uCosminexus Application Server

API Reference Guide

3020-3-Y21-10(E)

■ Relevant program products

See the manual *uCosminexus Application Server Overview*.

■ Export restrictions

■ Trademarks

■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names:

Abbreviation			Full name or meaning
Windows	Windows Server 2008	Windows Server 2008 x86	Microsoft(R) Windows Server(R) 2008 Standard 32-bit
			Microsoft(R) Windows Server(R) 2008 Enterprise 32-bit
		Windows Server 2008 x64	Microsoft(R) Windows Server(R) 2008 Standard
			Microsoft(R) Windows Server(R) 2008 Enterprise
		Windows Server 2008 R2	Microsoft(R) Windows Server(R) 2008 R2 Standard
			Microsoft(R) Windows Server(R) 2008 R2 Enterprise
			Microsoft(R) Windows Server(R) 2008 R2 Datacenter
	Windows Server 2012	Windows Server 2012 Standard	Microsoft(R) Windows Server(R) 2012 Standard
		Windows Server 2012 Datacenter	Microsoft(R) Windows Server(R) 2012 Datacenter
	Windows XP		Microsoft(R) Windows(R) XP Professional Operating System
	Windows Vista	Windows Vista Business	Microsoft(R) Windows Vista(R) Business (32 bit)
		Windows Vista Enterprise	Microsoft(R) Windows Vista(R) Enterprise (32 bit)
		Windows Vista Ultimate	Microsoft(R) Windows Vista(R) Ultimate (32 bit)

Abbreviation			Full name or meaning
Windows	Windows 7	Windows 7 x86	Microsoft(R) Windows(R) 7 Professional (32 bit)
			Microsoft(R) Windows(R) 7 Enterprise (32 bit)
			Microsoft(R) Windows(R) 7 Ultimate (32 bit)
		Windows 7 x64	Microsoft(R) Windows(R) 7 Professional (64 bit)
			Microsoft(R) Windows(R) 7 Enterprise (64 bit)
			Microsoft(R) Windows(R) 7 Ultimate (64 bit)
	Windows 8	Windows 8 x86	Windows(R) 8 Pro (32 bit)
			Windows(R) 8 Enterprise (32 bit)
		Windows 8 x64	Windows(R) 8 Pro (64 bit)
			Windows(R) 8 Enterprise (64 bit)

Note that Windows 32 bit and Windows 64 bit are sometimes respectively referred to as Windows x86 and Windows x64.

■ Restrictions

■ Issued

Aug. 2013: 3020-3-Y21-10(E)

■ Copyright

Summary of amendments

The following table lists changes in the manual 3020-3-Y21-10(E) for uCosminexus Application Server 09-50, uCosminexus Application Server(64) 09-50, uCosminexus Client 09-50, uCosminexus Developer 09-50, uCosminexus Service Architect 09-50, uCosminexus Service Platform 09-50, uCosminexus Service Platform(64) 09-50 and product changes related to the manual:

Changes	Location
A description has been added for the client APIs of RESTful Web Services used to implement Web resource clients.	*1.1*
`@WebServiceRef` has been added to the support range of annotations included in the `javax.xml.ws` package.	*2.1.8*
A table has been added for the definable range of the annotations of Bean Validation.	*2.1.13*
An exception class of the EADs session failover disable functionality has been added to the APIs used with Web containers.	*3.1*
JDK6.0 is now supported.	*Appendix B*

In addition to the above changes, minor editorial corrections have been made.

Preface

For details on the prerequisites before reading this manual, see the manual *uCosminexus Application Server Overview*.

■ Non-supported functionality

Some functionality described in this manual is not supported. Non-supported functionality includes:

- Audit log functionality
- Compatibility functionality
- Cosminexus Component Transaction Monitor
- Cosminexus DABroker Library
- Cosminexus Reliable Messaging
- Cosminexus TPBroker and VisiBroker
- Cosminexus Web Service - Security
- Cosminexus XML Security - Core functionality
- JP1 linkage functionality
- Management portal functionality
- Migration functionality
- SOAP applications complying with specifications other than JAX-WS 2.1
- uCosminexus OpenTP1 linkage functionality
- Virtualized system functionality
- XML Processor high-speed parse support functionality

■ Non-supported compatibility functionality

"Compatibility functionality" in the above list refers to the following functionality:

- Basic mode
- Check of JSP source compliance (cjjsp2java) with the JSP1.1 and JSP1.2 specifications
- Database connection using Cosminexus DABroker Library
- EJB client application log subdirectory exclusive mode
- J2EE application test functionality
- Memory session failover functionality
- Servlet engine mode
- Simple Web server functionality
- Switching multiple existing execution environments
- Using EJB 2.1 and Servlet 2.4 annotation

Contents

1 Overview of APIs and Tag Libraries

This chapter describes the types of APIs and tag libraries used by Application Server and the formats for describing the APIs and tag libraries in this manual.

1.1 Types of APIs and Tag Libraries

This section describes the types of APIs and tag libraries used with Application Server.

In this manual, the APIs and tag libraries that can be used in each application are divided into the following three types:

- APIs and tag libraries that can be used for J2EE applications
- APIs that can be used for batch applications or EJB client applications
- APIs that can be used for the systems executing Web Services

The following table describes the APIs and the tag libraries that you can use for J2EE applications.

Table 1-1: APIs used with J2EE applications

Type of API and tag library	Description of API and tag library	Reference manual	Reference location
APIs used in the Web container	APIs used in the Web container	This manual	*Chapter 3*
APIs used in the EJB client application	APIs for setting the security and communication timeout of the EJB client.		*Chapter 4*
APIs used in TP1 inbound adapter to integrate with OpenTP1	APIs used in TP1 inbound adapter to integrate with OpenTP1.		*Chapter 5*
APIs used in the asynchronous parallel processing of threads	APIs used in the asynchronous parallel processing of threads.		*Chapter 6*
APIs used in the integrated user management framework	An integrated user management framework used for user authentication when the integrated user management function is used.	*uCosminexus Application Server Security Management Guide*	*Chapter 15*
Tag library used in the integrated user management framework	JSP tag library of the integrated user management framework used for user authentication when the integrated user management function is used.	*uCosminexus Application Server Security Management Guide*	*Chapter 16*
APIs used in the user log functionality	APIs used to output the user log when the log output by the J2EE application (user log) is to be output in the Hitachi trace common library format.	This manual	*Chapter 7*
APIs used to output audit logs	APIs used to output audit logs in J2EE applications.		*Chapter 8*
APIs used in the performance analysis trace	APIs for acquiring the root application information as a character string expression, when analyzing the processing efficiency of a Cosminexus system with the performance analysis trace.		*Chapter 9*
APIs used in JavaVM	APIs to acquire the memory information of the direct garbage collection from a Java program.		*Chapter 10*
APIs used in Cosminexus DABroker Library	APIs used to set the database information when you connect to the database by using Cosminexus DABroker Library.	*uCosminexus Application Server Compatibility Guide*	*Chapter 4*

Other than APIs and tag libraries, you can also use annotations and Dependency Injection. For details on annotations and Dependency Injection, see *2. Annotations and Dependency Injection Supported by Application Server*.

The following table describes the APIs that you can use for batch applications or EJB client applications.

Table 1‒2: APIs that can be used for batch applications or EJB client applications

Type of API and tag library	Description of API and tag library	Reference manual	Reference location
APIs used in the EJB client application	APIs used for setting up the security and the communication timeout of EJB client applications.	This manual	*Chapter 4*
APIs used in the user log functionality	APIs used to output the user log when you want to output the log (user log) to be output by batch application or EJB client application in the Hitachi trace common library format.		*Chapter 7*
APIs used to output audit logs	APIs used to output audit logs in batch applications or EJB client applications.		*Chapter 8*
APIs used in the performance analysis trace	APIs for acquiring the root application information as a character string expression, when analyzing the processing efficiency of a Cosminexus system with the performance analysis trace.		*Chapter 9*
APIs used in JavaVM	APIs to acquire the memory information of the direct garbage collection from a Java program.		*Chapter 10*
APIs used in Cosminexus DABroker Library	APIs used to set the database information when you want to connect to the database by using Cosminexus DABroker Library.	*uCosminexus Application Server Compatibility Guide*	*Chapter 4*

The following table describes the APIs that you can use with the systems executing Web Services.

