServicePointName - (String) AMI receiver service point name.</li> <li>• MessageName - (String) AMI message name for incoming message.</li> <li>• Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>• Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	Creates all of the objects needed to send a <b>message0</b> .
public void open() throws AmErrorException, AmWarningException, AMIHelperExceptionQueueDoesNotExist, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> <li>• Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>• Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>• Throws AMIHelperExceptionQueueDoesNotExist –  <i>An exception that indicates the queue manager may not be running.</i></li> <li>• Throws AMIHelperExceptionQueueManagerNotRunning –  <i>Exception that indicates the target queue may not exist.</i></li> </ul>	Opens all of the objects needed. This method initializes the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AMI helper object.
public String receive() throws AmErrorException, AmWarningException, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> <li>• Returns String – String containing text of received message.</li> </ul>	Receives a message and returns it in string form to the caller

<b>Com.lmfs.framework.amiHelpers.*</b>	
<p>received message.</p> <ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException – A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the incoming message is <i>get</i> inhibited.</li> <li>Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver</i> service point.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning – An exception that indicates the queue manager may not be running.</li> </ul>	
<p>public String browse() throws AmErrorException, AmWarningException,          AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited,          AMIHelperExceptionQueueManagerNotRunning</p>	
<ul style="list-style-type: none"> <li>Returns String – String containing text of browsed message.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver</i> service point.</li> <li>Throws AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the incoming message is <i>get</i> inhibited.</li> <li>Throws AMIHelperExceptionQueueManagerIsNotRunning – An exception that indicates the queue manager may not be running.</li> </ul>	Browses a message and returns it in string form to the caller. Unlike <b>receive</b> , the message remains in the underlying message queue.
<p>public void close() throws AmErrorException, AmWarningException</p>	
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	Closes all of the AMI MQSeries objects used by this class
<b>class RequestResponse</b>	
<p>public void create(String sessionName, String policyName, String senderName, String receiverName, String sendMessageName, String receiveMessageName) throws AmWarningException, AmErrorException</p>	
<ul style="list-style-type: none"> <li>SessionName - (String) AMI session name.</li> <li>PolicyName - (String) AMI policy name.</li> <li>SenderId - (String) AMI sender service point name.</li> <li>ReceiverName - (String) AMI receiver service point name. This is where the response is</li> </ul>	Creates all of the objects needed to <b>send and receive</b> requests and replies

<b>Com.lmfs.framework.amiHelpers.*</b>	
<ul style="list-style-type: none"> <li>expected to arrive.</li> <li>SendMessageName - (String) AMI message name of outgoing message.</li> <li>ReceiveMessageName - (String) AMI message name of incoming message.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	
public void open() throws AmWarningException, AmErrorException, AMIHelperExceptionQueueManagerNotRunning, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - Exception that indicates the queue manager may not be running.</li> <li>Throws AMIHelperExceptionQueueDoesNotExist - An exception that indicates the target queue may not exist.</li> </ul>	Opens all of the objects needed. This method initializes the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AMI helper object.
public void openAsResponder() throws AmWarningException, AmErrorException, AMIHelperExceptionQueueManagerNotRunning, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> <li>Throws AMIHelperExceptionQueueDoesNotExist - An exception that indicates the target queue may not exist.</li> </ul>	Opens all of the objects needed to act as a responder. This method initializes the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AMI helper object when acting as a responder.
public String receiveRequest() throws AmWarningException, AmErrorException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> <li>Returns String – String containing request message text.</li> <li>Throws AmErrorAn exception - A generic MQSeries AMI error An exception.</li> <li>Throws AmWarningAn exception - A generic MQSeries AMI warning error An exception.</li> <li>Throws AMIHelperAn exceptionNoMesageOnQueue – Indicates <i>no message</i> on the queue underlying receiver service point.</li> <li>Throws AMIHelperAn exceptionQueueIsGetInhibited – Indicates that</li> </ul>	Receive an incoming request message

<b>Com.lmfs.framework.amiHelpers.*</b>	
<p>the queue underlying receiver service point is <i>get</i> inhibited.</p> <ul style="list-style-type: none"> <li>Throws AMIHelperAn exceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> </ul>	
<pre>public void sendReply(String reply) throws AmWarningException, AmErrorException, AMIHelperExceptionQueueIsPutInhibited, AMIHelperExceptionQueueManagerNotRunning</pre>	
<ul style="list-style-type: none"> <li>reply - (String) The string containing the <i>reply</i> message text.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueIsPutInhibited – Indicates that the underlying queue for the sender is <i>put</i> inhibited.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> </ul>	Sends a <i>reply</i> to the previous request
<pre>public void sendRequest(String request) throws AmWarningException, AmErrorException, AMIHelperExceptionQueueIsPutInhibited, AMIHelperExceptionQueueManagerNotRunning</pre>	
<ul style="list-style-type: none"> <li>Request - (String) The string containing the message text to be sent.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueIsPutInhibited – Indicates that the underlying queue for the sender is <i>put</i> inhibited.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> </ul>	Sends a request to which a <i>reply</i> is expected
<pre>public String receiveReply() Throws AmWarningException, AmErrorException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueManagerNotRunning</pre>	
<ul style="list-style-type: none"> <li>Returns String – String containing the contents of the <i>reply</i> message.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver service point</i>.</li> </ul>	Receive the <i>reply</i> to the previous request

<b>Com.lmfs.framework.amiHelpers.*</b>	
<ul style="list-style-type: none"> <li>Throws – AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the receive message is get inhibited.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> </ul>	
<pre>public String receiveUncorrelatedReply() throws AmWarningException, AmErrorException,  AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited,  AMIHelperExceptionQueueManagerNotRunning</pre>	
<ul style="list-style-type: none"> <li>Returns String – String containing the contents of the received message.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException – A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver service point</i>.</li> <li>Throw AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the receive message is <i>get</i> inhibited.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> </ul>	Receives the next message on the <i>receive</i> queue, is not correlated with any previous request
<pre>public void close() throws AmWarningException, AmErrorException</pre>	
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	Closes all of the AMI MQSeries objects used internally by this class.
<pre>public void handleErrors(AmErrorException errorEx) throws AmWarningException, AmErrorException</pre>	
<ul style="list-style-type: none"> <li>eErrorEx - (AmErrorException) The exception to be handled.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	Process errors encountered during an MQSeries <b>send or receive</b> operation
<b>class SendAndForget</b>	
<pre>public void create(String sessionName, String policyName, String senderName, String messageName) throws  AmErrorException, AmWarningException</pre>	
<ul style="list-style-type: none"> <li>SessionName - (String) AMI session name.</li> <li>PolicyName - (String) AMI policy name.</li> <li>SenderId - (String) AMI sender service point name.</li> <li>MessageName - (String) AMI message name for outgoing message.</li> <li>Throws AmErrorException - A generic</li> </ul>	Creates all of the internal state objects of this AMI helper needed for sending messages.

<b>Com.lmfs.framework.amiHelpers.*</b>	
<ul style="list-style-type: none"> <li>MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	
public void open() throws AmErrorException, AmWarningException, AMIHelperExceptionQueueManagerNotRunning, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> <li>Throws AMIHelperExceptionQueueDoesNotExist - An exception that indicates the target queue may not exist.</li> </ul>	Opens all of the objects needed. This method initializes the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AMI helper object.
public void send(String message) throws AmErrorException, AmWarningException, AMIHelperExceptionQueueIsPutInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> <li>message - (String) String containing message text to be sent.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueIsPutInhibited – Indicates that the underlying queue for the sender is <i>put</i> inhibited.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> </ul>	Sends a datagram built from the <b>Message String Input Parameter</b>
public void close() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	Closes all of the AMI MQSeries objects used by this class
<b>class Subscriber</b>	
public void create(String sessionName, String policyName, String subscriberName, String sendMsgName, String receiveMsgName) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> <li>SessionName - (String) AMI session name.</li> <li>PolicyName - (String) AMI policy name.</li> <li>SubscriberName - (String) AMI subscriber service point name.</li> <li>sendMsgName - (String) AMI message name for the published message.</li> <li>ReceiveMsgName - (String) AMI message name for incoming message.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> </ul>	Creates all of the internal objects of the AMI helper class needed to <b>subscribe</b> to a topic and <i>receive</i> a published message.

<b>Com.lmfs.framework.amiHelpers.*</b>	
<ul style="list-style-type: none"> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	
public void open() throws AmErrorException, AmWarningException, AMIHelperExceptionQueueDoesNotExist, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionQueueDoesNotExist – An exception that indicates the target queue may not exist.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> </ul>	Opens all of the objects needed. This method initializes the <b>AmSession</b> and <b>AmPolicy</b> objects that are part of the internal state of the AI helper object.
public void subscribe(String topic) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> <li>topic - (String) The <i>subscribe</i> topic.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	<i>Subscribes</i> to a topic
public String receive() throws AmErrorException, AmWarningException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> <li>Returns String – String containing text of received message.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> <li>Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver</i> service point.</li> <li>Throws AMIHelperExceptionQueueIsGetInhibited- Indicates that the underlying queue for the incoming message is <i>get</i> inhibited.</li> <li>Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running.</li> </ul>	<i>Receives</i> a published message and returns it in string form to the caller
public void unsubscribe(String topic) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> <li>topic - (String) The topic from which to <i>unsubscribe</i>.</li> <li>Throws AmErrorException - A generic MQSeries AMI error exception.</li> <li>Throws AmWarningException - A generic MQSeries AMI warning error exception.</li> </ul>	Removes the subscription to a given topic
Public void close() throws AmErrorException, AmWarningException	

Com.lmfs.framework.amiHelpers.*	
<ul style="list-style-type: none"><li>• Throws AmErrorException - A generic MQSeries AMI error exception.</li></ul> <p>Throws AmWarningException - A generic MQSeries AMI warning error exception.</p>	Closes all of the AMI MQSeries objects used by this class

## 2.3 com.lmfs.framework.BOD.BOD (OAG BOD Base Class)

The Open Application Group's (OAG) **Business Object Document (BOD)** is the architecture used to communicate messages or business documents between software applications or components. Each BOD includes supporting details to enable the destination *Business application* to accomplish the action.

The BOD consists of two areas as shown in [Figure 13](#)[Figure 13](#).

- The Control Area
- The Business Data Area.

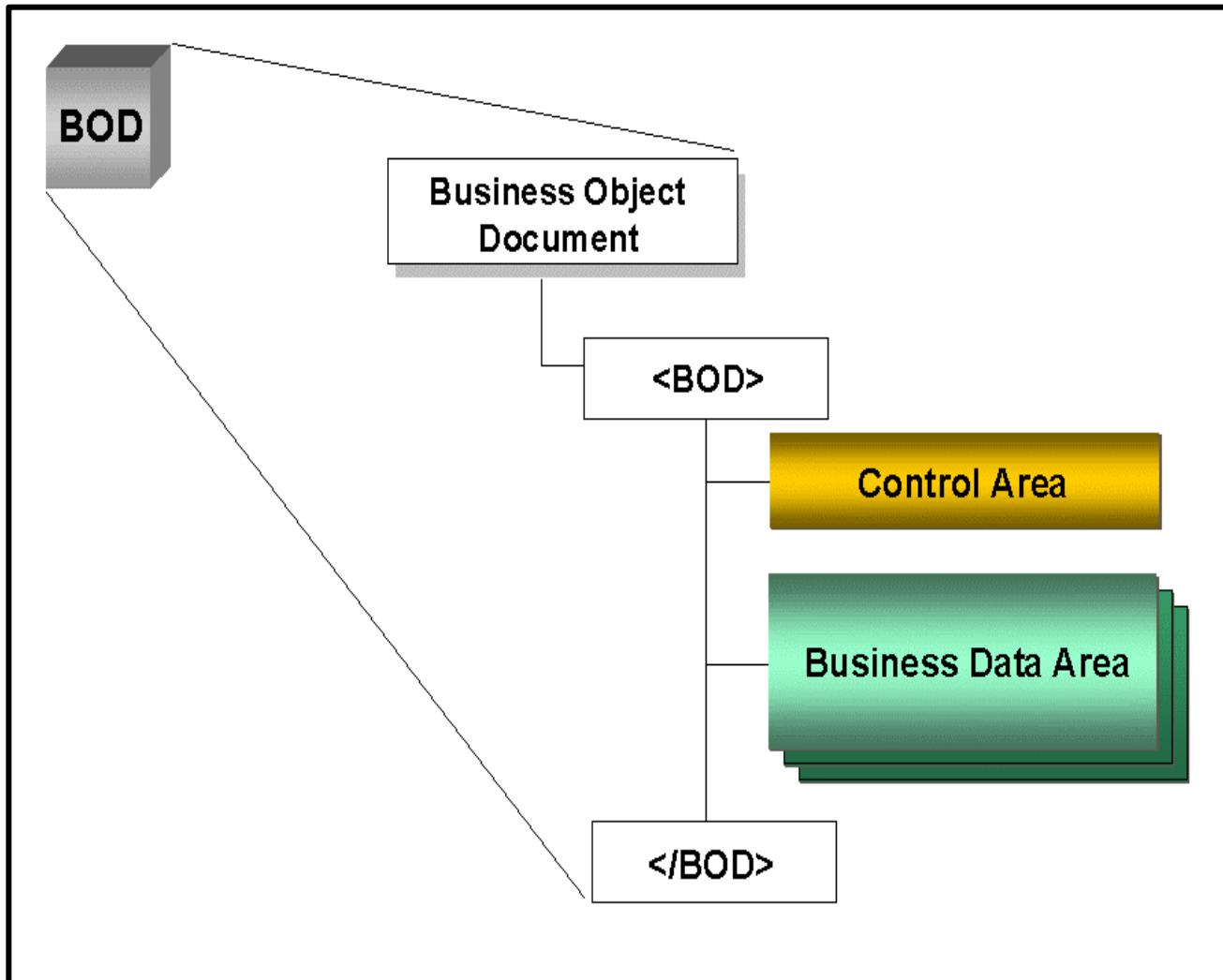


Figure 13: Business Object Document (BOD) Architecture

Provided with the IF is the BOD base class (**BOD.java**) as well as the application specific BODs which are extensions of that class used by the various provided test components. Several extended classes to the basic BOD class are included as examples for application developer use.

The base BOD class consists of the methods used to create, parse, and access the basic elements of the XML document. The *getter* and *setter* methods for elements in the *Control Area* of every BOD are in this class. The extension classes contain the *getter* and *setter* methods for each element in each of the other specific BODs.

The following example of an application uses the base BOD class to determine how to handle an incoming unsolicited message:

```
try { // BOD Handling
    // Reset the flag for confirm BOD - if generic BOD parser fails, do not send a response
    iBODConfirm = "" ;

    // Reset the flag for BOD Description
    iBODDescr = "" ;

    // Create a generic BOD
    BOD theBOD = new BOD(msgString);

    if ( logCat.isDebugEnabled() )

        logCat.debug("PDCSessionAO- " + location() + "::messageReady::Generic BOD parsed
successfully") ;

        // save the Description of the BOD
        iBODDescr = theBOD.getVERB() + theBOD.getNOUN() + theBOD.getREVISION() ;

        if ( (iBODDescr.equals("SYNCINVENTORY003")) ) { // SyncInventoryBOD
            if ( logCat.isDebugEnabled() )
                logCat.debug("PDCSessionAO- " + location() + "::messageReady::SyncInventoryBOD
received") ;

            // save the Control Area of the BOD
            iBODCtrlArea = theBOD.getCNTROLAREA() ;

            // save the Confirmation Flag
            iBODConfirm = theBOD.getCONFIRMATION() ;

            // SyncInventory BOD from EPD Wrapper to Enterprise PDC
            // parse and process the message content - invoke handleSyncInventoryBOD
            replyMsgString = handleSyncInventoryBOD(msgString) ;

            // send the reply if requested - ConfirmBOD
            if ( (iBODConfirm.equals("1")) || (iBODConfirm.equals("2")) ) {

                sendMessage(inMsg.replyToQ(),
                           inMsg.correlator(),
                           replyMsgString.getBytes(),
                           MQC.MQFMT_STRING ) ;

            }
        }
    }
}
```

```
        } // end if
    } // end if SyncInventoryBOD

    if ( logCat.isDebugEnabled() )
        logCat.debug("PDCSessionAO-" + location() + "::messageReady::Current
transaction committed") ;

} // end try - BOD handling
catch (BODEException be) {
    if ( iErrorMessage.equals("") )
        iErrorMessage = "Cannot parse the generic BOD, bad file passed" ;
    // Log Error
    logCat.error("PDCSessionAO-" + location() + "::messageReady::" + iErrorMessage,
be) ;
    throw be ;
} // end catch BODEException
```

**Figure 14: An Example of an Application using the Base BOD Class to Determine How to Handle an Incoming Unsolicited Message:**

The following is an example of the BOD class using application provided extensions:

```
private java.lang.String handleSyncInventoryBOD( java.lang.String theSyncInvBOD)
throws com.ibm.IManagedClient.IDuplicateKey,
PDCHelperModule.PDCEception
{
// <GeneratedMethodBody>
// <Body origin="user" xmi.uuid="DCE:96E70DA4-6808-11d4-8B62-000629059EDD:1">
// Insert Method modifications here
/**
 * This method shall handle a SyncInventoryBOD. It shall parse all the
 * data areas out of the BOD and process them.
 * @return java.lang.String
 * @param java.lang.String theSyncInvBOD
 * @exception com.ibm.IManagedClient.IDuplicateKey
 * @exception PDCHelperModule.PDCEception
 */

if ( logCat.isDebugEnabled() )
logCat.debug("Entering PDCSessionAO- " + location() + "::handleSyncInventoryBOD") ;

// local variables
//      return message string - Confirm BOD
String confirmStr = null ;
//      reference to pdc
PDCModule.PartsDataCollection thePDC = null ;

try {
// get the reference to PDC
thePDC = findPDCByPrimaryKey(1) ;
}
catch (com.ibm.IManagedClient.INoObjectWKey nowk) {
logCat.error("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::Cannot find
reference to PDC!!", nowk) ;
}

try { // Parse and process BOD
// create the SyncInventoryBOD from the string
SyncInventoryBOD theBOD = new SyncInventoryBOD(theSyncInvBOD) ;
if ( logCat.isDebugEnabled() )
logCat.debug("PDCSessionAO- " + location() +
)::handleSyncInventoryBOD::SyncInventoryBOD parsed successfully") ;

// save the Control Area of the BOD
iBODCtrlArea = theBOD.getCNTROLAREA() ;
// save the Confirmation Flag
iBODConfirm = theBOD.getCONFIRMATION() ;

// process the BOD
// get the number of data areas
int numOfDataAreas = theBOD.getNumOfDATAAREA() ;

for(int i = 0; i < numOfDataAreas; i+=1) {
// move to the current Data Area and initialize
```

```
theBOD.moveToDATAAREA(i) ;
theBOD.initDataAreapointers() ;
NDC.push("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Current Data Area = " +
theBOD.getCurrentLocationOfDATAAREA() );

// create a PartRecord
PDCHelperModule.PartRecord pRecord = new PDCHelperModule.PartRecord() ;

// get the values
// stock Number
pRecord.stockNumber = theBOD.getITEM() ;
if ( logCat.isDebugEnabled() )
logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Stock Number is: " +
pRecord.stockNumber) ;

// part Number
pRecord.partNumber = theBOD.getPARTNUM() ;
// item Description
pRecord.itemDescription = theBOD.getDESCRIPTN() ;
// manufacturer
pRecord.manufacturer = theBOD.getMANUFACTR() ;
// quantity
String itmQtyNumOfDec = theBOD.getITEMQTYNUMOFDEC() ;
String itmQtySign = theBOD.getITEMQTYSIGN() ;
String itmQtyUOM = theBOD.getITEMQTYUOM() ;

String itmQtyVal = null ;
int itmQty = 0 ;

if ( (itmQtyNumOfDec.equals("0")) && (itmQtySign.equals("+")) && ( (itmQtyUOM.equals("EACH"))
|| (itmQtyUOM.equals(" ")) ) ) {
    itmQtyVal = theBOD.getITEMQTYVALUE() ;
    itmQty = new Integer(itmQtyVal).intValue() ;
}
else if ( (itmQtyNumOfDec.equals("0")) && (itmQtySign.equals("+")) &&
(itmQtyUOM.equals("DOZEN")) ) {
    itmQtyVal = theBOD.getITEMQTYVALUE() ;
    itmQty = new Integer(itmQtyVal).intValue() ;
    itmQty *= 12 ;
}
pRecord.quantity = itmQty ;

// security Label
//pRecord.secLabel = theBOD.getSECLABEL();

// determine what to do based on the action code.
if (theBOD.getSYNCIND().equals("A")) {
    if ( logCat.isDebugEnabled() )
        logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Add a part from
SyncInvBOD") ;
    try {
        thePDC.addPart( pRecord ) ;
        iStatusLvl = "00" ;
        iReasonCd = "Message: SyncInventoryBOD-Add was processed successfully" ;
    }
}
```

```
if ( logCat.isDebugEnabled() )
    logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::" + iReasonCd) ;
}
catch (com.ibm.IManagedClient.IDuplicateKey idk) {
// Set the Error Message
iErrorMessage = "Duplicate Key, Part with StockNumber: " + pRecord.stockNumber + " already
exists!!";
iStatusLvl = "99";
iReasonCd = iErrorMessage;
logCat.warn("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::" + iErrorMessage, idk)
;
throw idk;
} // end catch idk - A
} // end if "A"
else if(theBOD.getSYNCIND().equals("C")) {
if ( logCat.isDebugEnabled() )
logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Update a part from
SyncInvBOD");
try {
thePDC.updatePart( pRecord );
iStatusLvl = "00";
iReasonCd = "Message: SyncInventoryBOD-Update was processed successfully";
if ( logCat.isDebugEnabled() )
logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::" + iReasonCd);
}
catch (com.ibm.IManagedClient.INoObjectWKey nk) {
// Set the Error Message
iErrorMessage = "Updating Part with StockNumber: " + pRecord.stockNumber + " does not exist!!"
;
iStatusLvl = "99";
iReasonCd = iErrorMessage;
logCat.warn("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::" + iErrorMessage, nk)
;
} // end catch nk - C
} // end else if "C"
else if(theBOD.getSYNCIND().equals("D")) {
if ( logCat.isDebugEnabled() )
logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Remove a part from
SyncInvBOD");
try {
thePDC.removePart( pRecord.stockNumber );
iStatusLvl = "00";
iReasonCd = "Message: SyncInventoryBOD-Delete was processed successfully";
if ( logCat.isDebugEnabled() )
logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::" + iReasonCd);
}
catch (com.ibm.IManagedClient.INoObjectWKey nk) {
// Set the Error Message
iErrorMessage = "Removing Part with StockNumber: " + pRecord.stockNumber + " does not
exist!!";
iStatusLvl = "99";
iReasonCd = iErrorMessage;
logCat.warn("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::" + iErrorMessage, nk)
;
} // end catch nk - D
```

```
NDC.pop();
} // end for
} // end try Parse and process BOD
catch (BODEException be) {
// Set the Error Message
iErrorMessage = "Error on parsing the SyncInventoryBOD" ;
logCat.error("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::" + iErrorMessage, be)
;
throw new PDCHelperModule.PDCEException( be.getMessage() ) ;
} // end catch be

// Build the Confirm BOD if required
try {
if ( (iBODConfirm.equals("1")) || (iBODConfirm.equals("2")) ) { // Send Confirm message
requested
// Build the Reply BOD - ConfirmBOD
ConfirmBOD confBOD = new ConfirmBOD() ;

// set ConfirmBOD's values
confBOD.setSTATUSLVL(iStatusLvl) ;
confBOD.setREASONCODE(iReasonCd) ;
confBOD.setDATAAREACNTROLAREA(iBODCtrlArea) ;
confBOD.setDESCRIPTN(iBODDescr) ;

// Get the String Version of the ConfirmBOD
confirmStr = confBOD.getBOD() ;
} // end if
}
catch (Exception exc) {
logCat.error("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Exception caught
(ConfirmBOD): ", exc) ;
throw new PDCHelperModule.PDCEException( exc.getMessage() ) ;
}

if ( logCat.isDebugEnabled() )
logCat.debug("Exiting PDCSessionAO-" + location() + "::handleSyncInventoryBOD") ;

return confirmStr ;

// End Method modifications here
// </Body>
// </GeneratedMethodBody>
}
```

Figure 15: Example of the BOD Class Using Application Provided Extensions

Refer to the *Open Applications Group Integration Specification* (OAGIS) for any information needed concerning BOD. Complete description of the structure of a BOD, the data elements that are contained in the control area, the data elements that have been defined and accepted for use with in BODs, and the allowable contents of a data element. The specification is available from the Open Applications Group's website at: <http://www.openapplications.org>.

**Table 4: com.lmfa.framework.BOD.BOD (Base Class)**

<b>com.lmfs.framework.BOD.BOD (Base Class)</b>	
Public BOD()	
• N/A	Constructor creates a new BOD object. This creates the document root, attaches a fully populated <b>CNTROLAREA</b> and a stub for the <b>DATAAREA</b>
Public BOD(String xmlBOD) throws BODException	
• XmlBOD - (String) An XML file (represented as a String) containing a BOD. • Throws BODException - When the SAX Parse fails.	Creates and parses a new BOD from an existing XML file
Public void addDATAAREA()	
• N/A	Adds another <b>DATAAREA</b> to the BOD and make the new <b>DATAAREA</b> the current <b>DATAAREA</b>
public void AddDateTimeQual(String qualifier, Element node)	
• qualifier - (String) Specifies the date time variety. For example, is this a date and time for Creation, delivery date, etc. • node - (Element) The created element in the BOD document.	Appends the <b>datetime</b> segment with the requested qualifier to the requested element, setting default time parameters. For example to add a <b>datetime</b> segment such as <b>Creation date/time</b> to a node such as <b>ReqLnHdr</b> in a BOD, you would use this method and specify the qualifier <b>Creation</b> and the node <b>ReqLnHdr</b> . The result would be the <b>Creation date/time</b> segment being appended to the <b>ReqLnHdr</b> node of the created BOD.
public void AddDateTimeQual(String qualifier, Element node, String yearSt, String monthSt, String daySt, String hourSt, String minSt, String secSt, String subsecSt, String tzSt )	
• qualifier - (String) Specifies the date time variety. For example, is this a date and time for Creation, delivery date, etc. • node - (Element) The element in the created BOD document. • yearSt - (String) The year to be used in the date field. • monthSt - (String) The month to be used in the date field. • daySt - (String) The day of the month to be used in the date field. • hourSt - (String) The hour of the day to be used in the date field. • minSt - (String) The minute within the hour to be used in the date field. • secSt - (String) The second within the minute to be used in the date field. • subsecSt - (String) The fractional part of the second to be used in the date field. • tzSt - (String) The timezone to be used in the date field.	Appends the <b>date/time</b> segment with the requested qualifier to the requested element, setting the passed in time parameters. For example, to add a <b>date/time</b> segment such as <b>Creation date/time</b> to a node such as <b>ReqLnHdr</b> in a BOD, you would use this method and specify the qualifier <b>Creation</b> , and the node <b>ReqLnHdr</b> . The result would be the <b>Creation date/time</b> segment would get appended to the <b>ReqLnHdr</b> node of the BOD being created. This method allows you to set the <b>date/time</b> fields at creation time.
public void BODToFile(String file)	
• file - (String) <i>File Name</i> of file to be written to.	Prints to a file to form an <b>XML</b> document. A ".xml" extension shall automatically be appended <b>This is a utility function only</b>
public void BODtoScreen()	
• N/A	Prints the <b>XML</b> file representing the BOD to the screen.

<b>com.lmfs.framework.BOD.BOD (Base Class)</b>	
	<b>This is a utility function only</b>
<code>public String getAUTHID()</code>	
• Returns String - The AUTHID.	Returns the <b>AUTHID</b> of the BOD
<code>public String getBOD()</code>	
• Returns String - The XML document.	Converts the <b>XML</b> document representing the BOD to a String and return the String
<code>public String getCNTROLAREA()</code>	
• Returns String - The CNTRLAREA.	Returns the entire <b>CNTRLAREA</b> of the BOD
<code>public String getCNTROLAREAdata()</code>	
• Returns String - The data contained in the CNTRLAREA.	Retrieves the data contained in the <b>CNTROLAREA</b> without any formatting or tags
<code>public String getCodepage()</code>	
• Returns String – The value of the CODEPAGE element.	Retrieves the value for the <b>CODEPAGE</b>
<code>public String getCOMPONENT()</code>	
• Returns String - The value of the COMPONENT element.	Retrieves the value for the <b>COMPONENT</b>
<code>public String getCONFIRMATION()</code>	
• Returns String - The value of the CONFIRMATION field. <ul style="list-style-type: none"><li>• 0 - Do NOT send a CONFIRM BOD</li><li>• 1 – Send a CONFIRM BOD only if there is an error.</li><li>• 2 - Send a CONFIRM BOD</li></ul>	Retrieves the <b>CONFIRMATION</b> field value
<code>public String getDAY()</code>	
• Returns String - The value of the DAY element.	Retrieves the value for <b>DAY</b>
<code>public String getHOUR()</code>	
• Returns String - The value of the HOUR element.	Retrieves the value for <b>HOUR</b>
<code>public String getLANGUAGE()</code>	
• Returns String - The value of the LANGUAGE element.	Retrieves the value for the <b>LANGUAGE</b>
<code>public String getLOGICALID()</code>	
• Returns String - The value of the LOGICALID element.	Retrieves the value for the <b>LOGICALID</b>
<code>public String getMINUTE()</code>	
• Returns String - The value of the MINUTE element.	Retrieves the value for the <b>MINUTE</b>
<code>public String getMONTH()</code>	
• Returns String – The value of the MONTH element.	Retrieves the value for the <b>MONTH</b>
<code>public String getNOUN()</code>	
• Returns String - The value of the NOUN element.	Retrieves the value for the <b>NOUN</b>
<code>public int getNumOfDATAAREA()</code>	
• Returns int – The number of DATAAREA's.	Retrieves the number of <b>DATAAREAs</b>
<code>public String getREFERENCEID()</code>	
• Returns String - The value of the REFERENCEID element.	Retrieves the value for the <b>REFERENCEID</b>

<b>com.lmfs.framework.BOD.BOD (Base Class)</b>	
public String getREVISION()	
• Returns String - The value of the REVISION element.	Retrieves the value for the <b>REVISION</b>
public String getSECOND()	
• Returns String – The value of the SECOND element.	Retrieves the value for the <b>SECOND</b>
public String getSUBSECOND()	
• Returns String – The value of the SUBSECOND element.	Retrieves the value for the <b>SUBSECOND</b>
public String getTASK()	
• Returns String - The value of the TASK element.	Retrieves the value for the <b>TASK</b>
public String getTimezone()	
• Returns String - The value of the TIMEZONE element.	Retrieves the value for the <b>TIMEZONE</b>
public String getVERB()	
• Returns String – The value of the VERB element.	Retrieves the value for the <b>VERB</b>
public String getYEAR()	
• Returns String - The value of the YEAR element.	Retrieves the value for the <b>YEAR</b>
public void moveToDATAAREA(int danum)	
• danum - (int) Which DATAAREA node to make current.	Move to the specified <b>DATAAREA</b> Note: the first one is zero. <b>SHALL BE A VALID INDEX</b>
public void moveToNextDATAAREA()	
• N/A	Moves to the next DATAAREA in the BOD
public void moveToPreviousDATAAREA()	
• N/A	Moves to the previous <b>DATAAREA</b> in the BOD
public void setAUTHID(String value)	
• Value - (String) The value of the AUTHID field.	Sets the <b>AUTHID</b> field
public void setCODEPAGE(String value)	
• value - (String) The value of the CODEPAGE field.	Sets the <b>CODEPAGE</b> field
public void setCOMPONENT(String value)	
• value - (String) The value of the COMPONENT field.	Sets the <b>COMPONENT</b> field
public void setCONFIRMATION(String value)	
• value - (String) The value of the CONFIRMATION field: <ul style="list-style-type: none"> <li>• 0 - Do NOT send a CONFIRM BOD</li> <li>• 1 - Send a CONFIRM BOD only if there is an error</li> <li>• 2 - Send a CONFIRM BOD</li> </ul>	Sets the <b>CONFIRMATION</b> field
public void setLANGUAGE(String value)	
• value - (String) The value of the LANGUAGE field.	Sets the <b>LANGUAGE</b> field
public void setLOGICALID(String value)	
• value - (String) The value of the LOGICALID field.	Sets the <b>LOGICALID</b> field

com.lmfs.framework.BOD.BOD (Base Class)	
public void setNOUN(String value)	
• value - (String) The value of the NOUN field.	Sets the <b>NOUN</b> field
public void setREFERENCEID(String value)	
• value - (String) The value of the REFERENCEID field.	Sets the <b>REFERENCEID</b> field
public void setREVISION(String value)	
• value - (String) The value of the REVISION field.	Sets the <b>REVISION</b> field
public void setTASK(String value)	
• value - (String) The value of the TASK field.	Sets the <b>TASK</b> field
public void setVERB(String value)	
• value - (String) The value of the VERB field.	Sets the <b>VERB</b> field

## 2.4 com.lmfs.framework.pubsub.MQRFH - Publish/Subscribe Helpers

The *publish/subscribe helper* classes encapsulate the MQSeries *publish* and *subscribe* message constructs and formats. When an application uses the *publish/subscribe* messaging style, the application communicates with the *publish/subscribe* broker of MQSeries to accomplish various tasks such as registering or de-registering a publisher or subscriber of information of a particular topic. The broker requires specially formatted messages for these tasks. The purpose of the *publish/subscribe helper* classes is to simplify the construction of the broker control messages. The application developer calls a method, which puts the formatted broker control message into a buffer that can then be forwarded to the broker.

An example of how the *Trigger Monitor Application* template provided with the Integration Framework uses the *publish/subscribe helper* bindings for C applications follows below:

This code reads lines from a file and looks for stanzas beginning with [**registerPublisher**], [**registerSubscriber**], [**deregisterPublisher**], and [**deregisterSubscriber**]. When it finds one of these tokens, it reads in additional input lines that are to be used as parameters when the **publish/subscribe** helper classes are to be invoked. Then the code calls the helper class, which corresponds to the stanza it is processing. The helper class formats a buffer with the message needed to instruct the broker to perform the given operation. Then the code sends the message to the broker using **putMsg()**.

Example:

```
while( !read_line(inputFile, myLine, sizeof(myLine)) ) {
    memset(msg, 0, sizeof(msg));

    if ( !strcmp(myLine, "[registerPublisher]" ) ) {

        debug( "found registerPublisher" );

        read_line(inputFile, exceptions, sizeof(exceptions));
        debug( "exceptions = ", exceptions );
        if (find_exception(exceptions)) {
            debug( "found exception ... skipping to next stanza" );
            break;
        }

        read_line(inputFile, topic, sizeof(topic));
        read_line(inputFile, stream, sizeof(stream));
        expand_loc(stream);
        read_line(inputFile, qmgr, sizeof(qmgr));
        read_line(inputFile, queue, sizeof(queue));
        expand_loc(queue);
        read_line(inputFile, broker_queue, sizeof(broker_queue));
        read_line(inputFile, broker_qmgr, sizeof(broker_qmgr));

        debug( "topic = ", topic );
        debug( "stream = ", stream );
        debug( "qmgr = ", qmgr );
        debug( "queue = ", queue );
        debug( "broker_queue = ", broker_queue );
        debug( "broker_qmgr = ", broker_qmgr );

registerPublisher(msg,
    topic, // topic to subscribe to
    stream, // stream to subscribe to
    qmgr, // qmgr where reply comes to
    queue); // queue name where reply comes to

        putMsg (broker_qmgr, broker_queue, (((PMQRFH)msg)->StrucLength), msg );

    } // if

    if ( !strcmp(myLine, "[registerSubscriber]" ) ) {
        debug( "found registerSubscriber" );

        read_line(inputFile, exceptions, sizeof(exceptions));
        debug( "exceptions = ", exceptions );
        if (find_exception(exceptions)) {
            debug( "found exception ... skipping to next stanza" );
            break;
        }
    }
}
```

```
read_line(inputFile, topic, sizeof(topic));
    read_line(inputFile, stream, sizeof(stream));
    expand_loc(stream);
    read_line(inputFile, qmgr, sizeof(qmgr));
    read_line(inputFile, queue, sizeof(queue));
    expand_loc(queue);
    read_line(inputFile, broker_queue, sizeof(broker_queue));
    read_line(inputFile, broker_qmgr, sizeof(broker_qmgr));

    debug( "topic = ", topic );
    debug( "stream = ", stream );
    debug( "qmgr = ", qmgr );
    debug( "queue = ", queue );
    debug( "broker_queue = ", broker_queue );
    debug( "broker_qmgr = ", broker_qmgr );

registerSubscriber(msg,
    topic, // topic to subscribe to
    stream, // stream to subscribe to
    qmgr, // qmgr where subscribed messages shall arrive
    queue); // queue name where subscribed messages shall

arrive

    putMsg (broker_qmgr, broker_queue, (((PMQRFH)msg)->StrucLength), msg );

} // if

if ( !strcmp(myLine, "[deregisterPublisher]" ) ) {

    debug( "found deregisterPublisher" );

    read_line(inputFile, exceptions, sizeof(exceptions));
    debug( "exceptions = ", exceptions );
    if (find_exception(exceptions)) {
        debug( "found exception ... skipping to next stanza" );
        break;
    }

    read_line(inputFile, topic, sizeof(topic));
    read_line(inputFile, stream, sizeof(stream));
    expand_loc(stream);
    read_line(inputFile, qmgr, sizeof(qmgr));
    read_line(inputFile, queue, sizeof(queue));
    expand_loc(queue);
    read_line(inputFile, broker_queue, sizeof(broker_queue));
    read_line(inputFile, broker_qmgr, sizeof(broker_qmgr));

    debug( "topic = ", topic );
    debug( "stream = ", stream );
    debug( "qmgr = ", qmgr );
    debug( "queue = ", queue );
    debug( "broker_queue = ", broker_queue );
    debug( "broker_qmgr = ", broker_qmgr );
```

```
deregisterPublisher(msg,
    topic, // topic to subscribe to
    stream, // stream to subscribe to
    qmgr, // qmgr where reply comes to
    queue); // queue name where reply comes to

    putMsg (broker_qmgr, broker_queue, (((PMQRFH)msg)->StrucLength), msg );

} // if

if ( !strcmp(myLine, "[deregisterSubscriber]" ) ) {

    debug( "found deregisterSubscriber" );

    read_line(inputFile, exceptions, sizeof(exceptions));
    debug( "exceptions = ", exceptions );
    if (find_exception(exceptions)) {
        debug( "found exception ... skipping to next stanza" );
        break;
    }

    read_line(inputFile, topic, sizeof(topic));
    read_line(inputFile, stream, sizeof(stream));
    expand_loc(stream);
    read_line(inputFile, qmgr, sizeof(qmgr));
    read_line(inputFile, queue, sizeof(queue));
    expand_loc(queue);
    read_line(inputFile, broker_queue, sizeof(broker_queue));
    read_line(inputFile, broker_qmgr, sizeof(broker_qmgr));

    debug( "topic = ", topic );
    debug( "stream = ", stream );
    debug( "qmgr = ", qmgr );
    debug( "queue = ", queue );
    debug( "broker_queue = ", broker_queue );
    debug( "broker_qmgr = ", broker_qmgr );

deregisterSubscriber(msg,
    topic, // topic to subscribe to
    stream, // stream to subscribe to
    qmgr, // qmgr where reply comes to
    queue); // queue name where reply comes to

    putMsg (broker_qmgr, broker_queue, (((PMQRFH)msg)->StrucLength), msg );

} // if

} // while
```

Figure 16: Example of Publish/Subscribe Helper Code

The following example of code demonstrates how a Java application would then use the **publish/subscribe** Java bindings to publish information to subscribers of a topic. In this example a buffer is created which contains a message to the broker instructing it to publish the data contained in **replyMsgString** using the topic **SYNCINVENTORY**. The buffer is then sent to the broker using **sendMessage()**.

```
// build the SyncInventoryBOD
String replyMsgString = buildSyncInventoryBOD( pRecord, "D" ) ;

// set the format variable

// Publish the remove to the BASEs that subscribe
// publish the change
// need String topic, String qname, String qmname,
// String puboptions, byte[] buffer
byte[] replyByteArray = mqps.publishWithData( "SYNCINVENTORY",
    "",
    "",
    "",
    replyMsgString ) ;

// set the destination queueName to be the Stream queue
// set to IF.<location>.DEFAULT.STREAM
// the broker shall put the SyncInventory BOD from Enterprise PDC (Publisher)
// to the Base 1, 2, 3 PDC's (Subscribers) whose queues are named as follows:
// PDC.<location>.SYNCINVENTORY.RECEIVER where loc = BASE1, BASE2, or
BASE3
String destQueue = "IF." + nameContext + ".DEFAULT.STREAM" ;
// set the correlator to be a empty string
String correlator = "" ;

// start a transaction
// obtain access to a transaction control object.
obj = CBSeriesGlobal.orb().resolve_initial_references("TransactionCurrent") ;
currentTransaction = org.omg.CosTransactions.CurrentHelper.narrow(obj) ;
currentTransaction.set_timeout(180) ;
if ( logCat.isDebugEnabled() )
    logCat.debug("PDCSessionAO-" + location() + "::removePart::Beginning current
transaction (Outbound)" ) ;
// begin transaction
currentTransaction.begin() ;

// Send the message
sendMessage( destQueue,
    correlator,
    replyByteArray,
    MQFMT_RF_HEADER ) ;

// end transaction
currentTransaction.commit(true) ;
```