Table 1‒3: APIs that can be used with the systems executing Web Services

Types of API	Explanation of API	Reference manual	Reference location
APIs used in the development of SOAP Web Services complying with the JAX-WS 2.2 specifications	APIs used when developing SOAP Web Services or Web Service clients.	*uCosminexus Application Server Web Service Development Guide*	*Chapter 19*
APIs used in the development of RESTful Web Services complying with the JAX-RS 1.1 specifications	APIs used when developing RESTful Web Services (Web resources). The HTTP client is developed using the client APIs for RESTful Web Services or the standard Java APIs.		*Chapter 24*
The client APIs for RESTful Web Services that are used for implementing Web resource clients	APIs used when implementing clients of RESTful Web Services (Web Services), with the client APIs for RESTful Web Services.		*Chapter 25*

1.2 Format for describing annotation

Chapter 2 describes the annotations in the following format. Note that each annotation is described in alphabetical order.

(1) **Description**

This section describes the function of the annotations.

(2) **Attribute**

This describes the attributes included in an annotation. Each attribute is described in the following format:

(a) Attribute name

Type

Indicates the type of the attributes.

Description

This section describes the function of the attributes.

Default value

Indicates the default value of the attributes.

1.3 Coding Format of APIs

APIs are described in the following format from chapter 3 to chapter 10. Each API is described according to the alphabetical order.

Description

This section describes the functions of the API.

Format

This section describes the coding format of the API.

Parameters

This section describes the API parameters.

Exceptions

This section describes the exceptions that occur when using the API.

Return value

This section describes the return value of the API.

Caution

This section describes the precautions to be taken when using the API.

2 Annotations and Dependency Injection Supported by Application Server

This chapter describes the annotations and Dependency Injection supported by Application Server.

If you are using the annotation reference disable functionality, an annotation specification is not referenced. For details on the annotation reference disable functionality, see *12.5 Controlling the annotation references* in the *uCosminexus Application Server Common Container Functionality Guide*.

2.1 Scope for the supported annotations

An annotation is a language specification that enables you to attach a comment to the source code.

The following table lists the annotations supported by Application Server.

2.1.1 Scope of support for the annotations included in the javax.annotation package

This section describes the applicability of annotations of the `javax.annotation` package. The following sections describe the annotations that you can code in each component:

(1) WAR file (Servlet 3.0 compliant)

The following table lists the annotations that you can code in a WAR file:

Table 2-1: Annotations (javax.annotation package) that can be coded in WAR file (Servlet 3.0 compliant)

Annotation name	Servlet specifications						JSP Specifications				Exception Class	Managed Bean(JSF)	Other class
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSPfile	Tag handler		Tag library event listener			
								Classic tag handler	Simple tag handler				
`@PostConstruct`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y[#]	--
`@PreDestroy`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y[#]	--
`@Resource`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
`@Resources`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not supported by standard specifications.

#

Annotation depends on JSF. For the support scope, see the JSF specification document.

(2) WAR file (Supported by Servlet 2.5)

The following table lists the annotations that you can code in a WAR file.

Table 2-2: Annotations (javax.annotation package) that can be coded in a WAR file (Supported by Servlet 2.5)

Annotation name	Servlet specifications			JSP specifications				Other class
	Servlet	Servlet filter	Event listener	JSP file	Tag handler		Tag library event listener	
					Classic tag handler	Simple tag handler		
`@PostConstruct`	Y	Y	Y	--	Y	Y	N	--

Annotation name	Servlet specifications			JSP specifications				Other class
	Servlet	Servlet filter	Event listener	JSP file	Tag handler		Tag library event listener	
					Classic tag handler	Simple tag handler		
`@PreDestroy`	Y	Y	Y	--	Y	Y	N	--
`@Resource`	Y	Y	Y	--	Y	Y	N	--
`@Resources`	Y	Y	Y	--	Y	Y	N	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(3) EJB-JAR File (EJB3.1/3.0 compliant)

The following table lists the annotations that you can code in an EJB-JAR file:

Table 2-3: Annotations (javax.annotation package) that can be coded in an EJB-JAR file (EJB3.1/3.0 compliant)

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than the default interceptor	Default interceptor		
`@PostConstruct`	--	Y	--	N	Y	Y	--	--
`@PreDestroy`	--	Y	--	N	Y	Y	--	--
`@Resource`	--	Y	--	N	Y	Y	--	--
`@Resources`	--	Y	--	N	Y	Y	--	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(4) Library JAR file (Servlets or JSPs)

The following table lists the annotations that you can code in a servlet or a JSP of a library JAR file:

Table 2-4: Annotations (javax.annotation package) that can be coded in a library JAR file (Servlets or JSPs)

Annotation name	Servlet specifications						JSP specifications			
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP file	Tag handler		Tag library event listener
								Classic tag handler	Simple tag handler	
`@PostConstruct`	--	--	Y	--	Y	--	--	Y	Y	N

Annotation name	Servlet specifications						JSP specifications			
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP file	Tag handler		Tag library event listener
								Classic tag handler	Simple tag handler	
`@PreDestroy`	--	--	Y	--	Y	--	--	Y	Y	N
`@Resource`	--	--	Y	--	Y	--	--	Y	Y	N
`@Resources`	--	--	Y	--	Y	--	--	Y	Y	N

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(5) Library JAR file (Enterprise Bean, exception class, or other classes)

The following table lists the annotations that you can coded in the Enterprise Beans, exception classes, or the other classes of a library JAR file:

Table 2-5: Annotations (javax.annotation package) that can be coded in a library JAR file (Enterprise Beans, exception classes, or other classes)

Annotation name	Enterprise Bean					Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor		
`@PostConstruct`	--	--	--	N	Y	--	--
`@PreDestroy`	--	--	--	N	Y	--	--
`@Resource`	--	--	--	N	Y	--	--
`@Resources`	--	--	--	N	Y	--	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

2.1.2 Scope of support for the annotations included in the javax.annotation.security package

This subsection describes the applicability of annotations included in the `javax.annotation.security` package. The following sections describe the annotations that you can code in each component:

(1) WAR File (Servlet 3.0 compliant)

The following table lists the annotations that you can code in a WAR file.

Table 2-6: Annotations (javax.annotation.security package) that can be coded in WAR file (Servlet 3.0 compliant).

Annotation name	Servlet specifications						JSP specifications				Exception class	Managed Bean(JSF)	Other class
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP file	Tag handler		Tag library event listener			
								Classic tag handler	Simple tag handler				
@DeclareRoles	Y	Y	Y	--	Y	--	--	--	--	--	--	--	--
@RunAs	Y	N	--	--	--	--	--	--	--	--	--	--	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not supported by standard specifications.

(2) WAR file (Supported by Servlet 2.5)

The following table lists the annotations that you can code in a WAR file:

Table 2-7: Annotations (javax.annotation.security package) that can be coded in a WAR file (Supported by Servlet 2.5)

Annotation name	Servlet specifications			JSP specifications				Other class
	Servlet	Servlet filter	Event listener	JSP file	Tag handler		Tag library event listener	
					Classic tag handler	Simple tag handler		
@DeclareRoles	Y	--	--	--	--	--	--	--
@RunAs	Y	Y	Y	--	--	--	--	--

Legend:

Y: Supported.

--: Not applicable.

(3) EJB-JAR file (EJB3.1/EJB3.0 compliant)

The following table lists the annotations that you can code in an EJB-JAR file:

Table 2-8: Annotations (javax.annotation.security package) that can be coded in an EJB-JAR file (EJB3.1/ EJB3.0 compliant)

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than the default interceptor	Default interceptor		
`@DeclareRoles`	--	Y	--	N	--	--	--	--
`@DenyAll`	--	Y	--	N	--	--	--	--
`@PermitAll`	--	Y	--	N	--	--	--	--
`@RolesAllowed`	--	Y	--	N	--	--	--	--
`@RunAs`	--	Y	--	N	--	--	--	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(4) Library JAR file (Servlets or JSPs)

The following table lists the annotations that you can code in a servlet or JSP of a library JAR file:

Table 2-9: Annotations (javax.annotation.security package) that can be coded in a library JAR file (Servlets or JSPs)

Annotation name	Servlet specifications						JSP specifications			
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP file	Tag handler		Tag library event listener
								Classic tag handler	Simple tag handler	
`@DeclareRoles`	--	--	Y	--	Y	--	--	--	--	--

Legend:

Y: Supported.