Figure 17: Example that Shows Use of publish/subscribe Java bindings

**Table 5: PubSubHelpers (Bindings for C Applications)**

<b>Pub-Sub Helpers (Bindings for C Applications)</b>	
void buildMQRFHeader(PMQRFH pRFHeader)	
<ul style="list-style-type: none"> <li>• PRFHeader - (PMQRFH) A pointer to the region of pre-allocated memory in which the <i>MQRFHeader</i> is to be constructed.</li> </ul>	This function builds the RFH which is used for all <b>publish/subscribe</b> message
void deletePublication (void *pMsgBuffer, char *pTopic, char *pStream)	
<ul style="list-style-type: none"> <li>• pMsgBuffer - (void *) A pre -allocated buffer that returns the completed <i>name/value pair</i> string</li> <li>• pTopic - (char *) The topic of the publication to be deleted. If none is provided, all shall be deleted for that stream.</li> <li>• PStream - (char *) The topic stream for which the publication shall be deleted. If none is provided the default stream shall be used.</li> </ul>	This function builds <b>NameValueString</b> to delete a retained publication
void deregisterPublisher (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	
<ul style="list-style-type: none"> <li>• pMsgBuffer - (void *) A pre -allocated buffer which shall be used to return the completed <i>name/value pair</i> string.</li> <li>• pTopic - (char *) The topic of the publication to be deleted. A topic shall be provided.</li> <li>• pStream - (char *) The topic stream for the topics named. If none is provided the default stream is used.</li> <li>• pQMgrName - (char *) The queue manager named on the registerPublisher call. If none is provided the values from the message descriptor shall be used.</li> <li>• pQName - (char *) The queue named on the registerPublisher call. If none is provided the values from the message descriptor shall be used.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a publisher no longer publishes data for the topic indicated
void deregisterSubscriber (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	
<ul style="list-style-type: none"> <li>• pMsgBuffer - (void *) A pre -allocated buffer which shall be used to return the completed <i>name/value pair</i> string.</li> <li>• pTopic - (char *) The topic of the publication to be deleted. A topic shall be provided.</li> <li>• pStream - (char *) The topic stream for the topics named. If none is provided the default stream shall be used.</li> <li>• pQMgrName - (char *) The queue manager named on the <i>registerSubscriber</i> call. If none is provided the values from the message descriptor shall be used.</li> <li>• pQName - (char *) The queue named on the registerSubscriber call. If none is provided the values from the message descriptor shall be used.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a subscriber shall no longer receive data for the topic indicated
void publish (void *pMsgBuffer, char *pTopic, char *pQMgrName, char *pQName, char *pPublishOptions)	
<ul style="list-style-type: none"> <li>• pMsgBuffer - (void *) A pre -allocated buffer</li> </ul>	This function builds <b>NameValueString</b> to publish a

<b>Pub-Sub Helpers (Bindings for C Applications)</b>	
<p>which shall be used to return the completed <i>name/value pair</i> string.</p> <ul style="list-style-type: none"> <li>• pTopic - (char *) The topic of the publication. A topic shall be provided.</li> <li>• pQMgrName - (char *) The queue manager named on the <i>registerPublisher</i> call. If none is provided the values from the message descriptor shall be used.</li> <li>• pQName - (char *) The queue named on the <i>registerPublisher</i> call. If none is provided the values from the message descriptor shall be used.</li> <li>• pPublishOptions - (char *) MQPS_RETAIN_PUBLICATION or none.</li> </ul>	publication
void registerPublisher (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	
<ul style="list-style-type: none"> <li>• pMsgBuffer - (void *) A pre-allocated buffer which shall be used to return the completed <i>name/value pair</i> string.</li> <li>• pTopic - (char *) The topic of the publication to be published. A topic shall be provided.</li> <li>• pStream - (char *) The topic stream for the topics named. If none is provided the default stream shall be used.</li> <li>• pQMgrName - (char *) The queue manager to which confirmation messages are to be sent. If none is provided the values from the message descriptor shall be used.</li> <li>• pQName - (char *) The queue name to which confirmation messages are to be sent. If none is provided the values from the message descriptor shall be used.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a publisher shall publish data for the topic indicated
void registerSubscriber (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	
<ul style="list-style-type: none"> <li>• pMsgBuffer - (void *) A pre-allocated buffer which shall be used to return the completed <i>name/value pair</i> string.</li> <li>• pTopic - (char *) The topic of the publication to be deleted. A topic shall be provided.</li> <li>• pStream - (char *) The topic stream for the topics named. If none is provided the default stream shall be used.</li> <li>• pQMgrName - (char *) Name of queue manager to which messages shall be sent. If none is provided the values from the message descriptor shall be used.</li> <li>• pQName - (char *) Name of queue to which messages shall be sent. If none is provided the values from the message descriptor shall be used.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a subscriber shall receive data for the topic indicated
void requestUpdate (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	
<ul style="list-style-type: none"> <li>• pMsgBuffer - (void *) A pre-allocated buffer which shall be used to return the completed <i>name/value pair</i> string.</li> </ul>	This function builds <b>NameValueString</b> to request retained publications for the topic indicated

<b>Pub-Sub Helpers (Bindings for C Applications)</b>	
<ul style="list-style-type: none"> <li>• pTopic - (char *) The topic of the publications requested.</li> <li>• pStream - (char *) The topic stream for the topics named. If none is provided the default stream shall be used.</li> <li>• pQMgrName - (char *) The queue manager to which confirmation messages are to be sent. If none is provided the values from the message descriptor shall be used.</li> <li>• pQName - (char *) The queue name to which confirmation messages are to be sent. If none is provided the values from the message descriptor shall be used.</li> </ul>	
<b>com.lmfs.framework.pubsub.MQRFH - Pub-Sub Helpers (Bindings for Java Applications)</b>	
public void reset() throws Exception	
<ul style="list-style-type: none"> <li>• Throws Exception-Any exceptions thrown by the underlying string classes are re-thrown.</li> </ul>	This method sets all RFH attributes to default values.
public void readIn(MQMessage msg) throws Exception	
<ul style="list-style-type: none"> <li>• msg - (MQMessage) The message object to parse.</li> <li>• Throws Exception-Any exceptions thrown by the underlying string classes are re-thrown.</li> </ul>	Parses the RFH and <b>NameValueString</b> from an RFH format MQSeries message object
public void padNameValuePair() throws Exception	
<ul style="list-style-type: none"> <li>• Throws Exception-Any exceptions thrown by the underlying string classes are re-thrown.</li> </ul>	Adds white space to ensure full <i>Word Boundary Alignment</i> .
public void addNameValuePair (String name, String value) throws Exception	
<ul style="list-style-type: none"> <li>• name - (String) Name to append.</li> <li>• value - (String) Value to append.</li> <li>• Throws Exception-Any exceptions thrown by the underlying string classes are re-thrown.</li> </ul>	Concatenates the name and value strings passed to the global <b>NameValueString</b> .
public void writeOut(MQMessage msg) throws Exception	
<ul style="list-style-type: none"> <li>• msg - (MQMessage) The message object to build.</li> <li>• Throws Exception- Any exceptions thrown by the underlying string classes are re-thrown.</li> </ul>	Builds a <b>MQMessage</b> data buffer using the RFH header and global <b>NameValueString</b> .
public int registerPublisher(String topic, String stream, String qname, String qmname, byte[] buffer ) throws Exception	
<ul style="list-style-type: none"> <li>• topic - (String) The topic to register.</li> <li>• stream - (String) The stream to register.</li> <li>• qname - (String) The queue to which confirmations are to be sent.</li> <li>• qmname - (String) The queue manager. to which confirmations are to be sent.</li> <li>• buffer - (byte[]) The buffer in which the <i>namevalue</i> string is stored.</li> <li>• Returns int – The length of the <i>publish/subscribe</i> header.</li> <li>• Throws Exception- Any exceptions thrown by the underlying string classes are re-thrown. An exception is thrown if the topic is not supplied.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a publisher shall publish data for the topic indicated

<b>Pub-Sub Helpers (Bindings for C Applications)</b>	
public void registerPublisher(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) The topic to register</li> <li>stream - (String) The stream to register</li> <li>qname - (String) The queue to which confirmations are to be sent.</li> <li>qmname - (String) The queue manager to which confirmations are to be sent.</li> <li>msg - (MQMessage) The message object in which the name value string is stored.</li> <li>Throws Exception- Any exceptions thrown by the underlying <i>string</i> classes are re-thrown. An exception is thrown if the topic is not supplied.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a publisher shall publish data for the topic indicated
Public int deletePublication (String topic, String stream, byte[] buffer) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) Topic of publication to delete.</li> <li>stream - (String) Stream of publication to delete.</li> <li>buffer - (byte[]) The buffer in which the name value string is stored.</li> <li>Returns int – The length of the publish/subscribe header.</li> <li>Throws Exception- - Any exceptions thrown by the underlying <i>string</i> classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to delete a retained publication
Public void deletePublication (String topic, String stream, MQMessage msg) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) Topic of publication to delete.</li> <li>stream - (String) Stream of publication to delete.</li> <li>msg - (MQMessage) The message object in which the name value string is stored.</li> <li>Throws Exception- - Any exceptions thrown by the underlying string or MQSeries message classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to delete a retained publication
public int deregisterPublisher(String topic, String stream, String qname, String qmname, byte[] buffer ) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) Topic to deregister.</li> <li>stream - (String) Stream to deregister.</li> <li>qname - (String) Name of queue where confirmations are to be sent.</li> <li>qmname - (String) Name of queue manager where confirmations are to be sent.</li> <li>Buffer - (byte[]) The buffer in which the name value string is stored.</li> <li>Returns int – The length of the publish/subscribe header.</li> <li>Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a publisher shall no longer publish data for the topic indicated
public void deregisterPublisher(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) Topic to deregister.</li> <li>stream - (String) Stream to deregister.</li> <li>qname - (String) Name of queue where confirmations are to be sent.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a publisher shall no longer publish data for the topic indicated

<b>Pub-Sub Helpers (Bindings for C Applications)</b>	
<ul style="list-style-type: none"> <li>• qmname - (String) Name of queue manager where confirmations are to be sent.</li> <li>• Msg - (MQMessage) The message object in which the <i>name value</i> string is stored.</li> <li>• Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown.</li> </ul>	
<pre>public int deregisterSubscriber(String topic, String stream, String qname, String qmname, byte[] buffer ) throws Exception</pre>	
<ul style="list-style-type: none"> <li>• topic - (String) Topic to deregister.</li> <li>• stream - (String) Stream to deregister.</li> <li>• qname - (String) Name of queue where publications are sent.</li> <li>• qmname - (String) Name of queue manager where publications are sent.</li> <li>• buffer - (byte[]) The buffer in which the name value string is stored.</li> <li>• Returns int - The length of the publish/subscribe header.</li> <li>• Throws Exception- Any exceptions thrown by the underlying string or MQSeries message classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a subscriber shall no longer receive data for the topic indicated
<pre>public void deregisterSubscriber(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception</pre>	
<ul style="list-style-type: none"> <li>• topic - (String) Topic to deregister.</li> <li>• stream - (String) Stream to deregister.</li> <li>• qname - (String) Name of queue where publications are sent.</li> <li>• qmname - (String) Name of queue manager where publications are sent.</li> <li>• msg - (MQMessage) The message object in which the <i>name value</i> string is stored.</li> <li>• Throws Exception- Any exceptions thrown by the underlying string classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a subscriber shall no longer receive data for the topic indicated
<pre>public int publish(String topic, String qname, String qmname, String puboptions, byte[] buffer ) throws Exception</pre>	
<ul style="list-style-type: none"> <li>• topic - (String) the topic of the publication. A topic shall be provided.</li> <li>• qname - (String) the queue named on the registerPublisher call.</li> <li>• qmname - (String) the queue manager named on the registerPublisher call.</li> <li>• puboptions - (String) MQPS_RETAIN_PUBLICATION or none.</li> <li>• buffer - (byte[]) The buffer in which the <i>name value</i> string is stored.</li> <li>• Returns int - The length of the publish/subscribe header</li> <li>• Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to publish a publication

<b>Pub-Sub Helpers (Bindings for C Applications)</b>	
public byte [] publishWithData( String topic, String qname, String qmname, String puboptions, String pubData ) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) the topic of the publication. A topic shall be provided.</li> <li>qname - (String) the queue named on the <i>registerPublisher</i> call.</li> <li>qmname - (String) the queue manager named on the <i>registerPublisher</i> call.</li> <li>Puboptions - (String) MQPS_RETAIN_PUBLICATION or none.</li> <li>pubData -(String) The message data to <i>publish</i>.</li> <li>Returns byte[] - The buffer in which the <i>name value</i> string is stored.</li> <li>Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to publish a publication including the publication data
public void publish(String topic, String qname, String qmname, String puboptions, MQMessage msg) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) the topic of the publication. A topic shall be provided.</li> <li>qname - (String) the queue named on the <i>registerPublisher</i> call.</li> <li>qmname - (String) the queue manager named on the <i>registerPublisher</i> call.</li> <li>puboptions - (String) MQPS_RETAIN_PUBLICATION or none.</li> <li>msg - (MQMessage) The message object in which the <i>name value</i> string is stored.</li> <li>Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to publish a publication
public int registerSubscriber(String topic, String stream, String qname, String qmname, byte[] buffer ) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) The topic being <i>subscribed to</i>.</li> <li>stream – (String) the stream being <i>subscribed to</i>.</li> <li>qname - (String) the name of the queue where messages are to arrive.</li> <li>qmname - (String) the name of the queue manager where messages are to arrive.</li> <li>Buffer - (byte[]) The buffer in which the <i>name value</i> string is stored.</li> <li>Returns int – The length of the publish/subscribe header.</li> <li>Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown.</li> </ul>	This function builds <b>NameValueString</b> to indicate that a subscriber shall receive data for the topic indicated
public void registerSubscriber(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception	
<ul style="list-style-type: none"> <li>topic - (String) The <i>subscribed</i> topic.</li> <li>stream - (String) the stream being <i>subscribed to</i>.</li> <li>qname - (String) the name of the queue where</li> </ul>	This function builds <b>NameValueString</b> to indicate that a subscriber shall receive data for the topic indicated

<b>Pub-Sub Helpers (Bindings for C Applications)</b>	
<p>messages are to arrive.</p> <ul style="list-style-type: none"> <li>• qmname - (String) the name of the queue manager where messages are to arrive.</li> <li>• msg - (MQMessage) The message object in which the name value string is stored.</li> <li>• Throws Exception- Any exceptions thrown by the underlying string or MQSeries message classes are re-thrown.</li> </ul>	
<pre>public int requestUpdate(String topic, String stream, String qname, String qmname, byte[] buffer ) throws Exception</pre>	<p>This function builds <b>NameValueString</b> to request retained publications for the topic indicated</p>
<ul style="list-style-type: none"> <li>• topic - (String) The topic to receive updates.</li> <li>• Stream - (String) The stream to receive updates.</li> <li>• qname - (String) the name of the queue where the updates are to arrive.</li> <li>• qmname - (String) the name of the queue manager where the updates are to arrive.</li> <li>• Buffer - (byte[]) The buffer in which the <i>name value</i> string is stored.</li> <li>• Returns int – The length of the publish/subscribe header.</li> <li>• Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown.</li> </ul>	
<pre>public void requestUpdate(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception</pre>	<p>This function builds <b>NameValueString</b> to request retained publications for the topic indicated</p>
<ul style="list-style-type: none"> <li>• topic - (String) The topic to get updates.</li> <li>• stream - (String) ) The stream to get updates for.</li> <li>• qname - (String) the name of the queue where the updates are to arrive.</li> <li>• qmname - (String) the name of the queue manager where the updates are to arrive.</li> <li>• msg - (MQMessage) The message object in which the <i>name value</i> string is stored.</li> <li>• Throws Exception-A re-thrown exception thrown by the underlying string or MQSeries message classes.</li> </ul>	

## 2.5 com.lmfs.framework.servlet.IFServlet

In the **Servlet Structure** (hierarchy) shown in 2, the **PDCServlet** is the base class for the PDC Application. **IFServlet** is the Framework *Base* class which all IF Servlet Applications should extend. The purpose of the **IFServlet** is to allow common function (methods) and framework integration as well as security across all applications. It allows abstract methods to be implemented down the hierarchy. An abstract class is one that has to be implanted by the class, which extends from it.

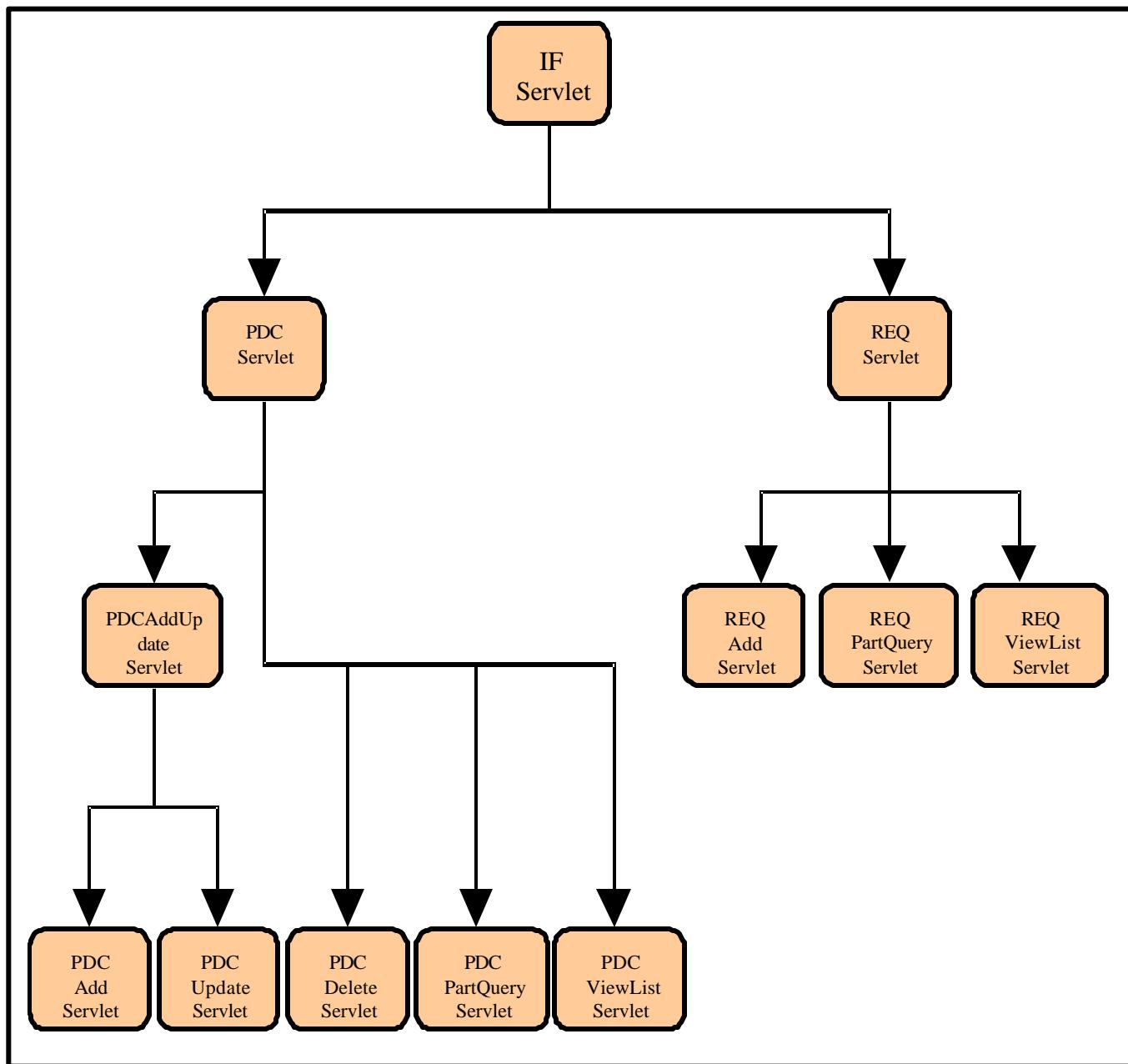


Figure 18 IFServlet Extended Class Diagram

**Table 6: com.lmfs.framework.servlet.IFServlet**

<b>com.lmfs.framework.servlet.IFServlet</b>	
public void doGet(HttpServletRequest req, HttpServletResponse res)	
<ul style="list-style-type: none"> <li>Req - (HttpServletRequest) Object that encapsulates the request to the Servlet.</li> <li>Res - (HttpServletResponse) Object that encapsulates the response from the Servlet.</li> </ul>	Process incoming <b>HTTP GET</b> requests specific to all <b>IFServlets</b>
Public abstract void doGetService(HttpServletRequest request, HttpServletResponse response) throws Exception	
<ul style="list-style-type: none"> <li>Request - (HttpServletRequest) Object that encapsulates the request to the Servlet.</li> <li>Response - (HttpServletResponse) Object that encapsulates the response from the Servlet.</li> </ul>	An abstract method, that Servlets derived from this base shall implement, since it is called in the above <b>doGet</b> method. Its purpose is to handle the incoming get requests specific to an application.
public void doPost(HttpServletRequest req, HttpServletResponse res)	
<ul style="list-style-type: none"> <li>req - (HttpServletRequest) Object that encapsulates the request to the Servlet.</li> <li>res - (HttpServletResponse) Object that encapsulates the response from the Servlet.</li> </ul>	Process incoming <b>HTTP POST</b> requests
public abstract void doPostService(HttpServletRequest request, HttpServletResponse response) throws Exception	
<ul style="list-style-type: none"> <li>request - (HttpServletRequest) Object that encapsulates the request to the Servlet.</li> <li>Response - (HttpServletResponse) Object that encapsulates the response from the Servlet.</li> </ul>	An abstract method, that Servlets derived from this base shall implement, since it is called in the above <b>doPost</b> method. Its purpose is to handle the incoming post requests specific to an application.
Protected HttpSession getHttpSession(HttpServletRequest request)	
<ul style="list-style-type: none"> <li>Request - (HttpServletRequest) Object that encapsulates the request to the Servlet.</li> <li>Returns HttpSession - The session information extracted from the request.</li> </ul>	Creates a new session.
public java.lang.String getParameter(HttpServletRequest request, String parameterName, boolean checkRequestParameters, boolean checkInitParameters, boolean isParameterRequired, String defaultValue) throws Exception	
<ul style="list-style-type: none"> <li>Request - (HttpServletRequest) Object that encapsulates the request to the Servlet.</li> <li>ParameterName - (String) The name of the parameter value to return.</li> <li>CheckRequestParameters - (boolean) When <b>true</b> the request parameters <i>are</i> searched . When <b>false</b> the request parameters <i>are not</i> searched.</li> <li>CheckInitParameters - (boolean) When <b>true</b> the Servlet init parameters <i>are</i> searched. When <b>false</b> init parameters <i>are not</i> searched.</li> <li>IsParameterRequired - (boolean) When <b>true</b> an exception is thrown when the parameter cannot be found. If <b>false</b> no exception shall be thrown if parameter is not found.</li> <li>DefaultValue - (String) The default value to return when the parameter is not found.</li> <li>Returns java.lang.String - The value of the specified parameter.</li> </ul>	Returns the value of the requested parameter identified by the parameter name specified.
protected SessionInfoStruct getSessionInfoStruct(HttpServletRequest request, String location) throws Exception	
<ul style="list-style-type: none"> <li>request - (HttpServletRequest) Object that encapsulates the request to the Servlet.</li> <li>Location - based on the user making the request.</li> </ul>	Creates a security related <b>SessionInfoStruct</b> from <b>EXPECTED</b> parameters taken from the Servlet request. The <b>EXPECTED</b> Servlet request parameters are: <b>LDAP</b>

<b>com.lmfs.framework.servlet.IFServlet</b>	
<ul style="list-style-type: none"> <li>• <b>BaseNameQualifier</b> – based on the user making the request</li> <li>• Returns <b>SessionInfoStruct</b> - The completed <i>SessionInfoStruct</i> that is required for method invocations to Component Broker. This is used to check access privileges of the user making the request.</li> </ul>	<b>Groups and PAC Credentials.</b> The menu system or JSPs need to supply the "location" and <b>BaseNameQualifier</b> . The Framework currently retrieves this info from LDAP in the Menu URLs as parameters. A users menu is filtered by security checks as to what he has access to, this is how location and <b>BaseNameQualifier</b> is related to a specific user.
<b>public java.lang.Object getValueFromSession(HttpServletRequest request, String key)</b>	
<ul style="list-style-type: none"> <li>• <b>request</b> - (<b>HttpServletRequest</b>) Object that encapsulates the request to the Servlet</li> <li>• <b>key</b> - (String) The identifier of the value to be retrieved from the HTTP session.</li> </ul>	Retrieves an object, specified by the parameter <i>key</i> , from the session that is contained in the request.
<b>public void init(ServletConfig config) throws javax.servlet.ServletException</b>	
<ul style="list-style-type: none"> <li>• <b>config</b> - (<b>ServletConfig</b>) The <i>GenericServlet</i> interface is passed to the super class.</li> </ul>	Contains the common code required to initialize the Servlets. The logging service is initialized. The connection to the Component Broker environment is established. If security between the Servlet engine and the Component Broker environment is turned on, the <b>ServletLogonHelper</b> is invoked to perform the login into Component Brokers' DCE cell. Typically, this method shall not be overridden by the client Servlet.
<b>protected abstract void initializePropertyManager()</b>	
<ul style="list-style-type: none"> <li>• N/A</li> </ul>	An abstract method, Servlets derived from this base case shall provide the implementation. The purpose is to insure each application gets client properties based on unique file names. The <b>IFFilePropertyManager</b> class is used for getting properties (this class wrappers the <b>PropertyResourceBundle</b> ) which are used to configure the application at runtime.
<b>protected abstract void initializeServerConnection()</b>	
<ul style="list-style-type: none"> <li>• N/A</li> </ul>	<p>An abstract method, Servlets derived from this base case shall provide the implementation. Its purpose is to allow the deriving class to implement a specific server connection policy.</p> <p>This abstract method is the same for both CORBA and EJB components that require the initialization of the CBSeriesGlobal Object for security.</p> <p>(i.e. CORBA &amp; EJB: CBSeriesGlobal.Initialize(args, ORBProperties))</p>
<b>public void putValueOnSession(HttpServletRequest request, String key, Object object)</b>	
<ul style="list-style-type: none"> <li>• <b>request</b> - (<b>HttpServletRequest</b>) Object that encapsulates the request to the Servlet.</li> <li>• <b>key</b> - (String) The value that is used to identify the object being placed on the session object.</li> <li>• <b>object</b> - (Object) The object to be placed on the session.</li> </ul>	Places the given object on the HTTP session, contained in the request object, using the key specified as the name for future retrieval.

## 2.6 com.lmfs.framework.\* - Servlet Utility Helpers

Table 7: com.lmfs.framework.\* - Servlet Utility Helpers

com.lmfs.framework.* - Servlet Utility Helpers	
Abstract class IFPropertyManager	
<code>public IFPropertyManager(String resourceName)</code>	
• ResourceName	Abstract Wrapper class used to get resources It is not called directly by IF developers
<code>public abstract String getString(String key)</code>	
• Key - (String) bundle key	Wrappers bundle <b>getString</b>
• Returns String	
class IFFilePropertyManager	
<code>public IFFilePropertyManager(String resourceName)</code>	
• resourceName - (String) resource name	<b>Wrapper</b> class used to get bundle resources from a file
<code>public String getString(String key)</code>	
• key - (String) bundle key	Wrappers bundle <b>getString</b>
Returns String	
class ServletLoginHelper	
<code>synchronized public boolean checkLogin(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, String userID, String password) throws LoginHelperGeneralException, LoginHelperBadCredentialsException, LoginHelperLoginFailedException, ServletException, IOException</code>	
• HttpServletRequest - (HttpServletRequest) Object that encapsulates the request to the Servlet.	Checks the session that is contained in the Servlet request to see if the user has previously logged in and that the login is still valid. If the user has not been logged in before, or the session is no longer valid, the user shall be logged in using the <i>userID</i> and <i>password</i> provided
• HttpServletResponse - (HttpServletResponse) Object that encapsulates the response from the Servlet.	
• userID - (String) The <i>userID</i> to be used to login to the Component Broker DCE cell.	
• Password - (String) The <i>password</i> to be used to login to the Component Broker DCE cell.	
• Throws LoginHelperGeneralException – Is thrown in <i>login helper</i> when errors are of general nature, passed to <i>invoking</i> class.	
• Throws LoginHelperBadCredentialsException - Is thrown in <i>login helper</i> when errors are <b>Bad Credentials</b> passed to <i>invoking</i> class.	
• Throws LoginHelperLoginFailedException – Is thrown in <i>login helper</i> when errors deal with <b>Login failure</b> , passed to invoking class.	
• Throws ServletException – Servlet related errors are passed to <i>invoking</i> class.	
• Throws IOException – IO errors are passed to <i>invoking</i> class	
<code>Protected Object doLogin(String string1, String string2) throws LoginHelperLoginFailedException</code>	
• string1 - (String) The <i>userID</i> to be used to login to the Component Broker DCE cell.	Logs the user into the Component Broker DCE cell using the <i>userID</i> and <i>password</i> provided.
• String2 - (String) The <i>Password</i> to be used to login to the Component Broker DCE cell.	
• Returns Object - The security level 2 credentials that result from a successful login.	
• Throws LoginHelperLoginFailedException – Is thrown in <i>login helper</i> when errors deal with	

com.lmfs.framework.* - Servlet Utility Helpers	
<i>Login failure</i> , passed to <i>invoking class</i> .	
public void init() throws LoginHelperGeneralException	
<ul style="list-style-type: none"><li>Throws LoginHelperGeneralException – Is thrown in <i>login helper</i> when errors are of general nature, passed to <i>invoking class</i>.</li></ul>	Sets up the connection to the Component Broker security context and creates the <i>system login helper</i> that shall ultimately be used to log users into the Component Broker DCE cell.
public static void logoff(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException	
<ul style="list-style-type: none"><li>HttpServletRequest - (HttpServletRequest) Object that encapsulates the request to the Servlet.</li><li>HttpServletResponse - (HttpServletResponse) Object that encapsulates the response from the Servlet.</li><li>Throws ServletException - Servlet related errors are passed to <i>invoking class</i>. Throws <i>IOException</i> – IO errors are passed to <i>invoking class</i>.</li></ul>	Logs the user out of the Component Broker DCE cell by removing the credentials from the session contained in the Servlet request.

### 3. Security Helpers

#### 3.1 com.lmfs.framework.LDAPHelper.\*

This package provides wrappers that assist in the interface to an *LDAP Directory Server*. The package was developed for the Menu test component to access an LDAP server. It was originally designed on and used in a Windows NT/x86 environment, and was also successfully tested on Solaris 2.7. The level of abstraction it provides greatly reduces the level of effort required when connecting to an LDAP server, but proportionally limits the flexibility available. It may or may not be used for other LDAP access.

The classes provided in this package are:

##### com.lmfs.framework.LDAPHelper.LDAPHelper

**LDAPHelper** provides abstracted access to an LDAP Server using a properties file.

##### com.lmfs.framework.LDAPHelper.LDAPResultsHelper

Translates an **LDAPSearchResults** object such that it is easier to sort/navigate.

##### com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator

A class used with the menu test component for sorting purposes.

##### com.lmfs.framework.LDAPHelper.LDAPTests

A class used for menu test component debugging purposes.

### 3.1.1 com.lmfs.framework.LDAPHelper.LDAPHelper

**LDAPHelper** is a class to facilitate connecting to and searching an *LDAP Server*. **LDAPHelper** uses a mixture of a properties file and various constructors. The properties file is read by default in the beginning of all constructors, and the constructor parameters override the values read from the properties file as described.

The example below shows initiating the LDAPHelper that obtains information about the LDAP directory it is connecting to from the GCSSRegistry.properties file. The starting point for our menu test component, and **request\_string** in this example, is '**ifmit=Main Menu,ou=Integration Framework,ou=usaf,ou=pki,ou=dod,o=U.S. Government,c=us**'. The menu system test component uses the ifmit object class for the menu items. Therefore the **pattern\_string** is set to '**ifmit=\***' in this example. The result is a set list of the distinguished names of all of the menu items found below our the starting point, that is the **request\_string**, in the LDAP directory.

```
private Vector loadMenu() {
    test();
    LDAPHelper ldh = null;
    try {
        ldh = new LDAPHelper();
    }
    catch (java.lang.Exception e) {
        menuLogger.error("Error constructing LDAP Helper " + "object in loadmenu():\n");
        e.printStackTrace();
    }
    LDAPSearchResults aSearchResults = ldh.findDN(REQUEST_STRING, PATTERN_STRING);
}
```

Figure 19: LDAPHelper Example from Menu.java file of the Menu System Test Component

Table 8: com.lmfs.framework.LDAPHelper.LDAPHelper

com.lmfs.framework.LDAPHelper.LDAPHelper	
public LDAPHelper ()	Constructor uses <GCSSRegistry.properties> file for all settings.
public LDAPHelper (int)	<ul style="list-style-type: none"> <li>In inSearchScope - The search scope to search with. Overrides corresponding GCSSRegistry.properties setting</li> </ul> Constructor uses <GCSSRegistry.properties> file for all but scope setting.
private LDAPHelper (int, java.lang.String)	<ul style="list-style-type: none"> <li>In ReferralFlag</li> <li>In inStrReferralURL - The URL of the referred-to LDAP Server.</li> </ul> This stub is a placeholder for the handling of referrals.
public LDAPHelper (java.lang.String)	<ul style="list-style-type: none"> <li>In iniFileLocation - A subdirectory in the classpath that contains the GCSSRegistry.properties file.</li> </ul> Constructor uses <GCSSRegistry.properties> file for all but the properties file location. The properties file shall still be named <GCSSRegistry.properties> but can be

com.lmfs.framework.LDAPHelper.LDAPHelper	
	placed in a different subdirectory name.  Example: <b>LDAPHelper x = new LDAPHelper("/SubDirName")</b> , where the properties file should be located in some directory in the classpath as: <b>/SubDirName/GCSSRegistry.properties</b> .
<b>public LDAPHelper (java.lang.String, int)</b>	
<ul style="list-style-type: none"> <li>• In inStrHost - The host to connect to. Overrides GCSSRegistry.properties setting.</li> <li>• In inIntPort - The port to connect to. Overrides GCSSRegistry.properties setting.</li> </ul>	Constructor uses <b>&lt;GCSSRegistry.properties&gt;</b> file for all but host and port settings.
<b>public LDAPHelper (java.lang.String, int, int)</b>	
<ul style="list-style-type: none"> <li>• In inStrHost - The host to connect to. Overrides corresponding GCSSRegistry.properties setting.</li> <li>• In inIntPort - The port to connect to. Overrides corresponding GCSSRegistry.properties setting.</li> <li>• In inIntSearchScope - The search scope to search with. Overrides corresponding GCSSRegistry.properties setting.</li> </ul>	Constructor uses <b>&lt;GCSSRegistry.properties&gt;</b> file for all but host, port, and scope settings.
<b>Private check_result ()</b>	
	This private method verifies that an LDAP operation succeeded, and logs a warning if not.
<b>private closeConnection ()</b>	
	Private function to close connection to LDAP host.
<b>private createConnection ()</b>	
	Private function to create connection to LDAP host.
<b>public finalize ()</b>	
	Finalizes private attribute <b>ldapConn_</b> .
<b>public findDN (java.lang.String, java.lang.String)</b>	
<ul style="list-style-type: none"> <li>• In inStrSearchBase - Base DN at which to begin the search.</li> <li>• In inStrFilt - The filter to search with, in format "<i>attribute=value</i>".</li> </ul>	Search for a DN in the Directory, where <b>inStrSearchBase</b> is a base DN to start from, and <b>inStrFilt</b> is in format " <b>attribute=value</b> "
<b>public getHost ()</b>	
	Returns the host attribute of <b>LDAPConnection</b> .
<b>private readINI ()</b>	
	Sets all default settings from <b>&lt;GCSSRegistry.properties&gt;</b>

### 3.1.2 com.lmfs.framework.LDAPHelper.LDAPResultsHelper

Translates a **LDAPSearchResults** object so that it is easier to sort and navigate. Methods **NextEntry** and **NextAttribute** read the next **LDAPEntry** and next attribute of the current entry from the returned **LDAPSearchResults**, respectively.

The following code example takes the list of distinguished names returned by the **com.lmfs.framework.LDAPHelper**. *LDAPHelper.findDN* method and reads all of the attributes and values from the LDAP Directory for each Distinguished Name in the list.

```
LDAPResultsHelper mh = new LDAPResultsHelper(aSearchResults);
Vector infoltems = new Vector();
int currentItemLevel = 0;
mh.nextEntry();
mh.nextAttribute();
while (mh.getDN() != null) {
    LDAPInfoItem itemInfo = new LDAPInfoItem();
    itemInfo.setDn(mh.getDN());
    while (mh.getAttributeName() != "") {
        itemInfo.addAttribute(mh.getAttributeName(), mh.getAttributeValue());
        mh.nextAttribute();
    }
    if (!itemInfo.getItemLabel().equals("Main Menu"))
        infoltems.addElement(itemInfo);
    mh.nextEntry();
    mh.nextAttribute();
}
return createMenu(authorizeMenuItems(infoltems));
```

**Figure 20: LDAPResultsHelper Code Example from the Menu.java file of the Menu System Test Component**

**Table 9: com.lmfs.framework.LDAPHelper.LDAPResultsHelper**

com.lmfs.framework.LDAPHelper.LDAPResultsHelper	
<b>public LDAPResultsHelper (netscape.ldap.LDAPSearchResults)</b>	
<ul style="list-style-type: none"> <li>In netscape.ldap. LDAPSearchResults inLDAPRes - The <i>resultset</i> to enumerate.</li> </ul>	Constructs a new <b>LDAPSearchResults</b> object accepting an <b>LDAPSearchResults</b> object as a parameter.
<b>public LDAPResultsHelper (netscape.ldap.LDAPSearchResults, netscape.ldap.LDAPEntryComparator)</b>	
<ul style="list-style-type: none"> <li>In netscape.ldap. LDAPSearchResults inLDAPRes - The <i>resultset</i> to enumerate.</li> <li>In netscape.ldap. LDAPEntryComparator inComparator - The <i>comparator</i> to use when sorting the resultset.</li> </ul>	Constructs a new <b>LDAPSearchResults</b> object accepting an <b>LDAPSearchResults</b> object and an <b>LDAPEntryComparator</b> object as parameters.
<b>public getAttributeName ()</b>	Returns the name of the currently enumerated attribute.
<b>public getAttributeValue ()</b>	Returns the value of the currently enumerated attribute.
<b>public getDN ()</b>	Returns the DN of the currently enumerated LDAP Entry.
<b>public nextAttribute ()</b>	Enumerates to next attribute in the attribute set of the current LDAP Entry.
<b>public nextEntry ()</b>	Enumerates to next <b>LDAPEntry</b> in the entries of the <b>LDAPResultSet</b> .

### 3.1.3 com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator

The **LDAPMenuEntryComparator** class is not currently used but was originally specifically developed for the Menu System Test Component.

The class is used in conjunction with **LDAPResultsHelper** to sort results obtained from a **FindDN** call. The class may be used as an example of a custom LDAP search results sorting routine. The constructors usually accept a string parameter **BaseDN** that indicates the portion of the DN that should be ignored when the returned entries are compared.

The comparison within **isGreater()** is done in the following manner:

- The **BaseDN** portion of each **LDAPEntry**'s DN is removed
- The remainder of the DNs are converted to a Policy Director namespace object name format (**reverse order, /-delimited**. Example "**cn=myName,ou=Organization1**" is converted to "**/Organization1/myName**")
- The strings are subsequently compared.
- If the first string is greater than the second, true is returned.

**Table 10: com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator**

com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator	
Public LDAPMenuEntryComparator (java.lang.String)	
• IN PDMMenuBaseDN – The Base DN to ignore when sorting search results by DN.	Use the <b>PDFormattedDN</b> to compare search results from an LDAP Search.
Public LDAPMenuEntryComparator (java.lang.String, boolean, java.lang.String)	<p>• IN PDMMenuBaseDN – The Base DN to ignore when sorting search results by DN.</p> <p>• IN useSortAttribute – Boolean to indicate that the sort attribute should be used. The value is expected to be true and is ignored. The parameter is meant to distinguish between constructors.</p> <p>• IN sortAttributeToUse – String value to indicate which attribute contains leaf node sorting values.</p> <p>Use the <b>sort</b> attribute (when available) instead of the <b>PDFormattedDN</b> to compare search results from an <b>LDAP Search</b>. If the <b>sort</b> attribute is nonexistent, the Policy Director-Formatted DN is used instead.</p> <p>The standard comparison ensures that the order of the distinguished name begins with the outermost container (<b>c=us,o=u.s. government,...,cn=Order Part</b>) versus beginning with the relative distinguished name (e.g. <b>cn=Order Part,...,c=us</b>)</p> <p>The <b>sortAttributeToUse</b> identifies that the comparison should use the value of this attribute in the object referenced by the Distinguished Names for the comparison rather than the Distinguished Name itself.</p>
Public LDAPMenuEntryComparator (boolean, java.lang.String)	
• IN useSortAttribute – Boolean to indicate that the sort attribute should be used. The value is expected to be true and is ignored. The parameter is meant to distinguish between constructors.	<p>Use the <b>sort</b> attribute (when available) instead of the Policy Director-Formatted DN to determine which one's greater.</p> <p>The standard comparison ensures that the order of the distinguished name begins with the outermost container (<b>c=us,o=u.s. government,...,cn=Order Part</b>) versus</p>

com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator	
sorting values.	beginning with the relative distinguished name (e.g. <b>cn=Order Part,...,c=us</b> )
The <b>sortAttributeToUse</b> identifies that the comparison should use the value of this attribute in the object referenced by the Distinguished Names for the comparison rather than the Distinguished Name itself.	
Public isGreater (netscape.ldap.LDAPEntry, netscape.ldap.LDAPEntry)	<ul style="list-style-type: none"> <li>• IN LDAPEntry - The first entry to compare.</li> <li>• IN LDAPEntry - The second entry to compare to the first.</li> </ul> <p>Required method for an <b>LDAPComparator</b>. Returns true if the first entry is greater than the second. <b>LDAPMenuEntryComparator</b> first uses the sort attribute if the <b>useSortAttribute flag</b> is set, followed by a Policy Director formatting of the DN.</p>

### 3.1.4 com.lmfs.framework.LDAPHelper.LDAPTests

**LDAPTests** A class used for menu test component debugging purposes only.

**LDAPTests** lists all entries with the DN prefix of "**ifmit=\***" after the hardcoded base DN: "**ifmit=Main Menu,ou=Integration Framework, ou=usaf, ou=pki, ou=dod, o=U.S.Government, c=usa.**" **LDAPTests** is meant for debugging purposes.