--: Not applicable.

(5) Library JAR file (Enterprise Bean, exception class, or other classes)

You cannot use the annotations in the Enterprise beans, exception classes, or the other classes of a library JAR file.

2.1.3 Scope of support for the annotations included in the javax.ejb package

This subsection describes the applicability of annotations of the `javax.ejb` package. The following sections describe the annotations that you can code in each component:

(1) WAR file (Servlet 3.0 compliant)

The following table lists the annotations that you can code in a WAR file:

Table 2-10: Annotations (javax.ejb package) that can be coded in WAR file (Servlet 3.0 compliant)

Annotation name	Servlet specifications						JSP specifications				Exception class	ManagedBean(JSF)	Other class
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP file	Tag handler		Tag library event listener			
								Classic tag handler	Simple tag handler				
`@ApplicationException`	--	--	--	--	--	--	--	--	--	--	Y	--	--
`@EJB`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
`@EJBs`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--

Legend:
Y: Supported.
N: Not supported by Application Server.
--: Not supported by standard specifications.

(2) WAR file (Supported by Servlet 2.5)

The following table lists the annotations that you can code in a WAR file:

Table 2-11: Annotations (javax.ejb package) that can be coded in a WAR file (Supported by Servlet 2.5)

Annotation name	Servlet specifications			JSP specifications				Other class
	Servlet	Servlet filter	Event listener	JSP file	Tag handler		Tag library event listener	
					Classic tag handler	Simple tag handler		
`@EJB`	Y	Y	Y	--	Y	Y	N	--
`@EJBs`	Y	Y	Y	--	Y	Y	N	--

Legend:
Y: Supported.
N: Not supported by Application Server.
--: Not applicable.

(3) EJB-JAR file (EJB3.1 compliant)

The following table lists the annotations that you can code in an EJB-JAR file:

Table 2-12: Annotations (javax.ejb package) that can be coded in an EJB-JAR file (EJB3.1 compliant)

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than default Interceptor	Default Interceptor		
@AccessTimeout[#1]	--	Y	--	--	--	--	--	--
@AfterBegin[#2]	--	Y	--	--	--	--	--	--
@AfterCompletion[#3]	--	Y	--	--	--	--	--	--
@ApplicationException	--	--	--	--	--	--	Y	--
@Asynchronous[#3]	--	Y	--	--	--	--	--	--
@BeforeCompletion[#2]	--	Y	--	--	--	--	--	--
@ConcurrencyManagement[#1]	--	Y	--	--	--	--	--	--
@DependsOn[#1]	--	Y	--	--	--	--	--	--
@EJB	--	Y	--	N	Y	Y	--	--
@EJBs	--	Y	--	N	Y	Y	--	--
@Init[#2]	--	Y	--	--	--	--	--	--
@Local	Y	Y	--	--	--	--	--	--
@LocalBean	--	Y	--	--	--	--	--	--
@LocalHome	--	Y	--	--	--	--	--	--
@Lock[#1]	--	Y	--	--	--	--	--	--
@Remote	Y	Y	--	--	--	--	--	--
@RemoteHome	--	Y	--	--	--	--	--	--
@Remove[#2]	--	Y	--	--	--	--	--	--
@Schedule[#3]	--	Y	--	N	--	--	--	--
@Schedules[#3]	--	Y	--	N	--	--	--	--
@Singleton[#1]	--	Y	--	--	--	--	--	--
@Startup[#1]	--	Y	--	--	--	--	--	--
@Stateful[#2]	--	Y	--	--	--	--	--	--

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than default Interceptor	Default Interceptor		
`@Stateless`[#4]	--	Y	--	--	--	--	--	--
`@Timeout`[#3]	--	Y	--	N	--	--	--	--
`@TransactionAttribute`	--	Y	--	N	--	--	--	--
`@TransactionManagement`	--	Y	--	N	--	--	--	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not supported by standard specifications.

#1

Can be used only for Singleton Session Bean.

#2

Can be used only for Stateful Session Bean.

#3

Can be used only for Stateless Session Bean and Singleton Session Bean.

#4

Can be used only for Stateless Session Bean.

(4) EJB-JAR file (Supported by EJB3.0)

The following table lists the annotations that you can code in an EJB-JAR file:

Table 2-13: Annotations (javax.ejb package) that can be coded in an EJB-JAR file (Supported by EJB3.0)

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than default Interceptor	Default Interceptor		
`@ApplicationException`	--	--	--	--	--	--	Y	--
`@EJB`	--	Y	--	N	Y	Y	--	--
`@EJBs`	--	Y	--	N	Y	Y	--	--
`@Init`[#1]	--	Y	--	--	--	--	--	--

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than default Interceptor	Default Interceptor		
`@Local`	Y	Y	--	--	--	--	--	--
`@LocalHome`	--	Y	--	--	--	--	--	--
`@Remote`	Y	Y	--	--	--	--	--	--
`@RemoteHome`	--	Y	--	--	--	--	--	--
`@Remove`#1	--	Y	--	--	--	--	--	--
`@Stateful`#1	--	Y	--	--	--	--	--	--
`@Stateless`#2	--	Y	--	--	--	--	--	--
`@Timeout`#2	--	Y	--	N	--	--	--	--
`@TransactionAttribute`	--	Y	--	N	--	--	--	--
`@TransactionManagement`	--	Y	--	N	--	--	--	--

Legend:
Y: Supported.
N: Not supported by Application Server.
--: Not applicable.

#1
Can be used only for Stateless Session Bean.

#2
Can be used only for Stateful Session Bean.

(5) Library JAR file (Servlets or JSPs)

The following table lists the annotations that you can code in a servlet or JSP of a library JAR file:

Table 2-14: Annotations (javax.ejb package) that can be coded in a library JAR file (Servlets or JSPs)

Annotation name	Servlet specifications						JSP specifications			
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP file	Tag handler		Tag library event listener
								Classic tag handler	Simple tag handler	
`@EJB`	--	--	Y	--	Y	--	--	Y	Y	N
`@EJBs`	--	--	Y	--	Y	--	--	Y	Y	N

Legend:
Y: Supported.
N: Not supported by Application Server.
--: Not applicable.

(6) Library JAR file (Enterprise Bean, exception class, or other classes)

The following table lists the annotations that you can code in the Enterprise Beans, exception classes, or the other classes of a library JAR file:

Table 2-15: Annotations (javax.ejb package) that can be coded in a library JAR file (Enterprise Beans, exception classes, or other classes)

Annotation name	Enterprise Bean					Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor		
`@ApplicationException`	--	--	--	--	--	Y	--
`@EJB`	--	--	--	--	Y	--	--
`@EJBs`	--	--	--	--	Y	--	--
`@Local`	Y	--	--	--	--	--	--
`@Remote`	Y	--	--	--	--	--	--

Legend:
Y: Supported.
--: Not applicable.

2.1.4 Scope of support for the annotations included in the javax.interceptor package

This subsection describes the applicability of annotations of the `javax.interceptor` package. The following sections describe the annotations that you can code in each component:

You can also use the `javax.interceptor` package annotations in a CDI application. However, take care when you use these annotations by combining with EJB. For details on the precautions to be taken, see *9. Using CDI with Application Server* in the *uCosminexus Application Server Common Container Functionality Guide*.

(1) WAR file (Servlet 3.0/Servlet 2.5 compliant)

No annotations can be used in a WAR file.