**Table 11: com.lmfs.framework.LDAPHelper.LDAPTests**

com.lmfs.framework.LDAPHelper.LDAPTests	
public LDAPTests ()	
public static main (java.lang.String[])	
• IN initArgs[] - unused in this test routine.	<b>main()</b> is meant as a debugging tool. It uses <b>LDAPHelper</b> and <b>LDAPResultsHelper</b> to dump out everything with " <b>ifmit=*</b> " after hardcoded base DN: " <b>ifmit=Main Menu,ou=Integration Framework, ou=usaf, ou=pki, ou=dod,o=U.S.Government, c=usa.</b> "

## 3.2 com.lmfs.framework.security.\*

This package provides classes that interface the authentication and access control portions of the Integration Framework. It was originally designed on and used in the WebSphere AE and EE products in a Windows NT/x86 environment. A stubbed version of the classes is available on the Solaris 2.7 platform for WebSphere AE and EE.

The level of abstraction it provides greatly reduces the level of effort required when interfacing with **aznAPI**, but proportionally limits the flexibility available.

This package also contains classes to export a structure from an LDAP server in Policy Director Object Name format.

The Security Package contains the following classes/packages:

**com.lmfs.framework.security.IFSecurityException**

An exception raised by **PDAuthz** subclasses.

**? com.lmfs.framework.security.IFMenuPOSCreator**

- **com.lmfs.framework.security.IFMenuPOSCreator**

A class used by the menu test components. It exports LDAP entries to a format readable by Policy Director.

**com.lmfs.framework.security.PDCheckParser**

A class that filters HTML documents by replacing HTML tags with Policy Director values. This class is untested.

**? com.lmfs.framework.security.PDCheckEnvironment**

- **com.lmfs.framework.security.PDCheckEnvironment**

A class which contains data necessary for PDCheckParser.

**? com.lmfs.framework.security.AuthnInfo**

- **com.lmfs.framework.security.AuthnInfo**

Is an interface for retrieving authentication information.

### **3.2.1 com.lmfs.framework.security.IFSecurityException**

An exception raised when a security error occurs. Stores additional exception information as needed.

**Table 12: com.lmfs.framework.security.IFSecurityException**

com.lmfs.framework.security.IFSecurityException	
public IFSecurityException ()	
public IFSecurityException (java.lang.String)	
• IN s - Additional exception information.	Constructor allowing additional information to be passed as a string parameter.

### **3.2.2 com.lmfs.framework.security.IFMenuPOSCreator**

**IFMenuPOSCreator** exports the LDAP items used by the menu system to a file readable by Policy Director. Uses the **com.lmfs.framework.LDAPHelper.\*** classes to accomplish LDAP communication. See the routine **com.lmfs.framework.util.StringExt.getPDFFormattedDN** (String, String) for information on how this is done.

This is a self contained executable created for the express purpose of generating a text file of the menu system hierarchy. The text file is then placed on the Policy Director Security Management Server so that the Menu can be viewed in the Policy Director Object Namespace and to allow Access Control Lists to be applied to it.

**Table 13: com.lmfs.framework.security.IFMenuPOSCreator**

com.lmfs.framework.security.IFMenuPOSCreator	
Public IFMenuPOSCreator ()	Default Constructor.
Public static main (java.lang.String[])	
• IN args – Command-line arguments.	Method called from command line invocation. Creates <b>PD Object Namespace</b> from a DN in LDAP.

### 3.2.3 com.lmfs.framework.security.PDCheckParser

**PDCheckParser** provides filtering of HTML documents. This class is not used by any framework or Test Component code.

The **PDCheckParser** replaces some basic HTML tags as can be seen in the following example:

<IVUSER> is replaced with the currently logged on user.  
<IVGROUPS> is replaced with the currently logged on user's group list.  
<IVPAC> is replaced with the user's stringified PAC.  
In addition fine-grained authorization semantics are supported, delimited by <IVCHECK> and </IVCHECK> tags. <IVCHECK> provides "IF" semantics while </IVCHECK> provides "ENDIF" semantics. Also available is an optional <IVFAIL> tag, which provides "ELSE" semantics.

An <IVCHECK> tag specifies that any text between <IVCHECK> and </IVCHECK> is only available to users who pass the authorization checks listed in the <IVCHECK> tag.

For users who do not pass the checks, an optional <IVFAIL> tag is available. Text between the <IVFAIL> and </IVCHECK> tags are available to the users who do not pass the authorization check.

The format is:

```
<xmp>
<IVCHECK object="objname">
```

Text that is only available to a user with access  
<IVFAIL>

Text that is only available to a user WITHOUT access  
</IVCHECK>

```
</xmp>
```

where objname is a fully qualified protected element in the PD namespace.

If the user requesting the page has read access for that object then the text within the <IVCHECK> and </IVCHECK> tags shall be returned. Otherwise, it shall be skipped and the text between the <IVFAIL> and </IVCHECK> tags shall be returned.

```
An example:  
<html><body>  
<h1>This heading can be seen by anyone</h1>  
This is a line of text that anyone can see.  
<IVCHECK object="/stuff/foo">  
Congratulations, you have permission to access /stuff/foo.  
<IVFAIL>  
Sorry, you do not have access to /stuff/foo.  
</IVCHECK>  
<P>This text can be seen by anyone  
</body></html>
```

Figure 21: Example of PDCheckParser Code

The following restrictions shall be obeyed, otherwise the **<IVCHECK>**, **<IVFAIL>** and **</IVCHECK>** tags shall not be handled correctly:

- The **<IVFAIL>** tag is optional.
- The tags shall be on a single line.
- The tags shall not span lines.
- The tags shall be the only things on the line.
- **<IVCHECK>** and **<IVFAIL>** may be in lower, upper, or mixed case.
- The keyword **<i>object</i>** on the **<IVCHECK>** tag may be in lower, upper, or mixed case.
- You should place the object name inside of double quotes.
- The **<IVCHECK>** tags may be nested. If you are inside of an **<IVCHECK>** tag that failed authorization, however, you cannot access data even if that data is inside of an **<IVCHECK>** tag that would pass (unless you are inside an **<IVFAIL>** tag).

Example:

```
<IVCHECK object="/data/that/I/cannot/access">  
I shall not see this line.  
<IVCHECK object="/data/that/I/can/access">  
Normally I would have access to this, but since I am inside of  
another tag that failed, I shall not see this.  
</IVCHECK>  
<IVFAIL>  
<IVCHECK object="/data/that/I/can/access">  
Since I am inside of an IVFAIL, I can see this.  
</IVCHECK>  
</IVCHECK>  
<IVCHECK object="/data/that/I/can/access">  
I see this line.  
</IVCHECK>
```

Figure 22: Example of <IVCheck> HTML Tag

**Table 14: com.lmfs.framework.security.PDCheckParser**

com.lmfs.framework.security.PDCheckParser	
public PDCheckParser (com.lmfs.framework.security.PDAuthz.PDCheckAuthz, java.lang.String, java.lang.String, java.lang.String, java.lang.String)	
<ul style="list-style-type: none"> <li>• In templateFilename - The template file name</li> </ul>	Constructor that records the filename of the template file.
private filterHTML (java.io.BufferedReader, java.io.BufferedWriter)	<p>The <b>filterHTML</b> command shall write the processed HTML to the output stream. It shall search for tags that are completely on a single line (that is, the "&lt;" and "&gt;" are both on the same line).</p> <p><b>*If the tag is one that we care about, then we handle it. Otherwise, we try to write it to the output stream. Note that we may be inhibiting output, so it may not actually be written out if we are inside of a &lt;IVCHECK&gt; and &lt;/IVCHECK&gt; pair that failed authorization.</b></p>
private handleTag (java.lang.String, java.io.BufferedWriter)	
<ul style="list-style-type: none"> <li>• In line - A line of HTML data</li> </ul>	<p>The <b>handleTag</b> command shall scan a line for any of our supported tags and process it. If one of those tags is found, then <b>handleTag</b> shall return true to tell its caller that we handled the tag. When an &lt;IVCHECK&gt; tag is found, the following occurs:</p> <ul style="list-style-type: none"> <li>• A table of arguments is built (keyword/value pairs)</li> <li>• An authorization check is done based on the arguments.</li> <li>• The <b>inhibitOutput</b> flag is set based on the result of the authorization check</li> </ul> <p>With the &lt;IVCHECK&gt; there are two parameter combinations:</p> <ul style="list-style-type: none"> <li>• Only <b>objname</b> is provided</li> <li>• The <b>objname</b> is not provided</li> </ul> <p>When an &lt;IVFAIL&gt; tag is found the <b>inhibitOutput</b> flag is toggled.</p> <p>When an &lt;/IVCHECK&gt; tag is found the <b>inhibitOutput</b> flag is set to its previous state</p> <p>When an &lt;IVUSER&gt; tag is found the current username is substituted for the tag</p> <p>When an &lt;IVGROUPS&gt; tag is found the current user's group list is substituted for the tag</p> <ul style="list-style-type: none"> <li>• When an &lt;IVPAC&gt; tag is found the user's PAC is substituted for the tag</li> </ul>
private parseArgString (java.lang.String)	
<ul style="list-style-type: none"> <li>• In args - A line of name/value pairs, separated by equals signs</li> </ul>	<p>The command <b>parseArgString</b> shall scan a line for name/value pairs (separated by equals signs, "=") and return a hashtable of those pairs.</p> <p>The value data may be enclosed in single or double</p>

com.lmfs.framework.security.PDCheckParser	
	quotes. In addition, the name (key) shall be uppercased before putting it into the table, to make later retrieval easier.
<b>private setupStreamTokenizer (java.lang.String)</b>	
<ul style="list-style-type: none"> <li>• In args - A line of name and value pairs</li> </ul>	<b>SetupStreamTokenizer</b> shall initialize a <b>StreamTokenizer</b> object for parsing <IVCHECK> argument strings. The basic form of an argument string is <b>name=value</b> where value may be in single or double quotes (and may contain spaces). There may be several <b>name=value</b> pairs in the string, each delimited by white space.
<b>public toStream (java.io.Writer)</b>	
<ul style="list-style-type: none"> <li>• In writer - The writer to which we shall send the processed HTML text</li> </ul>	<b>ToStream</b> is used to write the HTML data represented by this component to the specified output writer. This class acts as a filter, allowing our super class to write its HTML to a writer that we provide. It then scans the text for tags, processing them, and writing the resulting HTML to the writer provided by our caller.
<b>private writeOutput (java.io.Writer, java.lang.String)</b>	
<ul style="list-style-type: none"> <li>• In w - The writer to which output is directed</li> <li>• In s - A string to write</li> </ul>	<b>WriteOutput</b> shall write a string to the writer if we are not inhibiting output. To make the source easier to read from the browser's view source option, a new line shall be appended to the line of text.

### 3.2.4 com.lmfs.framework.security.PDCheckEnvironment

**PDCheckEnvironment** contains the data necessary to make authorization decisions through <IVCHECK> tags in an HTML document. It contains the notion of a current environment, and a stack of saved environments (to support nesting <IVCHECK> tags).

This class is used by the **PDCheckParser** class. It is not called by any Integration Framework or Test Components.

The environment contains the following data:

- Name of the object that we want to check access to
- Current state of inhibit output - i.e. are we between an <IVCHECK> and </IVCHECK> tag and if we are, did the authorization succeed or fail?
- A flag to indicate if we are currently within an <IVFAIL> block

**Table 15: com.lmfs.framework.security.PDCheckEnvironment**

com.lmfs.framework.security.PDCheckEnvironment	
public PDCheckEnvironment (com.lmfs.framework.security.PDAuthz.PDCheckAuthz, java.lang.String, java.lang.String)	
<ul style="list-style-type: none"> <li>• In authChecker - The authorization interface.</li> <li>• In user - The username of the user we are</li> </ul>	Constructor for <b>PDCheckEnvironment</b> . It shall save the application component (from which we get the security context), and create a new environment stack.

com.lmfs.framework.security.PDCheckEnvironment	<ul style="list-style-type: none"> <li>check authorization for.</li> <li>In userpac - The PAC of the user we are check authorization for. This can be null, in which case we shall use the username for making auth API calls.</li> </ul>	
private checkAuth (java.lang.String, com.lmfs.framework.security.PDCheckEnvironment.StackElement)	<ul style="list-style-type: none"> <li>In objName - the name of the object</li> <li>In e - the current environment (a StackElement)</li> </ul>	<p><b>CheckAuth</b> shall check authorization given an object name, and set the <b>inhibitOutput indicator</b>. Authorization is performed using the <b>AuthAPI</b>. The rules are as follows:</p> <ol style="list-style-type: none"> <li>If <b>inhibitOutput</b> is false (meaning that we are currently in data that we can see) then do <b>auth check</b> and set <b>inhibitOutput</b> based on that check.</li> <li>If <b>inhibitOutput</b> is true (meaning that we are currently in data that we cannot see) then do not bother doing the <b>auth check</b>, or resetting <b>inhibitOutput</b>, since we shall inhibit output even if the <b>auth check</b> would have passed</li> </ol>
public enterFailureSection ()		The <b>enterFailureSection</b> is provided to support the <b>&lt;IVFAIL&gt;</b> tag. It changes the current value of <b>inhibitOutput</b> , but does not affect any stacked values. It also remembers that we have seen an <b>&lt;IVFAIL&gt;</b> tag, so that we do not allow the user to have more than one <b>&lt;IVFAIL&gt;</b> in a single <b>&lt;IVCHECK&gt;</b> .
public getInhibitOutputState ()		<b>GetInhibitOutputState</b> is an accessor function to allow the caller to determine what the current environment's inhibit output state is.
public newEnv (java.lang.String)	<ul style="list-style-type: none"> <li>In newObjName - the object name</li> </ul>	The <b>newEnv</b> tag shall save the current environment on the stack, and create a new current environment using the parameters passed.
private newObjectEnv (java.lang.String, com.lmfs.framework.security.PDCheckEnvironment.StackElement)	<ul style="list-style-type: none"> <li>In newObjName - the object name</li> <li>In e - the current stack element</li> </ul>	The <b>newObjectEnv</b> tag shall create a new current environment using the parameters passed. It shall determine if output is to be inhibited based on the <i>Check Authorization</i> on the <b>objName</b> ( <b>checkAuth</b> shall determine whether to inhibit output).
public restoreEnv ()		The <b>restoreEnv</b> tag shall retrieve the last environment saved on the stack and make it the current environment.

### 3.2.5 com.lmfs.framework.security.AuthnInfo

**AuthnInfo** provides an interface for retrieving authentication information. This allows the MA to abstract the implementation of the actual authentication technique and shield developers from future API and Java wrapper changes.

**Table 16: com.lmfs.framework.security.AuthnInfo**

com.lmfs.framework.security.AuthnInfo	
public getClientIdent ()	Retrieves the <i>Policy Director ID</i> for the user. The returned PAC should be in a format ready for authorization checks.
public getGroups ()	Returns the Groups to which a given user belongs.
public hashCode ()	Returns a unique integer value derived from the Client Identity, Groups, and PAC. It has not been implemented.
public getPAC ()	Retrieves the <i>Policy Director PAC</i> for the user. The returned PAC should be in a format ready for call to <b>com.ibm.pd.Authzn.Azn.pac_get_creds</b>
public setClientIdent (java.lang.String)	<ul style="list-style-type: none"> <li>In string - A string used to set the <i>Client Identity</i>.</li> </ul> <p><i>Setter</i> function for the <b>ClientIdent</b> attribute.</p>
public setGroups (java.lang.String)	<ul style="list-style-type: none"> <li>In java.lang.String - A string to set the Groups to.</li> </ul> <p><i>Setter</i> function for the <b>Groups</b> attribute.</p>
public setPAC (java.lang.String)	<ul style="list-style-type: none"> <li>In newPac – New PAC to set in AuthnInfo structure.</li> </ul> <p><i>Setter</i> function for the <b>PAC</b> attribute.</p>

### 3.2.6 com.lmfs.framework.security.aznAPIInitialize

**aznAPIInitialize** is an interface that describes the standard set of routines for initializing the azn API used within the PDAuthnInfoImpl or PDCheckAuthzImpl classes.

**Table 17: com.lmfs.framework.security.aznAPIInitialize**

com.lmfs.framework.security.aznAPIInitialize	
public init (IV.Auth.AZNAPI)	<ul style="list-style-type: none"> <li>In apiObj-A valid aznAPI to set to the aznAPI.</li> </ul> <p>Calls <b>initApi</b> to initialize the <b>aznAPI</b> (sets the <b>azn object</b>) using an already-initialized <b>azn object</b>). We assume that the sent <b>aznAPI</b> object is initialized.</p>
public init (java.util.Properties)	<ul style="list-style-type: none"> <li>Prp java.util.Properties-Properties to initialize the aznAPI with.</li> </ul> <p>Initialize the <b>aznAPI</b> using the properties passed.</p>
public init (com.lmfs.framework.util.PropertiesExt)	<ul style="list-style-type: none"> <li>In com.lmfs.framework.util.PropertiesExt prp - PropertiesExt containing the properties to initialize the aznAPI with.</li> </ul> <p>Initialize the <b>aznAPI</b> using the <b>propertiesExt</b> passed.</p>
public init (java.util.ResourceBundle)	

com.lmfs.framework.security.aznAPIInitialize	
<ul style="list-style-type: none"> <li>In java.util.ResourceBundle rb - Resource Bundle containing the properties to initialize the aznAPI with.</li> </ul>	Initialize the <b>aznAPI</b> using the <b>ResourceBundle</b> passed.

### 3.3 com.lmfs.framework.security.PDAuthn.\*

This package provides classes that interface the authentication portions of the Integration Framework. It is used for authentication in the WebSphere AE environment. It should not be used in the WebSphere EE (CB) environment. Refer to Section 3.6 and Developer's Guide Section 5.3 for information on authenticating in the WebSphere EE (CB) environment.

The level of abstraction it provides greatly reduces the level of effort required when interfacing with the **aznAPI**, but proportionally limits the flexibility available.

The Security Package contains the following classes/packages:

#### com.lmfs.framework.security.PDAuthn.PDAuthnInfo

Interface for retrieving authentication information. The developer should call the methods on this interface or its parent interface, com.lmfs.framework.security.AuthnInfo.

#### ? com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl

- com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl

A corresponding Policy Director implementation of the **AuthnInfo** interface.

#### ? com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl

- com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl

A Servlet-centered implementation of the **PDAuthnInfo** interface

#### 3.3.1 com.lmfs.framework.security.PDAuthn.PDAuthnInfo

**PDAuthnInfo** provides an interface for retrieving authentication information. This allows the MA to abstract the implementation of the actual authentication technique and shield developers from future API and Java wrapper changes.

**PDAuthnInfo** extends the **com.lmfs.framework.security.AuthnInfo** interface, adding the **LDAPIAuthenticateUser()** method and defining a separate, extendible interface for Policy Director (PD) Authentication Routines.

**Table 18: com.lmfs.framework.security.PDAuthn.PDAuthnInfo**

com.lmfs.framework.security.PDAuthn.PDAuthnInfo	
public equals (java.lang.Object)	
<ul style="list-style-type: none"> <li>In pdAuthnInfoObject - an object also of type PDAuthnInfo to compare to the parent object of the method.</li> </ul>	The <b>equals</b> method compares the attributes of the method's parent object to those of the object passed in the parameter, and returns true if all attribute values are the same.
Public getAPI()	

com.lmfs.framework.security.PDAuthn.PDAuthnInfo	
	Returns the internal AZNAPI object (set or unset). See also setAPI(AZNAPI).
Public getClientIdent ()	Returns the Client Identity.
Public getGroups ()	Returns the client Groups.
Public hashCode ()	Returns a hash of the Client Identity, PAC, and Groups.
Public getPAC ()	Returns the client's PAC.
String getPACFromID ( String userID ) throws AZNException	
• In string - user DN for which to retrieve the PAC.	Returns a PAC given any user identity.
Public LDAPAuthenticateUser (java.lang.String, java.lang.String)	
• In string - user DN to authenticate with.	Provides a method of authentication using LDAP.
• In string - user <i>password</i> to authenticate with.	
public setAPI(AZNAPI anAPI)	
• In AZNAPI – the API to store in the PDAuthn object	Sets the API attribute.
public setClientIdent (java.lang.String)	Sets the client Identity attribute.
public setGroups (java.lang.String)	Sets the client <b>Groups</b> attribute.
public setPAC (java.lang.String)	Sets the client PAC attribute.

### 3.3.2 com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl

**PDAuthnInfoImpl** provides an implementation of the **PDAuthnInfo** interface. It is used to store authentication information and to provide an authentication interface. It makes calls to, and is thus dependent on, **IV.Auth.AZN API (aznapi.class)**.

Refer to the following Section 3.3.3

com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl for code examples as the **com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl** is extended by the **com.lmfs.framework.security.PDAuthn.Servlet.PdAuthnInfoImpl** class.

**Table 19: com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl**

com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl	
public PDAuthnInfoImpl ()	Default Constructor.
public equals (java.lang.Object)	
• In anObject – Object to compare to this object. Classname shall be the same.	Returns true if the passed object's attributes are value-equivalent.
Public boolean gcssafAuthenticateUser(in string userDn,in string password,inout IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo )	
◆ In string userDn – The x.500 Distinguished	Returns a boolean flag to indicate whether a user is

com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl	
<ul style="list-style-type: none"> <li>◆ Name of the user from the Policy Director user registry</li> <li>◆ In string password – the password that matches the password attribute of the Distinguished Name in the PD user registry.</li> <li>◆ Inout IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo</li> </ul>	authenticated. It also outputs a <b>UserSessionInfo</b> struct containing the user's identity information (username, groups the user belongs to, and the credentials of the user).
<pre>public IFCBSecurityInfo::SessionInfoStruct gcssafSetSessionInfo(in IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo,in string baseNameQualifier,in string baseNameCtx,in IFCBSecurityInfo::DynInfoType appDynInfo,in IFCBSecurityInfo::DynInfoType sessionDynInfo );</pre>	
<ul style="list-style-type: none"> <li>• In IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo -- returned by the gcssafAuthenticateUser method</li> <li>• In string baseNameQualifier -- identifies the initial levels of the Policy Director Object Namespace where this application resides. It is composed of the fixed /GCSS-AFAPPS root context for GCSS-AF applications and the functional domain. For our pseudo-supply test application, the baseNameQualifier is /GCSS-AFAPPS/ILS</li> <li>• In string baseNameCtx – is the location that the data supports or represents. Examples from the IF Test Components for this field are: ENTERPRISE, BASE1, BASE2, and BASE3.</li> <li>• In IFCBSecurityInfo::DynInfoType appDynInfo – a dynamic field that can be used for adding additional information required by an application</li> <li>• In IFCBSecurityInfo::DynInfoType sessionDynInfo – a dynamic field that can be used to supply additional information about a specific session</li> </ul>	Used to facilitate construction and initialization of a SessionInfo structure. Typically used after issuing a gcssafAuthenticateUser for building the rest of a SessionInfoStruct for passing to additional methods.
<pre>public AZNAPI getAPI()</pre>	
	Returns the current AZNAPI object stored in member variable <b>m_api</b> .
<pre>public getClientIdent ()</pre>	
	Retrieves the Policy Director ID for the user. The returned PAC should be in a format ready for Authorization Checks.
<pre>public getGroups ()</pre>	
	Returns the groups to which a given user belongs.
<pre>public hashCode ()</pre>	
	Returns a unique integer value derived from the Client Identity, Groups, and PAC. Not Yet Implemented.
<pre>public getPAC ()</pre>	
	Retrieves the Policy Director PAC for the user. The returned PAC should be in a format ready for call to <b>com.ibm.pd.Authzn.Azn.pac_get_creds</b>
<pre>public String getPACFromID ( String userID ) throws AZNException</pre>	
<ul style="list-style-type: none"> <li>• In string - user DN for which to retrieve the PAC.</li> </ul>	Returns a PAC given any user identity.

com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl	
<b>public init (IV.Auth.AZNAPI)</b>	
<ul style="list-style-type: none"> <li>In apiObj-A valid aznAPI to set to the aznAPI.</li> </ul>	Calls <b>initApi</b> to initialize the <b>aznAPI</b> (sets the <b>azn object</b> ) using an already-initialized <b>azn object</b> ). We assume that the sent <b>aznAPI</b> object is initialized.
<b>public init (java.util.Properties)</b>	
<ul style="list-style-type: none"> <li>Prp java.util.Properties-Properties to initialize the aznAPI with.</li> </ul>	Initialize the <b>aznAPI</b> using the properties passed.
<b>public init (com.lmfs.framework.util.PropertiesExt)</b>	
<ul style="list-style-type: none"> <li>In com.lmfs.framework.util.PropertiesExt prp - PropertiesExt containing the properties to initialize the aznAPI with.</li> </ul>	Initialize the <b>aznAPI</b> using the <b>propertiesExt</b> passed.
<b>public init (java.util.ResourceBundle)</b>	
<ul style="list-style-type: none"> <li>In java.util.ResourceBundle rb- Resource Bundle containing the properties to initialize the aznAPI with.</li> </ul>	Initialize the <b>aznAPI</b> using the <b>ResourceBundle</b> passed.
<b>public init (java.lang.String)</b>	
<ul style="list-style-type: none"> <li>S java.lang.String-Name of aznAPI properties file.</li> </ul>	Initialize the <b>aznAPI</b> using the <b>aznAPI</b> properties file specified.
<b>private initApi (com.lmfs.framework.util.PropertiesExt, IV.Auth.AZNAPI)</b>	
<ul style="list-style-type: none"> <li>APIObject IV.Auth.AZNAPI</li> </ul>	Initialized aznAPI (sets the azn object) using an already-initialized azn object). We assume that the sent aznAPI object is already initialized if the properties parameter (1) is null.
<b>public LDAPAuthenticateUser (java.lang.String, java.lang.String)</b>	
<ul style="list-style-type: none"> <li>UserDN java.lang.String-DN of user to <i>Authenticate</i>.</li> <li>Password java.lang.String-<i>Password</i> of user to <i>Authenticate</i>.</li> </ul>	Provides a method of authenticating using LDAP.
<b>public void setAPI ( AZNAPI anAPI )</b>	
<ul style="list-style-type: none"> <li>In AZNAPI – the API to store in the PDAuthn object</li> </ul>	Sets the API attribute.
<b>public setClientIdent (java.lang.String)</b>	
<ul style="list-style-type: none"> <li>In string - A string to set the <i>Client Identity</i> to.</li> </ul>	<i>Setter</i> function for the <b>ClientIdent</b> attribute.
<b>public setGroups (java.lang.String)</b>	
<ul style="list-style-type: none"> <li>In java.lang.String - A string to set the Groups to.</li> </ul>	Groups function for the <b>Groups</b> attribute.
<b>public setPAC (java.lang.String)</b>	
<ul style="list-style-type: none"> <li>In newPac – New PAC to set in AuthnInfo structure.</li> </ul>	Groups function for the PAC attribute.

### 3.3.3 com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl

**PDServletAuthnInfoImpl** is a Servlet implementation of the **PDAuthn** interface. It extends the com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl implementation of the com.lmfs.framework.security.PDAuthn abstract interface.

The following code example extracts the Policy Director username and credential from the HTTP header. The *getClientIdent* and *getPAC* methods are invoked by the Menu System and the IFServlet. The actual extensions created by this class are not used by the Integration Framework or the Test Components.

```

if (sessionInfoStruct == null)
{
    PDServletAuthnInfoImpl helper = new PDServletAuthnInfoImpl(request);
    String[] dsa1 = {" "}; //Dynamic String Array
    String[] dsa2 = {" "}; //Dynamic String Array
    String[] dsa3 = {" "}; //Dynamic String Array
    String[] dsa4 = {" "}; //Dynamic String Array

    String frameworkBaseNameQualifier = getParameter(request, "basenamequalifier", true,
true, true, null);
    //String frameworkBaseNameQualifier =
propertyManager.getString("frameworkBaseNameQualifier");
    UserSeInfoStruct usec = new UserSeInfoStruct(helper.getClientIdent(),
helper.getGroups(), helper.getPAC(), dsa1);
    UserSessionInfoStruct usess = new UserSessionInfoStruct(usec, dsa2);
    AppSessionInfoStruct asis = new AppSessionInfoStruct(location,
frameworkBaseNameQualifier, dsa3);
    sessionInfoStruct = new SessionInfoStruct(usess, asis, dsa4);
}

```

**Figure 23: Examples of getClientIdent, getGroups, and getPAC from the IFServlet getSessionInfoStruct method**

**Table 20: com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl**

com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl	
public PDServletAuthnInfoImpl (javax.servlet.http.HttpServletRequest)	Create a new User object from an HTTP request. Read all data from the HTTP header.
private extractUserGroupsFromRequest (javax.servlet.http.HttpServletRequest) <ul style="list-style-type: none"> <li>In req – The incoming HTTP request.</li> </ul>	Retrieves the Policy Director username from the HTTP request.
private extractUserIdentFromRequest (javax.servlet.http.HttpServletRequest) <ul style="list-style-type: none"> <li>In req – The incoming HTTP request.</li> </ul>	
	Retrieves the Policy Director username from the HTTP request.
private extractUserPACFromRequest (javax.servlet.http.HttpServletRequest) <ul style="list-style-type: none"> <li>In req - The incoming HTTP request.</li> </ul>	Retrieves the Policy Director PAC for the user from the HTTP request, without the version information. The

com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl	returned PAC should be in a format ready for call to azn_pac_get_creds.
--	---

### 3.3.4 com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl

**PDAppAuthnInfoImpl** is an application implementation of the **PDAuthn** interface. It extends the **com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl** implementation of the **com.lmfs.framework.security.PDAuthn** abstract interface.

Neither the Integration Framework nor its Test Components currently use this class. It should not be used at this time.

**Table 21: com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl**

com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl	
public PDAppAuthnInfoImpl ()	Default Constructor..

### 3.3.5 com.lmfs.framework.security.PDAuthn.CB.PDCBAuthnInfoImpl

**PDCBAuthnInfoImpl** is an application implementation of the **PDAuthn** interface. It extends the **com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl** implementation of the **com.lmfs.framework.security.PDAuthn** abstract interface.

PDCBAuthnInfoImpl is used in the IFCBSecurityInfo component broker component.

**Table 222: com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl**

com.lmfs.framework.security.PDAuthn.CB.PDCBAuthnInfoImpl	
public PDCBAuthnInfoImpl ()	Default Constructor..

### 3.3.6 com.lmfs.framework.security.PDAuthn.EJB.PDEJBAuthnInfoImpl

**PDEJBAuthnInfoImpl** is an EJB implementation of the **PDAuthn** interface. It extends the **com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl** implementation of the **com.lmfs.framework.security.PDAuthn** abstract interface.

Neither the Integration Framework nor its Test Components currently use this class. It should not be used at this time.

**Table 23: com.lmfs.framework.security.PDAuthn.EJB.PDEJBAuthnInfoImpl**

com.lmfs.framework.security.PDAuthn.EJB.PDEJBAuthnInfoImpl
public PDEJBAuthnInfoImpl ()
Default Constructor.

### 3.4 com.lmfs.framework.security.PDAuthz.\*

This package provides classes that interface the authorization portions of the Integration Framework. It is used for authorization in the WebSphere AE environment. It should not be used in the WebSphere EE (CB) environment. Refer to appendix section 3.6 and Developer's Guide Section 5.7 for information on authenticating in the WebSphere EE (CB) environment.

The level of abstraction it provides greatly reduces the level of effort required when interfacing with the **aznAPI**, but proportionally limits the flexibility available.

The Security Package contains the following classes/packages:

#### com.lmfs.framework.security.PDAuthz.PDCheckAuthz

provides an interface for authorization checks. IF developers should use this interface as much as possible over specific implementations (below).

public void setAPI ( AZNAPI anAPI )	
• In AZNAPI – the API to store in the PDAuthn object	Sets the API attribute.

#### com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException

public void setAPI ( AZNAPI anAPI )	
? In AZNAPI – the API to store in the PDAuthn object	Sets the API attribute.

#### com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException

An exception only thrown when an authorization check specifically denies access to a resource for a user.

#### com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl

a generic implementation of the **PDCheckAuthz** interface for authorization checks. IF developers should use the **PDCheckAuthz** interface and this implementation over more specific implementations (below).

#### com.lmfs.framework.security.PDAuthz.Application.PDCBAuthzImpl

**PDCBAuthzImpl** designates a package and class for authorization checks against Component Broker objects and classes. It builds upon (extends) its parent package and implementation, **PDAuthz.PDCheckAuthzImpl**.

com.lmfs.framework.security.PDAuthz.PDAppAuthzImpl
--

`com.lmfs.framework.security.PDAuthz.PDAppAuthzImpl`

`public PDEJBAuthzImpl()`

`Default Constructor.`

`com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl`

**`com.lmfs.framework.security.PDAuthz.Application.PDCBAuthzImpl`**

**PDCBAuthzImpl** designates a package and class for authorization checks against Component Broker objects and classes. It builds upon (extends) its parent package and implementation, **PDAuthz.PDCheckAuthzImpl**.

`com.lmfs.framework.security.PDAuthz.PDAppAuthzImpl`

`public PDEJBAuthzImpl()`

`Default Constructor.`

`com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl`

designates a package and class for authorization checks against *Component Broker* objects and classes.

**`com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl`**

**`com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl`**

designates a package and class for authorization checks against *Enterprise Java Bean (EJB)* components

**`com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl`**

a Servlet implementation of the **PDCheckAuthz** interface for Authorization checks

### 3.4.1 com.lmfs.framework.security.PDAuthz.PDCheckAuthz

**PDCheckAuthz** provides an interface for authorization checks. This allows the MA to abstract the implementation of the actual authorization engine to allow for ease of use within an application or Servlet. The application developer shall be shielded from alterations to the **aznAPI** in future *Policy Director* releases (such as when the *Native Interface* is removed.)

**Table 24: com.lmfs.framework.security.PDAuthz.PDCheckAuthz**

<code>com.lmfs.framework.security.PDAuthz.PDCheckAuthz</code>	
<pre>public Boolean gcssafDecisionAccessAllowed(IFCBSecurityInfo.UserSessionInfoStruct inUserSessionInfoData, String inSecLabelString, String inPermission, String appPDLLabel, int inCacheTimeout)</pre>	<ul style="list-style-type: none"> <li>◆ In IFCBSecurityInfo::UserSessionInfoStruct In UserSessionInfoData – structure identifying the user including the user's name, groups the user belongs to, and the user's credential. The user's credential is the only piece that is used to make the actual authorization decision. The other attributes of the structure are available for audit purposes.</li> <li>◆ In string inSecLabelString – string that identifies the object within the Policy Director Object Namespace. This may be a fully qualified reference within the Namespace or a relative reference. Relative references are prepended</li> </ul> <p>Evaluates whether a given user is authorized to perform a specific action (permission) on a specific object. The method returns true if the user is authorized.</p>

com.lmfs.framework.security.PDAuthz.PDCheckAuthz	
<ul style="list-style-type: none"> <li>with the appPDLLabel to create a fully qualified reference within the PD Object Namespace.</li> <li>◆ In string inPermission – a single character that identifies the permission, the action, to be checked</li> <li>◆ In string appPDLLabel – a string that identifies the upper levels of the Policy Director Namespace for this request. It is only used when the inSecLabelString is a relative reference</li> <li>◆ In long inCacheTimeout – a time value in seconds that identifies how long a permission in cache would be considered valid. This parameter is currently ignored a caching mechanism has not currently been implemented. Constants of NOCACHE, DEFAULTCACHE, and MAXCACHE are specified in IFCBSecurityInfo</li> </ul>	
<b>public AZNAPI getAPI()</b>	
	Returns the current AZNAPI object stored in member variable m_api.
<b>Public getAuthorizedObjects (java.lang.String, java.lang.String[], java.lang.String)</b>	
<ul style="list-style-type: none"> <li>• In infoImpl – Uses the creds stored in ai to accomplish the authorization check.</li> <li>• In objects – Checks each element in the array of string.</li> <li>• In permissions - Permissions to be checked on each object.</li> </ul>	Steps through a list of object names, determining the access decision for each, and returns the object names the user has access to in a new string array.
<b>Public init (java.lang.String)</b>	
<ul style="list-style-type: none"> <li>• In initargs - A string to initialize from.</li> </ul>	<p>Initializes any necessary class members using <b>initargs</b>. One necessary argument is <b>AZNAPIPropertiesFileLocation</b>, which should be passed as follows:  <b>AZNAPIPropertiesFileLocation=AZNAPI</b>, or <b>AZNAPI</b> if it is the only argument.</p>
<b>public PDCheckAuthorization (java.lang.String, java.lang.String, java.lang.String)</b>	
<ul style="list-style-type: none"> <li>• In username – The username of the individual for the authorization check.</li> <li>• In object - The object in the protected object namespace.</li> <li>• In Permset – The permissions being requested.</li> </ul>	<b>PDCheckAuthorization</b> performs an authorization check given a string username, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.
<b>public PDCheckAuthorizationPAC (java.lang.String, java.lang.String, java.lang.String)</b>	
<ul style="list-style-type: none"> <li>• In userpac – The PAC of the individual for the authorization check.</li> <li>• In object - The object in the protected object namespace.</li> <li>• In Permset – The permissions being</li> </ul>	<b>PDCheckAuthorization</b> performs an authorization check given a string PAC for a user, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.

com.lmfs.framework.security.PDAuthz.PDCheckAuthz requested.	
public void setAPI ( AZNAPI anAPI )	
• In AZNAPI – the API to store in the PDAuthn object	Sets the API attribute.

### 3.4.2 com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException

**PDAuthzUnauthorizedException** is an exception class for the **PDAuthz** interface methods that perform authorization checks. It shall only be thrown when an authorization check specifically denies access to a resource for a user. Errors are thrown via normal Exception objects.

**Table 25: com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException**

com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException	
public PDAuthzUnauthorizedException (java.lang.String, java.lang.String, java.lang.String)	
• In id – Client identity for which the authorization was checked. • In object - Object the client was attempting to access. • In Permset - the permissions the client desired on the object.	The constructor takes as the parameters the information for the authorization check that resulted in the access denied. This could be used in an application-level audit trail.
public PDAuthzUnauthorizedException (java.lang.String, java.lang.String, java.lang.String, java.lang.String[])	
• In id – Client identity for which the authorization was performed. • In object - Object the client was attempting to access. • In Permset - permissions desired on the object. • In objects[] – String array of objects checked.	The constructor takes as the parameters the information for the authorization check that resulted in the access denied. This could be used in an application-level audit trail.
Public static formatStringArray (java.lang.String[])	
• In objects - String array of objects. Returns a string of each element prepended with "Object: " and appended with "\n".	The constructor takes as the parameters the information for the authorization check that resulted in the access denied. This could be used in an application-level audit trail.
public toString ()	Returns the exception as a string. The string representation is set during construction.

### 3.4.3 com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl

**PDCheckAuthzImpl** is an implementation of the **PDCheckAuthz** interface for authorization checks.

Usage:

1. Construct a **PDCheckAuthzImpl** object.
2. Call **PDCheckAuthzImpl.init(String)** with the NAME of the properties file that the implementation should use for **aznAPI** calls. The NAME refers to only the root of the filename. i.e. if the full file path is <c:\h\ifsservices\lib\AZNAPI\_en\_US.properties> leave off the entire path, locale information, and file information and send simply "aznAPI". The implementation shall search the classpath for a file with the specified root. (Example: **PDCheckAuthzImplObj.init("aznAPI")**);
3. Call 1 or more of **PDCheckAuthorization**, **PDCheckAuthorizationPAC**, or **getAuthorizedObjects** as needed.

This implementation is extended by  
**com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl**. See section 3.4.7  
**com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl** for code examples.