(2) EJB-JAR file (EJB3.1/EJB3.0 compliant)

The following table lists the annotations that you can code in an EJB-JAR file:

Table 2-16: Annotations (javax.interceptor package) that can be coded in an EJB-JAR file (EJB3.1/EJB3.0 compliant)

Annotation name	Enterprise Bean						Exception class	Other class
					Interceptor			
	Interface	Session Bean	Entity Bean	Message-driven Bean	Other than default Interceptor	Default Interceptor		
`@AroundInvoke`	--	Y	--	N	Y	Y	--	--
`@ExcludeClassInterceptors`	--	Y	--	N	--	--	--	--
`@ExcludeDefaultInterceptors`	--	Y	--	N	--	--	--	--

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than default Interceptor	Default Interceptor		
`@Interceptors`	--	Y	--	N	--	--	--	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(3) Library JAR file (Servlets or JSPs)

You cannot use the annotations in a servlet or JSP of a library JAR file.

(4) Library JAR file (Enterprise Bean, exception class, or other classes)

The following table lists the annotations that you can code in the Enterprise Beans, exception classes, or the other classes of a library JAR file:

Table 2-17: Annotations (javax.interceptor package) that can be use in a library JAR file (Enterprise Beans, exception classes, or other classes)

Annotation name	Enterprise Bean					Exception class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor	
`@AroundInvoke`	--	--	--	N	Y	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

2.1.5 Scope of support for the annotations included in the javax.jws package

For details on the support range of annotations included in the `javax.jws` package and each annotation, see *16.2 Customized mapping from Java to WSDL* in the *uCosminexus Application Server Web Service Development Guide*.

2.1.6 Scope of support for the annotations included in the javax.persistence package

The components in which the annotations of the `javax.persistence` package can be used differ based on the dependability on the JPA Provider. The annotations that depend on the JPA Provider and the annotations that do not depend on the JPA Provider are described separately:

(1) Annotations that depend on the JPA Provider

This point describes the applicability of the annotations that depend on the JPA Provider. The annotations that can be coded in each component are as follows:

(a) WAR file (Servlet 3.0 compliant)

The following table lists the annotations that you can code in a WAR file:

Table 2-18: Annotations (javax. persistence package) that can be coded in WAR file (Servlet 3.0 compliant)

Annotation name	Servlet specifications						JSP specifications				Exception class	Managed Bean (JSF)	Other class
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP FILE	Tag handler		Tag library event listener			
								Classic tag handler	Simple tag handler				
`@PersistenceContext`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
`@PersistenceContexts`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
`@PersistenceProperty`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
`@PersistenceUnit`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
`@PersistenceUnits`	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not supported by standard specifications.

(b) WAR file (Supported by Servlet 2.5)

The following table lists the annotations that you can code in a WAR file:

Table 2-19: Annotations (javax.persistence package) that can be coded in a WAR file (Supported by Servlet 2.5)

Annotation name	Servlet specifications			JSP specifications				Other class
	Servlet	Servlet filter	Event listener	JSP file	Tag handler		Tag library event listener	
					Classic tag handler	Simple tag handler		
`@PersistenceContext`	Y	Y	Y	--	Y	Y	N	--
`@PersistenceContexts`	Y	Y	Y	--	Y	Y	N	--
`@PersistenceProperty`	Y	Y	Y	--	Y	Y	N	--
`@PersistenceUnit`	Y	Y	Y	--	Y	Y	N	--
`@PersistenceUnits`	Y	Y	Y	--	Y	Y	N	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(c) EJB-JAR file (EJB3.1/EJB3.0 compliant)

The following table lists the annotations that you can code in an EJB-JAR file:

Table 2-20: Annotations (javax.persistence package) that can be coded in an EJB-JAR file (Supported by EJB3.0)

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than default Interceptor	Default Interceptor		
@PersistenceContext	--	Y	--	N	Y	Y	--	--
@PersistenceContexts	--	Y	--	N	Y	Y	--	--
@PersistenceProperty	--	Y	--	N	Y	Y	--	--
@PersistenceUnit	--	Y	--	N	Y	Y	--	--
@PersistenceUnits	--	Y	--	N	Y	Y	--	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(d) Library JAR file (Servlets or JSPs)

The following table lists the annotations that you can code in a servlet or JSP of a library JAR file.

Table 2-21: Annotations (javax.persistence package) that can be coded in a library JAR file (Servlets or JSPs)

Annotation name	Servlet specifications						JSP specifications			
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP FILE	Tag handler		Tag library event listener
								Classic tag handler	Simple tag handler	
@PersistenceContext	--	--	Y	--	Y	--	--	Y	Y	N
@PersistenceContexts	--	--	Y	--	Y	--	--	Y	Y	N
@PersistenceProperty	--	--	Y	--	Y	--	--	Y	Y	N
@PersistenceUnit	--	--	Y	--	Y	--	--	Y	Y	N
@PersistenceUnits	--	--	Y	--	Y	--	--	Y	Y	N

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(e) Library JAR file (Enterprise Bean, exception class, or other classes)

The following table lists the annotations that you can code in the Enterprise Beans, exception classes, or the other classes of a library JAR file:

Table 2-22: Annotations (javax.persistence package) that can be coded in a library JAR file (Enterprise Beans, exception classes, or other classes)

Annotation name	Enterprise Bean					Exception class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor	
`@PersistenceContext`	--	--	--	N	Y	--
`@PersistenceContexts`	--	--	--	N	Y	--
`@PersistenceProperty`	--	--	--	N	Y	--
`@PersistenceUnit`	--	--	--	N	Y	--
`@PersistenceUnits`	--	--	--	N	Y	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(2) Annotations that do not depend on the JPA Provider

Irrespective of the file type, you can use the annotations that do not depend on the JPA Provider, in the entity class.

For details on the list of annotations included in the `javax.persistence` package, see *2.7 javax.persistence package*.

2.1.7 Scope of support for the annotations included in the javax.servlet.annotation package

This subsection describes the applicable scope of the annotations of the `javax.servlet.annotation` package. The annotations that can be coded in each component are as follows:

(1) WAR file (Servlet 3.0 compliant)

The following table lists the annotations that you can code in a WAR file:

Table 2-23: Annotations (javax.servlet.annotation package) that can be coded in WAR file (Servlet 3.0 compliant)

Annotation name	Servlet specifications						JSP specifications				Exception class	Managed Bean(JSF)	Other class
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP FILE	Tag handler		Tag library event listener			
								Classic tag handler	Simple tag handler				
`@HandlesTypes`	--	--	--	--	--	--	--	--	--	--	--	--	Y
`@HttpConstraint`	Y	Y	--	--	--	--	--	--	--	--	--	--	--

Annotation name	Servlet specifications						JSP specifications				Exception class	Managed Bean(JSF)	Other class
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP FILE	Tag handler		Tag library event listener			
								Classic tag handler	Simple tag handler				
`@HttpMethodConstraint`	Y	Y	--	--	--	--	--	--	--	--	--	--	--
`@MultipartConfig`	Y	Y	--	--	--	--	--	--	--	--	--	--	--
`@ServletSecurity`	Y	Y	--	--	--	--	--	--	--	--	--	--	--
`@WebFilter`	--	--	Y	--	--	--	--	--	--	--	--	--	--
`@WebInitParam`	Y	--	Y	--	--	--	--	--	--	--	--	--	--
`@WebListener`	--	--	--	--	Y	--	--	--	--	--	--	--	--
`@WebServlet`	Y	--	--	--	--	--	--	--	--	--	--	--	--

Legend:

Y: Supported.

--: Not supported by standard specifications.

(2) EJB-JAR file

Annotation that can be coded in EJB-JAR file is not available.

(3) Library JAR (Servlet/JSP)

Annotation that can be coded in Servlet or JSP of library JAR is not available.

(4) Library JAR (Enterprise Bean/ Exception class/ Other classes)

Annotation that can be coded in Enterprise Bean of JAR library, exception class, and other class is not available.

2.1.8 Scope of support for the annotations included in the javax.xml.ws package

For details on the support range of annotations included in the `javax.xml.ws` package and each annotation, see *16.2 Customized mapping from Java to WSDL* in the *uCosminexus Application Server Web Service Development Guide*.

2.1.9 Support range of annotations included in the javax.xml.ws.soap package

For details on the support range of annotations included in the `javax.xml.ws.soap` package and each annotation, see *16.2 Customized mapping from Java to WSDL* in the *uCosminexus Application Server Web Service Development Guide*.

2.1.10 Support range of annotations included in the javax.xml.ws.spi package

For details on the support range of annotations included in the `javax.xml.ws.spi` package and each annotation, see *16.2 Customized mapping from Java to WSDL* in the *uCosminexus Application Server Web Service Development Guide*.