**Table 26: com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl**

com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl	
public PDCheckAuthzImpl ()	Default Constructor.
public Boolean gcssafDecisionAccessAllowed(IFCBSecurityInfo.UserSessionInfoStruct inUserSessionInfoData, String inSecLabelString, String inPermission, String appPDLLabel, int inCacheTimeout)	<ul style="list-style-type: none"> <li>◆ In IFCBSecurityInfo::UserSessionInfoStruct In UserSessionInfoData – structure identifying the user including the user's name, groups the user belongs to, and the user's credential. The user's credential is the only piece that is used to make the actual authorization decision. The other attributes of the structure are available for audit purposes.</li> <li>◆ In string inSecLabelString – string that identifies the object within the Policy Director Object Namespace. This may be a fully qualified reference within the Namespace or a relative reference. Relative references are prepended with the appPDLLabel to create a fully qualified reference within the PD Object Namespace.</li> <li>◆ In string inPermission – a single character that identifies the permission, the action, to be checked</li> <li>◆ In string appPDLLabel – a string that identifies the upper levels of the Policy Director Namespace for this request. It is only used when the inSecLabelString is a relative reference</li> <li>◆ In long inCacheTimeout – a time value in seconds that identifies how long a permission in cache would be considered valid. This parameter</li> </ul>

com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl	
<p>is currently ignored a caching mechanism has not currently been implemented. Constants of NOCACHE, DEFAULTCACHE, and MAXCACHE are specified in IFCBSecurityInfo</p>	
<pre>public string gessafGetAuthorizedObjects(in IFCBSecurityInfo::UserSessionInfoStruct userSessionInfoData,in string inListOfSecLabels,in string inPermission,in string appPDLLabel,in long inCacheTimeout )</pre> <ul style="list-style-type: none"> <li>◆ In IFCBSecurityInfo::UserSessionInfoStruct In UserSessionInfoData – structure identifying the user including the user's name, groups the user belongs to, and the user's credential. The user's credential is the only piece that is used to make the actual authorization decision. The other attributes of the structure are available for audit purposes.</li> <li>◆ In string inListOfSecLabels – string that identifies the object within the Policy Director Object Namespace. This may be a fully qualified reference within the Namespace or a relative reference. Relative references are prepended with the appPDLLabel to create a fully qualified reference within the PD Object Namespace.</li> <li>◆ In string inPermission – a single character that identifies the permission, the action, to be checked</li> <li>◆ In string appPDLLabel – a string that identifies the upper levels of the Policy Director Namespace for this request. It is only used when the inSecLabelString is a relative reference</li> <li>• In long inCacheTimeout – a time value in seconds that identifies how long a permission in cache would be considered valid. This parameter is currently ignored a caching mechanism has not currently been implemented. Constants of NOCACHE, DEFAULTCACHE, and MAXCACHE are specified in IFCBSecurityInfo</li> </ul>	<p>Evaluates whether a given user is authorized to perform a specific action (permission) on a set of objects. The method returns a list of objects that the user is authorized to use.</p> <p>This method is not currently implemented.</p>
<pre>public getAuthorizedObjects (com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl, java.lang.String[], java.lang.String)</pre> <ul style="list-style-type: none"> <li>• In infoImpl – Uses the creds stored in ai to accomplish the authorization check.</li> <li>• In objects – Checks each element in the array of string.</li> <li>• In permissions - Permissions to be checked on each object.</li> </ul>	<p>Passes a list of objects to <b>aznAPI.getAuthorizedObjects</b>, which steps through the list of object names, determining the access decision for each, and returns the subset of object names the user has access to in a new string array.</p>
<pre>public getAuthorizedObjects (java.lang.String, java.lang.String[], java.lang.String)</pre> <ul style="list-style-type: none"> <li>• In pac - Uses the creds stored in pac to accomplish the authorization check.</li> <li>• In objects – Checks each element in array objects.</li> <li>• In permissions - Checks for all permissions on</li> </ul>	<p>Passes list of objects to <b>AZNAPI.getAuthorizedObjects</b>, which steps through the list of object names, determining the access decision for each, and returns the object names the user has access to in a new String array.</p>

com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl	
each object.	
public init (java.lang.String)	
• In initargs - A string to initialize from.	Initializes any necessary class members using initargs. One necessary argument is <b>aznAPIPropertiesFileLocation</b> , which should be passed as follows: <b>AZNAPIPropertiesFileLocation=\AZNAPI</b> , or <b>\aznAPI</b> if it is the only argument.
public isInitialized ()	
	Initializes any necessary class members using initargs. One necessary argument is <b>aznAPIPropertiesFileLocation</b> , which should be passed as follows: <b>aznAPIPropertiesFileLocation=aznAPI</b> resource which exists in the <b>CLASSPATH</b> Example: <b>aznAPIPropertiesFileLocation=/aznAPI</b>
public PDCheckAuthorization (com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl, java.lang.String, java.lang.String)	
• In authenObj – a PDAuthnInfoImpl object containing the valid credentials of the user for which the check shall be done • In object - The object in the protected object namespace. • In Permset - the permissions being requested on the object specified.	<b>PDCheckAuthorization</b> performs an authorization check given a string username, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.
public PDCheckAuthorization (java.lang.String, java.lang.String, java.lang.String)	
• In username – The username of the individual for the authorization check. • In object - The object in the protected object namespace. • In Permset – The permissions being requested.	<b>PDCheckAuthorization</b> performs an authorization check given a string username, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.
public PDCheckAuthorizationPAC (java.lang.String, java.lang.String, java.lang.String)	
• In userpac – The PAC of the individual for the authorization check. • In object - The object in the protected object namespace. • In Permset – The permissions being requested.	<b>PDCheckAuthorizationPAC</b> performs an authorization check given a PAC for a user's credentials, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.
public setInitialized (boolean)	
• In bool – Boolean value to set the initialized flag to.	Sets the <b>isInitialized</b> attribute (which is false by default) during initialization.

### 3.4.4 com.lmfs.framework.security.PDAuthz.Application.PDCBAuthzImpl

**PDCBAuthzImpl** designates a package and class for authorization checks against Component Broker objects and classes. It builds upon (extends) its parent package and implementation, **PDAuthz.PDCheckAuthzImpl**.

com.lmfs.framework.security.PDAuthz.PDAppAuthzImpl
public PDEJBAuthzImpl ()
Default Constructor.

### 3.4.5 com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl

**PDCBAuthzImpl** designates a package and class for authorization checks against Component Broker objects and classes. It builds upon (extends) its parent package and implementation, **PDAuthz.PDCheckAuthzImpl**.

PDCBAuthnInfoImpl is used in the IFCBSecurityInfo component broker component.

**Table 27: com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl**

com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl	
Public PDCBAuthzImpl ()	Default Constructor.
•	

### 3.4.6 com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl

**PDEJBAuthzImpl** designates a package and class for authorization checks against *Enterprise Java Bean (EJB)* components. It builds upon (extends) its parent package and implementation, **PDAuthz.PDCheckAuthzImpl**.

Neither the integration framework nor the test components currently implement this class. EJBs and CORBA Components should use the **IFCBSecurityInfo** Service for making authorization and authentication calls within EJB and CORBA Components.

**Table 28: com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl**

com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl	
public PDEJBAuthzImpl ()	Default Constructor.
•	

### 3.4.7 com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl

**PDServletAuthzImpl** is a Servlet implementation of the **PDCheckAuthz** interface for authorization checks.

Usage:

1. Construct a **PDServletAuthzImpl** object.

2. Call **PDServletAuthzImpl.init(String)** with the NAME of the properties file that the implementation should use for **aznAPI** calls. The NAME refers to only the root of the filename. i.e. if the full file path is **c:\h\ifsservices\lib\aznAPI\_en\_US.properties**, leave off the entire path, locale information, and file information and send simply **aznAPI**. The implementation shall search the classpath for a file with the specified root.
3. Call one or more of **PDCheckAuthorization**, **PDCheckAuthorizationPAC**, or **getAuthorizedObjects** as needed.

The following code example shows the construction of the **PDServletAuthzImpl** object and the initialization of the object.

```
public void init(javax.servlet.ServletConfig param1) throws javax.servlet.ServletException {  
    super.init(param1);  
    •••  
    authObject = new com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl();  
    try {  
        menuServletLogger.info("Initializing the aznAPI...");  
        authObject.init("AZNAPIPropertiesFileLocation=AZNAPI");  
    } catch (java.lang.Exception e) {  
        menuServletLogger.error("An error occurred initializing the aznAPI in MenuServlet.init()");  
    •••  
    }  
    •••  
    private Menu getUserMenu(HttpServletRequest request) {  
        •••  
        /*  
         * If the session is new one or the user was changed:  
         *  
         * 1. put the new user info value into the session  
         * 2. create the new menu object based on the current user info  
         * 3. put the created menu value into the session  
         */  
         If the session is not new one get the menu object from the session  
         *  
         if (session.isNew()) {  
             menuServletLogger.info("Session is NEW.\n  Creating User Menu...");  
             session.putValue(MenuServlet.SESSION_USER, aRequestUser);  
             menu = new Menu(request, session, aRequestUser, authObject);  
             session.putValue(MenuServlet.SESSION_MENU, menu);  
         } else {  
             menuServletLogger.info("Recycling User Menu...");  
             menu = (Menu) session.getValue(MenuServlet.SESSION_MENU);  
         }  
         return menu;  
    }
```

Figure 24: Example of Constructing and Initializing a PDServletAuthzImpl Object taken from the MenuServlet.java File of the Menu System Test Component

The following code segment from the Menu System Test Component exercises the **PDCheckAuthorizationPAC** method and the Unauthorized Exception. None of the test components currently use the **PDCheckAuthorization** or **getAuthorizedObjects** methods.

```
public Menu( HttpServletRequest aRequest, HttpSession aSession,
    com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl aUser,
    com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl inAuth ){

    if (menuLogger.isInfoEnabled()) menuLogger.info("Constructing menu for user " + aUser);
    auth = inAuth;
    CurrentUserSecInfo = aUser;
    menuItems = loadMenu();
    •••

    private class LDAPMenuLoader {
        •••

        private Vector authorizeMenuItems(Vector infoItems) {
            •••

            while (enum.hasMoreElements()) {
                LDAPInfoItem item = (LDAPInfoItem) enum.nextElement();
                if (menuLogger.isInfoEnabled()) menuLogger.info("Getting PD Formatted DN");
                java.lang.String formattedDN = com.lmfs.framework.util.StringExt.getPDFormattedDN(item.dn,
pdMenuBase);
                try {
                    if (false) throw new
com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException("", "", "");
                    if (menuLogger.isInfoEnabled()) menuLogger.info("About to check authorization, Creds= "
+ creds + "\n" +
                    "                               PD Name:  " + "/IF Menu" + formattedDN + "\n" +
                    "                               Permission: " + "r");
                    //Check authorization on the requested object. Convert the DN to PD format
(/app/func/etc)
                    auth.PDCheckAuthorizationPAC(creds, "/IF Menu" + formattedDN, "r");
                    if (menuLogger.isInfoEnabled()) menuLogger.info("authorization checked!!!!!!!");
                } catch (com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException unauth) {
                    if (menuLogger.isInfoEnabled()) menuLogger.info("The authorization failed for " + "/IF
Menu" + com.lmfs.framework.util.StringExt.getPDFormattedDN(item.dn, pdMenuBase));
                    //unauth.printStackTrace();
                    isAuthorized = false;
                } catch (Exception e) {
                    e.printStackTrace();
                    isAuthorized = false;
                }
                if (isAuthorized){
                    result.addElement(item);
                } else {
                    isAuthorized = true;
                }
            }
        }
    }
}
```

Figure 25: Example of PDCheckAuthorizationPAC taken from the Menu.java File of the Menu System Test Component

Table 29: com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl

<code>com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl</code>	
<code>public PDServletAuthzImpl ()</code>	Default Constructor.
<code>public PDServletAuthzImpl (javax.servlet.http.HttpServletRequest)</code>	
<ul style="list-style-type: none"> <li>• In param – Servlet request with which to use credential information.</li> </ul>	Default Constructor.
<code>public getAuthorizedObjects (java.lang.String[], java.lang.String)</code>	
<ul style="list-style-type: none"> <li>• In Object java.lang.String-Objects being checked for access.</li> <li>• In Perm java.lang.String-Permissions requested.</li> </ul>	Returns a list of authorized objects in a new string array.
<code>private getPDAuthnInfo ()</code>	
<ul style="list-style-type: none"> <li>• Param javax.servlet.http.HttpServletRequest- Servlet request with which to use credential information.</li> </ul>	Returns identity information related to this authorization object.
<code>public PDCheckAuthorization (java.lang.String, java.lang.String)</code>	
<ul style="list-style-type: none"> <li>• Param javax.servlet.http.HttpServletRequest- Servlet request with which to use credential information.</li> <li>• Object java.lang.String-Object for which to check access.</li> <li>• Perm java.lang.String-Permissions desired on object being checked.</li> </ul>	Check authorization based on set identity information.
<code>private setPDAuthnInfo (com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl)</code>	
<ul style="list-style-type: none"> <li>• NewPDAuthn com.lmfs.framework.security.PDAuthn.PDAuthnImpl</li> </ul>	Sets identity information related to this authorization object.

### 3.5 com.lmfs.framework.util.\*

This package provides general-purpose utility classes. It was originally designed on and used in a Windows NT/x86 environment, and was successfully tested on Solaris 2.7.

The classes provided in this package are:

**Table 30: com.lmfs.framework.util Class**

<code>com.lmfs.framework.util.StringExt</code>	A class that does not extend String, but contains one method- <b>getPDFormattedDN</b> , which converts an LDAP DN to a name in Policy Director format.
<code>com.lmfs.framework.util.ArrayExt</code>	A class that does not extend Array, but contains methods to convert array data.
<code>com.lmfs.framework.util.PropertiesExt</code>	A class that translates a variety of data formats into the Properties structure.

### 3.5.1 com.lmfs.framework.util.StringExt

**StringExt** is an extension of functionality to **java.lang.String**, specifically created for DN-to-Policy Director Object Name parsing. The class is also available for other future generic and miscellaneous IF-related string manipulation routines.

**Table 31: com.lmfs.framework.util.StringExt**

com.lmfs.framework.util.StringExt	
public StringExt ()	Default Constructor.
public static getPDFormattedDN (java.lang.String, java.lang.String)	<ul style="list-style-type: none"> <li>In BaseDN - portion of the DN that should be ignored when creating the Policy Director-Formatted name.</li> </ul> <p>Returns a DN in reverse format, delimited with '\', and without the BaseDN.          Example: DN "cn=Mike,ou=MyOrgUnit,o=MyOrg,c=us" as parameter 1 passed with DN Base as "o=MyOrg,c=us", would return \MyOrgUnit\Mike.</p>
public static main (java.lang.String[])	<ul style="list-style-type: none"> <li>In args - Unused in this class.</li> </ul> <p>A command-line test method.</p>

### 3.5.2 com.lmfs.framework.util.ArrayExt

**ArrayExt** is a *helper* class to extend the functionality of **java.util.Array**.

The usage is method-specific, as the helper class does not actually extend the array class. See each method for usage information.

**Table 32: com.lmfs.framework.util.ArrayExt**

com.lmfs.framework.util.ArrayExt	
Public ArrayExt ()	Default Constructor.
Public static arrayToString (java.lang.Object[])	<ul style="list-style-type: none"> <li>In Object - an array of objects. Assumed to be strings or instantiations of classes that have the <code>toString</code> method.</li> </ul> <p>Appends all <code>toString()</code> elements of the array directly after one another in a string, and returns that string.</p>
Public static arrayToString (java.lang.Object[], java.lang.String)	<ul style="list-style-type: none"> <li>In Object - an array of objects. Assumed to be strings or instantiations of classes that have the <code>toString</code> method.</li> <li>In string - the suffix to append to each element in the object array.</li> </ul> <p>Appends suffix (parameter 2) to all the <code>toString()</code> elements of the array directly after one another in a string, and returns that string.</p>
Public static arrayToString (java.lang.String, java.lang.Object[])	<ul style="list-style-type: none"> <li>In Object - an array of objects. Assumed to be strings or instantiations of classes that have the <code>toString</code> method</li> <li>In string - the prefix to insert in front of each element in the object array.</li> </ul> <p>Prefixes prefix (parameter 1) to all the <code>toString()</code> elements of the array directly after one another in a string, and returns that string.</p>

com.lmfs.framework.util.ArrayExt	
Public static arrayToString (java.lang.String, java.lang.Object[], java.lang.String)	
<ul style="list-style-type: none"> <li>• In string - the prefix to insert in front of each element in the object array.</li> <li>• In Object - an array of objects. Assumed to be strings or instantiations of classes that have the <code>toString</code> method.</li> <li>• In string - the suffix to append to each element in the object array.</li> </ul>	Prefixes prefix (parameter 1) and appends suffix (parameter 2) to all the <code>.toString()</code> elements of the array directly after one another in a string, and returns that string.
Private static doWork (java.lang.String, java.lang.String[], java.lang.String)	
	Does the actual work of converting array elements to strings, and prefixing or appending desired prefixes or suffixes.
public static main (java.lang.String[])	
<ul style="list-style-type: none"> <li>• Args java.lang.String[]</li> </ul>	A unit test method.

### 3.5.3 com.lmfs.framework.util.PropertiesExt

**PropertiesExt** is an extension of functionality to **java.util.Properties**. It is used to translate a variety of data formats into the Properties structure. See each constructor for usage details.

The following code shows an example of the use of the **com.lmfs.framework.util.PropertiesExt** class.

```
...
public boolean readGlobalRefreshValue() {
    menuServletLogger.info("Reading refresh.properties file...");

    // Use the classpath to find and load the Refresh.properties
    java.util.ResourceBundle bundle = null;
    try {
        bundle = (java.util.PropertyResourceBundle)
java.util.PropertyResourceBundle.getBundle("MenuServlet");
    } catch (java.util.MissingResourceException mrb) {
        menuServletLogger.error("Missing Resource Exception opening Refresh.properties
file:\n" +
            "Be sure the file exists in the classpath, and that all necessary\n" +
            "settings exist in the properties file.\n" + mrb);
    }
    com.lmfs.framework.util.PropertiesExt pe = new
com.lmfs.framework.util.PropertiesExt(bundle);
    return (pe.getProperty("REFRESH") == "TRUE");
}
```

Figure 26: Example of PropertiesExt from the MenuServlet.java file of the Menu System Test Component

Table 33: com.lmfs.framework.util.PropertiesExt

com.lmfs.framework.util.PropertiesExt
public PropertiesExt ()

com.lmfs.framework.util.PropertiesExt	
	Calls default parent constructor.
public PropertiesExt (InputStream)	
<ul style="list-style-type: none"> <li>In InputStream – InputStream to read.</li> </ul>	<p>This constructor uses the internal class ConfParse to parse a properties file that is formatted like an .ini file. The key/value pairs are prefixed by the ini file stanzas. For example, a file formatted:</p> <pre>[Stanza1] key1=value1 [Stanza2] key1=value1 key2=value2</pre> <p>would be read into the PropertiesExt class as</p> <pre>Stanza1.key1=value1 Stanza2.key1=value1 Stanza2.key2=value2</pre>
public PropertiesExt (InputStream, String)	
<ul style="list-style-type: none"> <li>In InputStream – InputStream to read, or null to use the second String parameter to locate the file using the classloader.</li> <li>In String – Filename to load as an InputStream.</li> </ul>	<p>This constructor uses the internal class ConfParse to parse a properties file that is formatted like an .ini file. The key/value pairs are prefixed by the ini file stanzas. For example, a file formatted:</p> <pre>[Stanza1] key1=value1 [Stanza2] key1=value1 key2=value2</pre> <p>would be read into the PropertiesExt class as</p> <pre>Stanza1.key1=value1 Stanza2.key1=value1 Stanza2.key2=value2</pre>
public PropertiesExt (java.lang.String)	
<ul style="list-style-type: none"> <li>In string - the key to read and parse.</li> </ul>	Constructs from a string. The parameter is a string to read in and parse, the delimiter is " ". (Thus format is " <b>key=value</b> <b>key=value</b> <b>key=value</b> "
public PropertiesExt (java.lang.String, java.lang.String)	
<ul style="list-style-type: none"> <li>In string - a list of key/value pairs, that are delimited by parameter 2.</li> <li>In string - the delimiter that separates the key/value pairs in parameter 1.</li> </ul>	Constructs from two strings. The first parameter is string to parse and read in, the second parameter is a major delimiter (delimiter around key/value pairs).
public PropertiesExt (java.util.Properties)	
<ul style="list-style-type: none"> <li>In props - a Properties object, this should be copied into the <i>PropertiesExt</i> object.</li> </ul>	Calls parent constructor with default properties set.
public PropertiesExt (java.util.ResourceBundle)	Constructs a properties object from a <b> ResourceBundle</b> .
public PropertiesExt (java.util.ResourceBundle, java.lang.String)	
<ul style="list-style-type: none"> <li>In rb - a resource bundle to read into this <i>PropertiesExt</i> object.</li> <li>In string - a filename that indicates that should indicate the Resource Bundle to load into this <i>PropertiesExt</i>.</li> </ul>	Constructs a properties object from a <b> ResourceBundle</b> . If <b> ResourceBundle</b> is null, constructs <b> ResourceBundle</b> from a file named in parameter two.
private loadConfWithInputStream(InputStream)	
<ul style="list-style-type: none"> <li>In InputStream – InputStream to read</li> </ul>	Routine that is called by the InputStream constructors. It uses the internal class ConfParse to parse a properties file

com.lmfs.framework.util.PropertiesExt	that is formatted like an .ini file.
public test ()	A test method.

## 3.6 SessionInfo Structures

The **SessionInfo** structure provides a common, extendable security data structure for passing required session information around the GCSS-AF system. In the Framework Architecture, WebSphere Advanced Edition Servlets combine context information from the Menu Servlet with security information from Policy Director WebSeal, and place that combined information in a **SessionInfo** structure. The Servlets then pass that information to Component Broker servers to enable method-level authorization checks.

### IDL for Security Structures

The **SessionInfo** is defined in the **IFCBSecurityInfo.IFCBSecurityInfoAuthz** class file located in the **IFCBSecurityInfoC.jar** file and in the C++ **IFCBSecurityInfoFile.hh** header file, and in the CORBA **IFCBSecurityInfo** Model.

The following types are defined: **DynInfoType**, **UserSecInfoStruct**, **AppSessionInfoStruct**, **UserSessionInfoStruct**, and **SessionInfoStruct**.

**DynInfoType-DynInfoType** is a dynamic field to allow extensibility without having to redefine the structure and force applications to recompile

**typedef sequence<string > DynInfoType;**

**UserSecInfoStruct-UserSecInfoStruct** includes the user parameters required to make an authorization decision.

```
struct UserSecInfoStruct {  
    string username;  
    string groups;  
    string creds;  
    IFCBSecurityInfo::DynInfoType dynInfo;  
}; // end struct UserSecInfoStruct
```

Figure 27: Example of UserSecInfoStruct Code

**AppSessionInfoStruct-AppSessionInfoStruct** includes parameters to aid the application in identifying its location in the Policy Director Object Space.

```
struct AppSessionInfoStruct {  
    string baseNameCtx;  
    string baseNameQualifier;  
    IFCBSecurityInfo::DynInfoType dynInfo;  
}; // end struct AppSessionInfoStruct
```

Figure 28: Example of AppSessionInfoStruct Code

**UserSessionInfoStruct-UserSessionInfoStruct** is a structure to allow additional non-security related information to be captured and passed around about the user.

```
struct UserSessionInfoStruct {  
    IFCBSecurityInfo::UserSeclInfoStruct userSeclInfoData;  
    IFCBSecurityInfo::DynInfoType dynInfo;  
}; // end struct UserSessionInfoStruct
```

Figure 29: Example of UserSessionInfoStruct Code

**SessionInfoStruct-SessionInfoStruct** is combines the Application and User Information about a session in a single structure to pass as a parameter amongst applications (Servlets, EJBs, and CORBA components)

```
struct SessionInfoStruct {  
    IFCBSecurityInfo::UserSessionInfoStruct userSessionInfoData;  
    IFCBSecurityInfo::AppSessionInfoStruct appSessionInfoData;  
    IFCBSecurityInfo::DynInfoType dynInfo;  
}; // end struct SessionInfoStruct
```

Figure 30: Example of SessionInfoStruct Code

### 3.7 IFCBSecurityInfo

The **IFCBSecurityInfo** Model is a Integration Framework Component that should be used by EJB's and CORBA objects for authenticating an application if necessary (end user's are

authenticated by WebSEAL). It is also used for making authorization decisions. [Table 34:](#) [IFCBSecurityInfoTable 34: IFCBSecurityInfo](#) provides a description of the methods available in the IFCBSecurityInfo Model. A description of their use is provided below. For additional information, refer to Section 5.3.2.4 of the GCSS-AF Developer's Guide.

**Table 34: IFCBSecurityInfo**

IFCBSecurityInfo::IFCBSecurityInfoAuthz	
<pre>public boolean gcssafDecisionAccessAllowed (in IFCBSecurityInfo::UserSessionInfoStruct inUserSessionInfoData,in string inSecLabelString,in string inPermission,in string appPDLLabel,in long inCacheTimeout )</pre>	<ul style="list-style-type: none"> <li>◆ In IFCBSecurityInfo::UserSessionInfoStruct In UserSessionInfoData – structure identifying the user including the user's name, groups the user belongs to, and the user's credential. The user's credential is the only piece that is used to make the actual authorization decision. The other attributes of the structure are available for audit purposes.</li> <li>◆ In string inSecLabelString – string that identifies the object within the Policy Director Object Namespace. This may be a fully qualified reference within the Namespace or a relative reference. Relative references are prepended with the appPDLLabel to create a fully qualified reference within the PD Object Namespace.</li> <li>◆ In string inPermission – a single character that identifies the permission, the action, to be checked</li> <li>◆ In string appPDLLabel – a string that identifies the upper levels of the Policy Director Namespace for this request. It is only used when the inSecLabelString is a relative reference</li> <li>◆ In long inCacheTimeout – a time value in seconds that identifies how long a permission in cache would be considered valid. This parameter is currently ignored a caching mechanism has not currently been implemented. Constants of NOCACHE, DEFAULTCACHE, and MAXCACHE are specified in IFCBSecurityInfo</li> </ul>
<pre>public string gcssafGetAuthorizedObjects(in IFCBSecurityInfo::UserSessionInfoStruct userSessionInfoData,in string inListOfSecLabels,in string inPermission,in string appPDLLabel,in long inCacheTimeout )</pre>	<ul style="list-style-type: none"> <li>◆ In IFCBSecurityInfo::UserSessionInfoStruct In UserSessionInfoData – structure identifying the user including the user's name, groups the user belongs to, and the user's credential. The user's credential is the only piece that is used to make the actual authorization decision. The other attributes of the structure are available for audit purposes.</li> <li>◆ In string inListOfSecLabels – string that identifies the object within the Policy Director Object Namespace. This may be a fully</li> </ul>

<b>IFCBSecurityInfo::IFCBSecurityInfoAuthz</b>	
<p>qualified reference within the Namespace or a relative reference. Relative references are prepended with the appPDLabel to create a fully qualified reference within the PD Object Namespace.</p> <ul style="list-style-type: none"> <li>◆ In string <b>inPermission</b> – a single character that identifies the permission, the action, to be checked</li> <li>◆ In string <b>appPDLabel</b> – a string that identifies the upper levels of the Policy Director Namespace for this request. It is only used when the <b>inSecLabelString</b> is a relative reference</li> <li>● In long <b>inCacheTimeout</b> – a time value in seconds that identifies how long a permission in cache would be considered valid. This parameter is currently ignored a caching mechanism has not currently been implemented. Constants of NOCACHE, DEFAULTCACHE, and MAXCACHE are specified in IFCBSecurityInfo</li> </ul>	
<b>public string gcssafGetClientAuditData();</b>	
	Returns the audit data for the most recent authorization check performed by a given security object.
<b>public IFCBSecurityInfo::SessionInfoStruct gcssafSetSessionInfo(in IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo,in string baseNameQualifier,in string baseNameCtx,in IFCBSecurityInfo::DynInfoType appDynInfo,in IFCBSecurityInfo::DynInfoType sessionDynInfo );</b>	
<ul style="list-style-type: none"> <li>● In IFCBSecurityInfo::UserSessionInfoStruct <b>userSessionInfo</b> -- returned by the <b>gcssafAuthenticateUser</b> method</li> <li>● In string <b>baseNameQualifier</b> -- identifies the initial levels of the Policy Director Object Namespace where this application resides. It is composed of the fixed /GCSS-AFAPPS root context for GCSS-AF applications and the functional domain. For our pseudo-supply test application, the <b>baseNameQualifier</b> is /GCSS-AFAPPS/ILS</li> <li>● In string <b>baseNameCtx</b> – is the location that the data supports or represents. Examples from the IF Test Components for this field are: ENTERPRISE, BASE1, BASE2, and BASE3.</li> <li>●</li> <li>● In IFCBSecurityInfo::DynInfoType <b>appDynInfo</b> – a dynamic field that can be used for adding additional information required by an application</li> <li>● In IFCBSecurityInfo::DynInfoType <b>sessionDynInfo</b> – a dynamic field that can be used to supply additional information about a specific session</li> </ul>	Used to facilitate construction and initialization of a SessionInfo structure. Typically used after issuing a <b>gcssafAuthenticateUser</b> for building the rest of a SessionInfoStruct for passing to additional methods.
<b>Public boolean gcssafAuthenticateUser(in string userDn,in string password,inout IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo )</b>	

IFCFSecurityInfo::IFCFSecurityInfoAuthz	
<ul style="list-style-type: none"> <li>◆ In string userDn – The x.500 Distinguished Name of the user from the Policy Director user registry</li> <li>◆ In string password – the password that matches the password attribute of the Distinguished Name in the PD user registry.</li> <li>◆ Inout IFCFSecurityInfo::UserSessionInfoStruct userSessionInfo</li> </ul>	Returns a boolean flag to indicate whether a user is authenticated. It also outputs a <b>UserSessionInfo</b> struct containing the user's identity information (username, groups the user belongs to, and the credentials of the user).

### 3.7.1.1.1 Authentication

The GCSS-AF Integration Framework relieves the MA from providing a separate authentication mechanism. The MA has the responsibility for trusting that WebSEAL has properly authenticated a user and accepts the identity that is passed to it as the originating identity for all actions. The MA will use this identity for Access Control and Audit requirements.

The MA has the responsibility for defining component identities in the scenarios identified in [Table 35: MA Authentication Scenarios](#). This is not meant to be an exhaustive list, but merely to stimulate the MA to think about how access control and the authentication mechanisms as part of its design task.

**Table 35: MA Authentication Scenarios**

Situation	Example
Components that are not invoked directly by an end user may need to authenticate to the system as a component identity.	<p>For example a trigger application that reads and processes messages off of a queue may not have access to the originating user information in a form that would allow it to make specific authorization decisions based on that originating user. The methods that the application is invoking require an identity to make its access control decisions.</p> <p>Another example is a wrapped legacy application where a file is FTP'd into a directory. An MA can process the file and invoke secured methods using the component's identity.</p>

Situation	Example
Components that need to act on behalf of a user to invoke methods that the user would not normally be given carte blanche access to may need to authenticate to the system as a component identity.	A Finance MA may provide a method to <b>checkForSufficientFunds</b> on an account based on an account number and a dollar amount. The finance MA doesn't want to open this method up to all end users as this may allow unauthorized users to gather information about other accounts. The finance MA creates another method called <b>userCheckForSufficientFunds</b> that verifies that the user has access to that account before providing feedback. Users are allowed access to this method and a component identity is created that is allowed access to the original <b>checkForSufficientFunds</b> method.

The IF provides the **gcssafAuthenticateUser** method that is part of the **IFCBSecurityInfoAuthz Object** in Component Broker to perform this function for the application. The **sessionInfoStruct** created by this function can be passed as a parameter to CORBA and EJB methods just as the **sessionInfoStruct** created for the user and passed in by the Servlet or EJB. To make applications work properly, it may be necessary to add additional parameters, as required by the invoked application. For example, the *location* and *basenamequalifier* parameters of the **sessionInfoStruct** are required by the test components.

```
com.ibm.IBOIMExtLocal._IUUIDPrimaryKeyImpl UUIDKey = new
com.ibm.IBOIMExtLocal._IUUIDPrimaryKeyImpl();
UUIDKey.generate();
static private org.omg.CORBA.Object obj=null;
obj = theHome.createFromPrimaryKeyString(UUIDKey.getUuid());
x = IFCBSecurityInfoAuthzHelper.narrow(obj);
IFCBSecurityInfoStructObjectHelper usi = new IFCBSecurityInfoStructObjectHelper();
UserSessionInfoStructHolder h = new UserSessionInfoStructHolder();
h.value = usi.sessionInfoStructData.userSessionInfoData;
booleanResult = x.gcssafAuthenticateUser("cn= ILS -BASE1-
PDC.TRIGMON,ou=USAF,ou=PKI,ou=DoD,o=U.S. Government,c=us",
"AppsPassword", h );
```

Figure 31: IFCBSecurityInfoStruct

Typically this would be followed by a call to **gcssafSetSessionInfo** for creating a **SessionInfoStruct** that can be passed around as the input parameter to other CORBA and EJB methods.

### 3.7.1.1.2 Providing Access Control Checks in MA Software

The tasks required to perform this Step:

- A. Accept the **sessionInfoStruct** type as a parameter to the method
- B. Build the parameters required by the **gcssafDecisionAccessAllowed** method

- C. Find/Create a **CBSecurityInfo** model instance (scope host-scope-widened)
- D. Invoke the **gcsafDecisionAccessAllowed** method
- E. Turning security on in SMUI

#### **Step A -Accept the sessionInfoStruct type as a parameter to the method**

Each method that needs to be protected with access control checks shall accept a **sessionInfoStruct** structure as a parameter. If the method does not directly make access control checks, but invokes other methods that do require access control checks, then the method would still need to accept a **sessionInfoStruct** structure to be able to pass it to the invoked methods. The sections below describe how to perform this for CORBA and EJB Server Components.

**CORBA Server Components** -See [Figure 32: Example IDL from PDCSessionModule of the PDC Test Component](#) [Figure 32: Example IDL from PDCSessionModule of the PDC Test Component](#) provides an example IDL for a CORBA Server Component. CORBA Server Components need to create a dependency on the IFCBSecurityInfo model in the Component Broker Object Builder tool when any of the methods use the **sessionInfoStruct** type. This dependency is reflected in the IDL by having the **IFCBSecurityInfoFile.idl** include file.

```
...
#include <IFCBSecurityInfoFile.idl>
...
module PDCSessionModule {
    interface PDCSessionAO;

    interface PDCSessionAO : IManagedClient::IManageable
    {
        ...
        ...
        void addPart(in long pdcID,in PDCHelperModule::PartRecord pRecord,in
IFCBSecurityInfo::SessionInfoStruct sessInfoStruct ) raises
        (IManagedClient::IDuplicateKey,PDCHelperModule::PDCException);
        ...
    }; // end interface PDCSessionAO
}; // end of module PDCSessionModule
```

Figure 32: Example IDL from PDCSessionModule of the PDC Test Component<sup>2</sup>

**EJB Server Components** -See [Figure 33: Example IDL from com\\_lmfs\\_framework\\_testcomponents\\_requisition\\_OrderingTie.java of the Requisition Component Test Component](#)  
[Figure 33: Example IDL from com\\_lmfs\\_framework\\_testcomponents\\_requisition\\_OrderingTie.java of the Requisition Component Test Component](#) provides an example IDL for an EJB Server Component. EJB Server Components need to have the **IFSServices.jar** file in their classpath in order to use the **sessionInfoStruct** type.

<sup>2</sup> Snippet taken from PDCSessionClasses.idl

```
...
public interface com_lmfs_framework_testcomponents_requisition_OrderingTie extends java.rmi.Remote
{
...
...
// Bean-specific business methods
    public java.lang.String placeAnOrder_object_(IFCBSecurityInfo.SessionInfoStruct arg0,
java.lang.String arg1, java.lang.String arg2, java.lang.String arg3, int arg4, java.lang.String arg5) throws
java.rmi.RemoteException, java.lang.Exception;
...
}
```

**Figure 33: Example IDL from com\_lmfs\_framework\_testcomponents\_requisition\_OrderingTie.java of the Requisition Component Test Component**

### **Step B -Build the parameters required by the gcssafDecisionAccessAllowed method**

To build the parameters required by the Access Control method, the MA Developer has to use information available from the code itself and information obtained and coordinated with the Operations and Support Organization.

The parameters of the *gcssafDecisionAccessAllowed* method are:

**Table 36: Parameters of the GCSSAF/DecisionAccessAllowed Method**

Parameter	How Obtained
UserSessionInfoData	Obtained from <b>sessionInfoStruct</b> parameter.  <b>This structure is created and passed in from the invoking application (Servlet, JSP, other CORBA or EJB Server Component) as part of the sessionInfoStruct parameter.</b>
SecLabelString	The Policy Director Object Space (relative or fully qualified context). If this field is formatted as a relative context, then it is pre-pended with the AppPdLabel field to define the Policy Director Object space to check the permission against. Additional information for determining the contents of this field is provided below
Permission	Action that the method is trying to perform.
AppPdLabel	Initial context. Built with information from the sessionInfoStruct parameter and MA specified information. Additional information for determining the contents of this field is provided below.
CacheTimeout	Should be set to 0 for now. The design was anticipating a caching mechanism in the future.

**UserSessionInfoData** -The **UserSessionInfoData** parameter is contained in the **sessionInfoStruct** that is passed as a parameter to applications that require security access control checks. The data for the **userSessionInfoData** piece of the **sessionInfoStruct** is created by Policy Director as a part of the authentication process.

**SecLabelString** -The purpose of the *SecLabelString* and the *AppPdLabel* in the *gcssafDecisionAccessAllowed* method is to identify the object as it is represented in Policy Director.

To create **SecLabelString**, the MA Developer can determine the names of the module, interface, and method of the method they are trying to protect. In the code example, the format that the Access Control method is expecting this information is as follows: **String secLabelStr = "[./PDCSessionModule/PDCSessionAO/addPart/]"**; Because it was prepended with a “.” it is a relative Policy Director Namespace Context and the **AppPdLabel** parameter will be used to fully qualify it.

**AppPdLabel** -To create the **AppPdLabel**, the MA Developer obtains certain information from the **sessionInfoStruct** parameter. The **sessionInfoStruct** parameter contains the **baseNameQualifier** and **baseName** context. The **baseNameQualifier** parameter identifies the initial levels of the Policy Director Object Namespace where this application resides. It is composed of the fixed /GCSS-AFAPPS root context for GCSS-AF applications and the functional domain. For our pseudo-supply test application, the *baseNameQualifier* is **/GCSS-AFAPPS/ILS**.

The **baseNameCtx** is the location that the data represents. Examples from the IF Test Components for this field are: ENTERPRISE, BASE1, BASE2, and BASE3.

It is incumbent on the invoking app to determine this information and supply

Appended to the end of the **AppPdLabel** is the Application Identity. As identified in **Step 4 - Determine Level and Type of Protection** of this process, this label is created by the MA and coordinated with the Operations and Support Organization.

The design of the application may determine an alternate approach of obtaining the **AppPdLabel** parameter. The recommendation is to limit the amount of hard coding, except for static content.

```
public void addPart( int pdcID, PDCHelperModule.PartRecord pRecord,  
IFCBSecurityInfo.SessionInfoStruct sessInfoStruct)  
  
throws com.ibm.IManagedClient.IDuplicateKey,  
PDCHelperModule.PDCEException  
{  
    ...  
    // local variables  
    // security Label String  
    String secLabelStr = "./PDCSessionModule/PDCSessionAO/addPart/" ;  
    // appPDLLabel String  
    String appPDLLabelStr = sessInfoStruct.appSessionInfoData.baseNameQualifier +  
    "/" + sessInfoStruct.appSessionInfoData.baseNameCtx +  
    "/PDC" ;  
    ...  
}
```

Figure 34: Example of Obtaining Parameters Required by the gcssDecisionAccessAllowed Method

**Step C -Find/Create an IFCBSecurityInfo model instance (scope server-scope-widened)**

[Figure 35: Code Example of Finding IFCBSecInfo Home](#) [Figure 35: Code Example of Finding IFCBSecInfo Home](#) and [Figure 36: Code Example of Creating IFCBSecurityInfo Instance](#) [Figure 36: Code Example of Creating IFCBSecurityInfo Instance](#) provide an example of the code necessary to create and locate an **IFCBSecurityInfo** model instance.

```
private com.ibm.IManagedClient.IHome getIFCBSecInfoHome()
{
/*
 * This method will locate the Home for the IFCBSecInfo class
 * It will use a server-scope-widened server based on the current
 * serverName to locate that home.
 */
...
// local variables
//      temp CORBA Object
org.omg.CORBA.Object obj = null ;

try {
    if (iIFCBSecInfoHome == null ) {
        // getSSWFactFinder() returns a name service with the following parameters
        // "host/resources/factory-finders/" +
        // CBSeriesGlobal.serverName() + "-server-scope-widened"
        obj = getSSWFactFinder().
            find_factory_from_string("IFCBSecurityInfo::IFCBSecurityInfoAuthz.object interface") ;
        ...
        iIFCBSecInfoHome = com.ibm.IManagedClient.IHomeHelper.narrow(obj) ;
        ...
    } // end if
}
catch (Exception exc) {
    logCat.error("PDCSessionAO-" + location() + "::getIFCBSecInfoHome::Exception caught: ", exc) ;
}
...
return iIFCBSecInfoHome ;
...
}
```

Figure 35: Code Example of Finding IFCBSecInfo Home

```
private IFCBSecurityInfo.IFCBSecurityInfoAuthz createIFCBSeclnfoAuthz()

{
    ...
    /**
     * This method creates the IFCBSecurityInfoAuthz object using the home
     * @return IFCBSecurityInfo.IFCBSecurityInfoAuthz
     */
    if ( iFCbSeclnfoAuthz == null ) {

        ...
        try {
            // create and generate the uuid key
            com.ibm.IBOIMExtLocal._IUUIDPrimaryKeyImpl iFCbSecKey = new
            com.ibm.IBOIMExtLocal._IUUIDPrimaryKeyImpl();

            iFCbSecKey.generate();

            org.omg.CORBA.Object obj = getIFCBSeclnfoHome().createFromPrimaryKeyString(
                iFCbSecKey.getUuid() );
            iFCbSeclnfoAuthz = IFCBSecurityInfo.IFCBSecurityInfoAuthzHelper.narrow( obj );
        }
        catch (Exception exc) {
            logCat.error("PDCSessionAO-" + location() + "::createIFCBSeclnfoAuthz::Exception caught: ", exc);
        }
        ...
    } // end if

    return iFCbSeclnfoAuthz;
}
}
```

Figure 36: Code Example of Creating IFCBSecurityInfo Instance

#### Step D -Invoke the gcsafDecisionAccessAllowed method

Refer to [Figure 37: Code Example of Invoking the gcsafDecisionAccessAllowed Method](#)  
[Figure 37: Code Example of Invoking the gesafDecisionAccessAllowed Method](#) as an example for invoking the *gcssafDecisionAccessAllowed* method.

```
public void addPart( int pdcID, PDCHelperModule.PartRecord pRecord,
IFCBSecurityInfo.SessionInfoStruct sessInfoStruct)

    throws com.ibm.IManagedClient.IDuplicateKey,
PDCHelperModule.PDCEception

{

    ...

    // check if user is allowed to access the method

    boolean accessAllowed = createIFCBSecInfoAuthz().gcssafDecisionAccessAllowed
( sessInfoStruct.userSessionInfoData,
secLabelStr,
"K",
appPDLLabelStr,
0 ) ;

    ...

if ( accessAllowed ) {

    try {

// Perform the action

    ...

}

    catch (com.ibm.IManagedClient.INoObjectWKey nowk) {

// used as an example catch for the code snippet, other exceptions are handled as well

    ...

    }

    ...

} // end if accessAllowed

    else {

// User not authorized. Perform error handling

        logCat.warn("PDCSessionAO-" + location() + "::addPart::ACCESS DENIED!!") ;

        throw new PDCHelperModule.PDCEception( "PDCSessionAO-" + location() + "::addPart::ACCESS DENIED!!" ) ;

    }

    ...

}

}
```

Figure 37: Code Example of Invoking the gcsafDecisionAccessAllowed Method

The method **gcessafGetClientAuditData** returns the audit data for the most recent authorization check performed by a given security object. This method works when each client obtains its own reference to a different security object, so the audit data storage is unique per client. Thus, the behavior of **gcssafGetClientAuditData** from an object with multiple or static references is undefined. The string returned contains the following data if available: the username, the user's credentials, the object for which access was checked, the permission desired on the object, and the result of the decision.