2.1.11 List of supported CDI annotations

The following table lists and describes the supported CDI annotations.

Package	Included annotation
`javax.decorator`	`@Decorator`
	`@Delegate`
`javax.enterprise.context`	`@ApplicationScoped`
	`@ConversationScoped`
	`@Dependent`
	`@NormalScope`
	`@RequestScoped`
	`@SessionScoped`
`javax.enterprise.event`	`@Observes`
`javax.enterprise.inject`	`@Alternative`
	`@Any`
	`@Default`
	`@Disposes`
	`@Model`
	`@New`
	`@Produces`
	`@Specializes`
	`@Stereotype`
	`@Typed`
`javax.inject`	`@inject`
	`@Named`
	`@Qualifier`
	`@Scope`
	`@Singleton`

The following sections describe the annotations (`@inject` annotations) that can be coded in each component. Annotations other than `@inject` depend on CDI. For details on annotations that depend on CDI, see the CDI specification documents.

(1) WAR file (Servlet 3.0 compliant)

The following table lists the annotations that you can code in a WAR file:

Table 2-24: Annotations (javax.inject package) that can be coded in WAR file (Servlet 3.0 compliant)

Annotation name	Servlet specifications						JSP specifications				Exception class	Managed Bean (JSF)	Other class
								Tag handler		Tag library event listener			
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP FILE	Classic tag handler	Simple tag handler				
`@Inject`	Y	--	Y	--	Y	--	N	N	N	N	N	Y	Y#

Legend:

Y: Supported.

--: Not supported by standard specifications.

N: Not supported by Application Server.

#

You can use corresponding components only if the components include CDI functions.

(2) EJB-JAR file (EJB3.1 compliant)

The following table lists the annotations that you can code in an EJB-JAR file:

Table 2-25: Annotations (javax. inject package) that can be coded in an EJB-JAR file (Supported by EJB3.0)

Annotation name	Enterprise Bean						Exception class	Other class
					Interceptor			
	Interface	Session Bean	Entity Bean	Message-driven Bean	Other than default Interceptor	Default Interceptor		
`@Inject`	N	Y	N	N	N	N	N	Y#

Legend:

Y: Supported.

N: Not supported by Application Server.

#

You can use corresponding components only if the components include CDI functions.

(3) Library JAR (Servlet/JSP)

Annotation that can be coded in Servlet or JSP of library JAR is not available.

(4) Library JAR (Enterprise Bean/ Exception class/ Other classes)

The following table lists the annotations that can be coded in Enterprise Bean of library JAR, Exception class, or other classes of library JAR:

Table 2-26: Annotations (javax. inject package) that can be coded in Library JAR (Enterprise Bean/ Exception class/Other classes)

Annotation name	Enterprise Bean					Exception class	Other classes
	Interface	Session Bean	Entity Bean	Message -driven Bean	Interceptor		
`@Inject`	N	N	N	N	N	N	Y#

Legend:
Y: Supported.
N: Not supported by Application Server.

#
You can use corresponding components only if the components include CDI functions.

2.1.12 List of supported JSF annotations

The following table lists and describes the supported JSF annotations:

Package	Included annotations
`javax.faces.application`	`@ResourceDependencies`
	`@ResourceDependency`
`javax.faces.bean`	`@ApplicationScoped`
	`@CustomScoped`
	`@ManagedProperty`
	`@NoneScoped`
	`@ReferencedBean`
	`@RequestScoped`
	`@SessionScoped`
	`@ViewScoped`
`javax.faces.component`	`@FacesComponent`
`javax.faces.component.behavior`	`@FacesBehavior`
`javax.faces.convert`	`@FacesConverter`
`javax.faces.event`	`@ListenerFor`
	`@ListenersFor`
	`@NamedEvent`
`javax.faces.render`	`@FacesBehaviorRenderer`
	`@FacesRenderer`
`javax.faces.validator`	`@FacesValidator`

The following sections describe the annotations (`@ManagedBean` annotations) that can be coded in each component. Note that the annotations other than `@ManagedBean` depend on JSF. For annotations that depend on JSF, see the JSF specification documents.

(1) WAR file (Servlet 3.0 compliant)

The following table lists the annotations that you can code in a WAR file:

Table 2-27: Annotations (javax.faces.bean package) that can be coded in a WAR file (Servlet 3.0 compliant)

Annotation name	Servlet specifications						JSP specifications				Exception class	ManagedBean (JSF)	Other class
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP FILE	Tag handler		Tag library event listener			
								Classic tag handler	Simple tag handler				
`@ManagedBean`	--	--	--	--	--	--	--	--	--	--	--	Y	--

Legend:
Y: Supported.
--: Not supported by standard specifications.

(2) EJB-JAR file (EJB3.1 compliant)

Annotation that can be coded in an EJB-JAR file is not available.

(3) Library JAR (Servlet/JSP)

Annotation that can be coded in Servlets or JSPs of library JAR is not available.

(4) Library JAR (Enterprise Bean/ Exception class/ Other classes)

Annotation that can be coded in Enterprise Bean, Exception class, and other library JAR classes is not available.

2.1.13 List of supported Bean Validation annotations

The following table lists and describes the supported Bean Validation annotations. Note that with Application Server, you can use Bean Validation from JSF and CDI.

Package	Annotation
`javax.validation`	`@Constraint`
	`@GroupSequence`
	`@OverridesAttribute`
	`@OverridesAttribute.List`
	`@ReportAsSingleViolation`
	`@Valid`
`javax.validation.constraints`	`@AssertFalse`
	`@AssertFalse.List`

Package	Annotation
`javax.validation.constraints`	`@AssertTrue`
	`@AssertTrue.List`
	`@DecimalMax`
	`@DecimalMax.List`
	`@DecimalMin`
	`@DecimalMin.List`
	`@Digits`
	`@Digits.List`
	`@Past`
	`@Pattern.List`
	`@Future`
	`@Future.List`
	`@Max`
	`@Max.List`
	`@Min`
	`@Min.List`
	`@Size`
	`@Size.List`
	`@NotNull`
	`@NotNull.List`
	`@Null`
	`@Null.List`
	`@Pattern`
	`@Pattern.List`

For Bean Validation annotations, see the Bean Validation specification documents.

The following table describes the definable range for the annotations of Bean Validation.

Item No.	Linkage target	javax.validation package	javax.validation.constraintspackage	Supported version
1	JSF linkage	Class on the class path	Class that specifies `@ManagedBean`	09-00
2	CDI linkage user application	Class on the class path	`JavaBeans` class#	09-50

#

When a user program manages the instances of the `JavaBeans` class, you can use annotations of Bean Validation with that class.

When a container (such as Servlet/EJB) manages the instances of the `JavaBeans` class, you cannot use annotations of Bean Validation with that class.

2.2 javax.annotation package

The following table lists the annotations included in the `javax.annotation` package:

List of annotations

Annotation name	Functionality
`@PostConstruct`	Specify the method that is called back immediately after the Servlet and Enterprise Bean instance are generated.
`@PreDestroy`	Specify the method that is called back immediately before deleting the Servlet, Enterprise Bean instance.
`@Resource`	Declare the resource reference.
`@Resources`	Specify multiple `@Resources`.

The following subsections describe the details of each annotation.

2.2.1 @PostConstruct

(1) Description

Set the method that is called back immediately after the Servlet and Enterprise Bean instance are generated.

(2) Element

`@PostConstruct` does not have any elements.

2.2.2 @PreDestroy

(1) Description

Set the method that is called back immediately before the Servlet, Enterprise Bean instance, and others are deleted.

(2) Element

`@PreDestroy` does not have any elements.

2.2.3 @Resource

(1) Description

Declare the resource reference. You can specify in a class, method, and field. When specified in a method or field, the annotation becomes a target for Dependency Injection. However, the method must be the `set` method.

(2) Element

The following table lists the elements of `@Resource`:

Element name	Function
`name`	Specify the name of resource reference. The specified name is used as a JNDI name. You can omit the element description if the annotation is specified in a method or field.

Element name	Function
`type`	Specify the `Java` type of a resource. You can omit the element description if the annotation is specified in a method or field.
`authenticationType`	Specify the authentication type used in the resource.
`shareable`	Specify whether the resource is to be shared.
`mappedName`	Specify the resource display name and queue name for specifying the referenced resource.
`lookup`	Specify the Portable Global JNDI name of any other resource that you reference or a resource alias.
`description`	Specify the resource description.

The details of each element are as follows:

(a) name element

Type

String

Description

Specify the name of resource reference. The specified name is used as a JNDI name. You can omit the element description if the annotation is specified in a method or field.

You can also specify a resource alias. For details on specifying a J2EE resource alias, see *2.6.6 Setting the optional names for the J2EE resources* in the *uCosminexus Application Server Common Container Functionality Guide*.

Default value

- When set in a method
  Property of the class name or `set` method in which the annotation is specified
- When set in a field
  Class name or field name in which the annotation is specified

(b) type element

Type

Class

Description

Specify the `Java` type of a resource. You can omit the element description if the annotation is specified in a method or field.

Default value

- When specified in a method
  Argument type of the method
- When set in a field
  Field type

`type` element and the corresponding DD

As the type element differs from J2EE specifications, the corresponding DD changes depending on the set value (Java Type). The following table describes the corresponding DD that changes depending on the Java Type:

Table 2-28: Table for the corresponding DD depending on the type element

`Type` element	DD tag corresponding to J2EE specifications	DD tag supported with Cosminexus Application Server specifications#1
`java.lang.String`#2	`Env-entry`	`env-entry`

Type element	DD tag corresponding to J2EE specifications	DD tag supported with Cosminexus Application Server specifications#1
java.lang.Character#2	env-entry	env-entry
java.lang.Integer#2	env-entry	env-entry
java.lang.Boolean#2	env-entry	env-entry
java.lang.Double#2	env-entry	env-entry
java.lang.Byte#2	env-entry	env-entry
java.lang.Short#2	env-entry	env-entry
java.lang.Long#2	env-entry	env-entry
java.lang.Float#2	env-entry	env-entry
javax.xml.rpc.Service	service-ref	Exception #3
javax.xml.ws.Service	service-ref	Exception #3
javax.jws.WebService	service-ref	Exception #3
javax.sql.DataSource	resource-ref	resource-ref
javax.jms.ConnectionFactory	resource-ref	resource-ref
javax.jms.QueueConnectionFactory	resource-ref	resource-ref
javax.jms.TopicConnectionFactory	resource-ref	resource-ref
javax.mail.Session	resource-ref	resource-ref
java.net.URL	resource-ref	Exception #3
javax.resource.cci.ConnectionFactory	resource-ref	resource-ref
org.omg.CORBA_2_3.ORB	resource-ref	resource-ref
Other connection factories defined by resource adapter	resource-ref	resource-env-ref
javax.jms.Queue	message-destination-ref	resource-env-ref
javax.jms.Topic	message-destination-ref	resource-env-ref
javax.resource.cci.InteractionSpec	resource-env-ref	Exception #3
javax.transaction.UserTransaction	resource-env-ref	resource-env-ref
javax.xml.ws.WebServiceContext	Undefined	resource-env-ref#4
All types other than those mentioned above#5	resource-env-ref	resource-env-ref

#1
: If `!#` is included in the `mappedName` element, correspond to `<resource-env-ref>`, irrespective of the Java Type.

#2
: You cannot acquire a value from a standard DD, as a result, the value is displayed in the element file but DI is not performed.

#3
: Exception in the case of import.

#4

With Application Server version 08-70 or earlier versions, this element is handled same as all types other than those mentioned above.

#5

With Application Server version 09-00, subclasses of the `java.lang.Class` and `java.lang.Enum` are not handled in the `<env-entry>` tag.

(c) authenticationType element

Type

AuthenticationType

Description

Specify the authentication type used in the resource.

Default value

`CONTAINER`

(d) shareable element

Type

boolean

Description

Specify whether the resource is to be shared.

Default value

`true`

(e) mappedName element

Type

String

Description

Specify the resource display name and queue name for specifying the referenced resource.

When characters other than single--byte alphanumeric characters and underscores (_) are to be included in the resource display name, replace them with underscores (_).

Default value

`""`

Setting conditions of the `mappedName` element

The setting conditions of the `mappedName` element change based on the `type` element. The following table describes the setting conditions of the `mappedName` element in `@Resource`:

Table 2-29: Setting conditions of mappedName() in @Resource

Setting condition (Java Type, resource)	Availability[#1]
`java.lang.String`	N
`java.lang.Character`	N
`java.lang.Integer`	N
`java.lang.Boolean`	N
`java.lang.Double`	N
`java.lang.Byte`	N
`java.lang.Short`	N
`java.lang.Long`	N

Setting condition (Java Type, resource)	Availability[#1]
`java.lang.Float`	N
`javax.xml.rpc.Service`	N
`javax.sql.DataSource`	Y
`javax.jms.ConnectionFactory`	Y
`javax.jms.QueueConnectionFactory`	Y
`javax.jms.TopicConnectionFactory`	Y
`javax.mail.Session`	Y
`java.net.URL`	N
`javax.resource.cci.ConnectionFactory`	Y
`org.omg.CORBA_2_3.ORB`	N
`javax.jms.Queue`[#2]	Y
`javax.jms.Topic`	Y
`javax.resource.cci.InteractionSpec`	N
`javax.transaction.UserTransaction`	N
`javax.ejb.EjbContext`	N
`javax.ejb.SessionContext`	N
`javax.ejb.TimerService`	N
JavaBeans resource	Y

Legend:

Y: Can be used.

N: Cannot be used.

#1

Mapping to the object to be managed is established with the `mappedName` element, irrespective of the Java Type. Use `!#` to demarcate the display name of the resource adapter and the name of the object to be managed.

#2

If you use `javax.jms.Queue` when using TP1/Message Queue - Access or Cosminexus Reliable Messaging, use `#` as the delimiter of the resource adapter display name and queue display name.

(f) lookup attribute

Type

`String`

Description

Specify the Portable Global JNDI name of any other resource or a resource alias that you want to reference.

Default value

`""`

(g) description element

Type

String

Description

Specify the resource description.

Default value
""

2.2.4 @Resources

(1) Description

Specify multiple `@Resource`. Note that you can specify multiple resources only in a class.

(2) Element

The following table lists the elements of `@Resources`:

Element name	Function
`value`	Define multiple resources (`@Resource`).

The details of each element are as follows:

(a) value element

Type
Resource[]

Description
Define multiple resources (`@Resource`).

Default value
None

2.3 javax.annotation.security package

The following table lists the annotations included in the `javax.annotation.security` package:

List of annotations

Annotation name	Functionality
`@DeclareRoles`	Set the security role reference.
`@DenyAll`	Set in a method that denies access to all security roles.
`@PermitAll`	Set in a class or method that permits access to all security roles.
`@RolesAllowed`	Set a security role for permitting access to a class or method.
`@RunAs`	Set the security role that is applied when executing Servlet or Enterprise Bean.

2.3.1 @DeclareRoles

(1) Description

Set the security role reference. Note that you can specify multiple resources only in a class.

(2) Element

The following table lists the elements of `@DeclareRoles`:

Element name	Function
`value`	Specify the security role name to be referenced.

The details of each element are as follows:

(a) value element

Type
String[]

Description
Specify the security role name to be referenced.

Default value
None

2.3.2 @DenyAll

(1) Description

Set in a method that denies access to all security roles.

(2) Element

`@DenyAll` does not have any elements.

2.3.3 @PermitAll

(1) Description

Set in a class or method that permits access to all security roles.

(2) Element

`@PermitAll` does not have any elements.

2.3.4 @RolesAllowed

(1) Description

Set a security role for permitting access to a class or method.

(2) Element

The following table lists the elements of `@RolesAllowed`:

Element name	Function
`value`	Specify the list of roles that have permission to access methods in an application.

The details of each element are as follows:

(a) value element

Type

String[]

Description

Specify the list of roles that have permission to access methods in an application.

Default value

None

2.3.5 @RunAs

(1) Description

Set the security role that is applied when executing Servlet or Enterprise Beans. Note that you can specify multiple resources only in a class.

(2) Element

The following table lists the elements of @RunAs:

Element name	Function
`value`	Specify the security role name that is applied when executing Enterprise Beans.

The details of each element are as follows:

(a) value element

Type

String

Description

Specify the name of the security role that is applied when executing Servlet or Enterprise Beans.

Default value

None

2.4 javax.ejb package

The following table lists the annotations included in the `javax.ejb` package:

List of annotations

Annotation name	Functionality
`@AccessTimeout`	Specify the timeout value of the concurrent access of Singleton Session Bean set by the Container Managed Concurrency.
`@AfterBegin`	Specify in a method that is called back immediately after starting the transaction of Stateful Session Bean.
`@AfterCompletion`	Specify in a method that is called back after completing the transaction of Stateful Session Bean.
`@ApplicationException`	Specify in an exception class that is considered as an application exception.
`@Asynchronous`	Specify in a business method that is executed asynchronously. Specify in a class and method of Stateless Session Bean or Singleton Session Bean.
`@BeforeCompletion`	Specify in a method that is called back immediately before completing the transaction of Stateful Session Bean.
`@ConcurrencyManagement`	Specify the type of the ConcurrencyManagement of Singleton Session Bean. Specify this annotation only in the Singleton Session Bean class.
`@DependsOn`	Specify to specify the dependency relation between Singleton Session Beans. Specify this annotation only in Singleton Session Bean class.
`@EJB`	Specify the reference to EJB business interface or home interface.
`@EJBs`	Specify multiple @EJB.
`@Init`	Specify in a method that is called back when create `<METHOD>()` defined in the home interface of Stateful Session Bean is executed.
`@Local`	Specify the local business interface of Enterprise Beans.
`@LocalBean`	Specify if Session Bean is specified as No-Interface view. Specify this annotation only in the Session Bean class.
`@LocalHome`	Specify in an Enterprise Bean class that supports the invocation using the local home interface, and local component interface.
`@Lock`	Specify the method to perform exclusive control when you are accessing business methods of a Singleton Session Bean in which the Container Managed Concurrency is set.
`@PostActivate`	Specify in a method that is called back immediately after Stateful Session Bean is activated.
`@PrePassivate`	Specify in a method that is called back immediately before Stateful Session Bean is passivated.
`@Remote`	Specify the remote business interface of Enterprise Beans. When an annotation is specified in an interface, that interface becomes a remote business interface.
`@RemoteHome`	Specify in an Enterprise Bean class that supports the invocation using a remote home interface, and remote component interface.
`@Remove`	Specify in a business method that deletes Stateful Session Beans.
`@Schedule`	Specify in a timeout method in which calendar base automatic generation timer of EJB timer service is called back.
`@Schedules`	Specify multiple`@Schedules`. Specify in a timeout method that is called back.
`@Singleton`	Specify this annotation in Singleton Session Bean class.

Annotation name	Functionality
`@Startup`	Specify this annotation when a Singleton Session Bean starts concurrently with application start up. Specify this annotation in Singleton Session Bean class.
`@Stateful`	Specify in Stateful Session Bean class.
`@Stateless`	Specify in a Stateless Session Bean class.
`@Timeout`	Specify in a timeout method that is called back when TimerService is used.
`@TransactionAttribute`	Specify transaction attributes when Enterprise Bean operates in CMT.
`@TransactionManagement`	Specify the transaction management type of the Enterprise Bean.

2.4.1 @AccessTimeout

(1) Description

Specify the timeout value of the concurrent access of Singleton Session Bean in which the Container Managed Concurrency is set.

(2) Attribute

The following table lists the `@AccessTimeout` attributes:

Attribute name	Functionality
`value`	Specify the time out value.
`unit`	Specify the unit of the timeout value.

Details of each attribute are as follows:

(a) value attribute

Type
`long`

Description
Specify the time out value.

Default value
`None`

(b) unit attribute

Type
`TimeUnit`

Description
Specify the unit of the timeout value.

Default value
`MILLISECONDS`

2.4.2 @AfterBegin

(1) Description

Specify in a method that is called back immediately after starting the transaction of Stateful Session Bean.

(2) Attribute

`@AfterBegin` attributes do not exist.

2.4.3 @AfterCompletion

(1) Description

Specify in a method that is called back after completing the transaction of Stateful Session Bean.

(2) Attribute

`@AfterCompletion` attributes do not exist.

2.4.4 @ApplicationExceptionn

(1) Description

Specify in an exception class that is considered as an application exception.

(2) Attribute

The following table lists the `@ApplicationException` properties:

Attribute name	Functionality
`rollback`	Specify whether the container performs the roll back of transaction when an exception occurs.
`inherited`	To decide whether this attribute is considered as an application exception, specify whether the definition set in the class is also applied in the subclass.

Details of each attribute are as follows.

(a) rollback attribute

Type

`boolean`

Description

Specify whether the container performs the roll back of transaction when an exception occurs.

Default value

`false`

(b) inherited attribute

Type

`boolean`

Description

To decide whether this attribute is considered as an application exception, specify whether the definition set in the class is also applied to the subclass.

Default value

`true`

2.4.5 @Asynchronous

(1) Description

Specify in a business method that is executed asynchronously. Specify in the class and method of Stateless Session Bean or Singleton Session Bean.

(2) Attribute

`@Asynchronous` attributes do not exist.

2.4.6 @BeforeCompletion

(1) Description

Specify in a method that is immediately called back before completing the transaction of Stateful Session Bean.

(2) Attribute

`@BeforeCompletion` attributes do not exist.

2.4.7 @ConcurrencyManagement

(1) Description

Specify the type of the ConcurrencyManagement of Singleton Session Bean. Set this annotation only in Singleton Session Bean class.

(2) Attribute

The following table lists the `@ConcurrencyManagement` attributes:

Attribute name	Functionality
`value`	Specify the type of the ConcurrencyManagement of Singleton Session Bean.

Details of each attribute are as follows:

(a) value attribute

Type

`ConcurrencyManagementType`

Description

Specify the type of the ConcurrencyManagement of Singleton Session Bean.

Default value

`CONTAINER`

2.4.8 @DependsOn

(1) Description

Specify the dependency relation between Singleton Session Beans. Specify this annotation only in the Singleton Session Bean class.

(2) Attribute

The following table lists the `@DependsOn` attribute:

Attribute Name	Functionality
`value`	Enumerate the EJB name of the dependent Singleton Session Bean.

Details of each attribute are as follows:

(a) value attribute

Type

`String[]`

Description

Enumerate the EJB name of the dependent Singleton Session Bean.

Default value

`None`

2.4.9 @EJB

(1) Description

Specify the reference to EJB business interface or home interface. You can specify in a class, method, and field. When specified in a method or field, the annotation becomes a target for Dependency Injection. However, the method must be set method.

(2) Element

The following table lists the elements of `@EJB`:

Element name	Functionality
`name`	Specify the name of resource reference. The specified name is used as a JNDI name. You can omit the element description if the annotation is specified in a method or field.
`beanInterface`	Specify the business interface class or home interface class. You can omit the element description if the annotation is specified in a method or field.
`beanName`	Specify the class name without the EJB package to be referenced. However, when the `name` element is specified in the annotation (`@Stateless`, `@Stateful`) that defines the EJB class to be referenced, specify the `name` element. Further, in the case of an EJB that supports the definition in DD, specify the value of DD `<ejb--name>tag`.
`mappedName`	You can specify the element, but you cannot run elements on Cosminexus because Cosminexus does not support elements.
`lookup`	Specify the Portable Global JNDI or the optional name of EJB that is referenced. However, if the `beanName` attribute or `mappedName` attributes are specified, settings of `beanName` attributes or `mappedName` attributes are preferred.
`description`	Specify the description of the EJB to be referenced.

The details of each element are as follows:

(a) name element

Type

String

Description

Specify the name of resource reference. The specified name is used as a JNDI name. You can omit the element description if the annotation is specified in a method or field.

Default value

- When set in a method
  Property of the class name or `set` method in which the annotation is specified
- When set in a field
  Class name or field name in which the annotation is specified

(b) beanInterface element

Type

Class

Description

Specify the business interface class or home interface class. You can omit the element description if the annotation is specified in a method or field.

Default value

- When specified in a method
  Argument type of the method
- When set in a field
  Field type

(c) beanName element

Type

String

Description

Specify a class name without the package of the referenced EJB. However, if the `name` attribute is specified in an annotation (`@Stateless`, `@Stateful`, `@Singleton`) that defines the EJB class to be referenced, specify the value of the `name` attribute. For the EJB that supports definition according to DD, specify the value of the `<ejb-name>` tag of DD.

Default value

""

(d) mappedName element

Type

String

Description

You can specify the element, but you cannot run elements on Cosminexus because Cosminexus does not support elements.

Default value

None

(e) lookup attribute

Type
`String`

Description
Specify the Portable Global JNDI or the optional name of EJB that is referenced. However, if `beanName` attributes or `mappedName` attributes are specified, settings of `beanName` attributes or `mappedName` attributes are preferred.

Default value
""

(f) description element

Type
String

Description
Specify the description of the EJB to be referenced.

Default value
""

2.4.10 @EJBs

(1) Description

Specify multiple `@EJB`. Note that you can specify multiple resources only in a class.

(2) Element

The following table lists the elements of `@EJBs`:

Element name	Function
`value`	Specify `@EJB`.

The details of each element are as follows:

(a) value element

Type
EJB[]

Description
Specify `@EJB`.

Default value
None

2.4.11 @Init

(1) Description

Set in a method that is called back when create `<METHOD>()` defined in the home interface of Stateful Session Bean is executed.

(2) Element

The following table lists the elements of `@Init`:

Element name	Function
`value`	Specify the corresponding `create<METHOD>()` name.

The details of each element are as follows:

(a) value element

Type

String

Description

Specify the corresponding `create<METHOD>()` name.

Default value

""

2.4.12 @Local

(1) Description

Set the local business interface of Enterprise Beans.. When you specify an annotation in an interface, that interface becomes a local business interface. When specifying the annotation in the `Bean` class, you need to specify the local business interface in the `value` element.

(2) Element

The following table lists the elements of `@Local`:

Element name	Function
`value`	When specifying annotations in the `Bean` class, specify the class of the local business interface.

The details of each element are as follows:

(a) value element

Type

Class[]

Description

When specifying annotations in the `Bean` class, specify the class of the local business interface.

Default value

`{}`

2.4.13 @LocalBean

(1) Description

Specify if the Session Bean is specified as No-Interface view. Specify this annotation only in the Session Bean class.

(2) Attribute

`@LocalBean` attributes do not exist.

2.4.14 @LocalHome

(1) Description

Specify in an Enterprise Bean class that supports the invocation using the local home interface, and local component interface.

(2) Element

The following table lists the elements of `@LocalHome`:

Element name	Function
`value`	Specify the local home interface.

The details of each element are as follows:

(a) value element

Type

Class

Description

Specify the local home interface.

Default value

None

2.4.15 @Lock

(1) Description

Specify a method to perform exclusive control when you are accessing business methods of Singleton Session Bean in which the Container Managed Concurrency is set.

(2) Attribute

The following table lists the `@Lock` attributes:

Attribute name	Functionality
`value`	Specify whether concurrent access is allowed (READ) or not allowed (WRITE) when you access business methods.

Details of each attribute are as follows:

(a) value attribute

Type

`LockType`

Description

Specify whether access is allowed (READ) or not allowed (WRITE) when you access business methods.

Default value
`WRITE`

2.4.16 @PostActivate

(1) Description

Set in a method that is called back immediately after Stateful Session Bean is activated. You can specify the annotations but you cannot run Cosminexus because Cosminexus does not support the status change of activation and passivation.

(2) Element

`@PostActivate` does not have any elements.

2.4.17 @PrePassivate

(1) Description

Set in a method that is called back immediately before Stateful Session Bean is passivated. You can specify the annotations but you cannot run Cosminexus because Cosminexus does not support the status change of activation and passivation.

(2) Element

`@PrePassivate` does not have any elements.

2.4.18 @Remote

(1) Description

Specify the remote business interface of Enterprise Beans. When an annotation is specified in an interface, that interface becomes a remote business interface. When specifying in the Bean class, you need to specify the remote business interface in the value element.

(2) Element

The following table lists the elements of `@Remote`:

Element name	Function
`value`	When specifying the annotation in the Bean class, specify the class of remote business interface.

The details of each element are as follows:

(a) value element

Type
Class[]

Description
When specifying the annotation in the Bean class, specify the class of remote business interface.

Default value
`{}`

2.4.19 @RemoteHome

(1) Description

Specify in an Enterprise Bean class that supports the invocation using the remote home interface, and remote component interface.

(2) Element

The following table lists the elements of `@RemoteHome`:

Element name	Function
`value`	Specify the remote home interface.

The details of each element are as follows:

(a) value element

Type
    Class

Description
    Specify the remote home interface.

Default value
    None

2.4.20 @Remove

(1) Description

Specify in the business method that deletes Stateful Session Beans.

(2) Element

The following table lists the elements of `@Remove`:

Element name	Function
`retainIfException`	Specify whether the element description is to be deleted when the method is ended abnormally in application exception.

The details of each element are as follows:

(a) retainIfException element

Type
    boolean

Description
    Specify whether the element description is to be deleted when the method is ended abnormally in application exception.

Default value
    `false`

2.4.21 @Schedule

(1) Description

Specify in the timeout method in which the calendar base automatic generation timer of the EJB timer service is called back.

(2) Attribute

The following table lists the `@Schedule` attributes:

Attribute name	Functionality
`dayOfMonth`	Set a day of the month for timeout.
`dayOfWeek`	Set a day of the week for timeout.
`hour`	Set the hour for timeout.
`info`	Set the optional character information related to the timer.
`minute`	Set the minutes for timeout.
`month`	Set the month for timeout.
`persistent`	Set persistence of timer. Since the persistence functionality of timer is not supported, the timer is not persistent even if you specify `true` and operate by considering as non persistent (`false`).
`second`	Set the seconds for timeout.
`timezone`	Set the time zone for timeout.
`year`	Set the year for timeout.

Details of each attribute are as follows:

(a) dayOfMonth attribute

Type

`String`

Description

Set a day of the month for timeout.

Default value

`"*"`

(b) dayOfWeek attribute

Type

`String`

Description

Set a day of the week for timeout.

Default value

`"*"`

(c) hour attribute

Type

`String`

Description

Set the hour for timeout.

Default value

`"0"`

(d) info attribute

Type

`String`

Description

Set the optional character information related to the timer.

Default value

`""`

(e) minute attribute

Type

`String`

Description

Set the minutes for timeout.

Default value

`"0"`

(f) month attribute

Type

`String`

Description

Set the month for timeout.

Default value

`"*"`

(g) persistent attribute

Type

`boolean`

Description

Set persistence of timer. Because the persistence functionality of timer is not supported, timer is not persistent even if you specify true and operate by considering as non persistent (false).

Default value

`true`

(h) second attribute

Type

`String`

Description

Set the timeout seconds.

Default value

`"0"`

(i) timezone attribute

Type
: `String`

Description
: Set the timeout time zone.

Default value
: `""`

(j) year attribute

Type
: `String`

Description
: Set the timeout year.

Default value
: `"*"`

2.4.22 @Schedules

(1) Description

Specify multiple numbers of `@Schedule` in the timeout method that is called back.

(2) Attribute

The following table lists the `@Schedules` attributes:

Attribute name	Functionality
`value`	Set `@Schedule`.

Details of each attribute are as follows:

(a) value attribute

Type
: `Schedule[]`

Description
: Set `@Schedule`.

Default value
: `None`

2.4.23 @Singleton

(1) Description

Specify this annotation in the Singleton Session Bean class.

(2) Attribute

The following table lists the `@Singleton` properties:

Attribute name	Functionality
`name`	Specify the name of Singleton Session Bean.
`mappedName`	Specify the optional name of Singleton Session Bean.
`description`	Specify the description of Singleton Session Bean.

Details of each attribute are as follows:

(a) name attribute

Type
`String`

Description
Specify the name of Singleton Session Bean.

Default value
`Class name` (excluding the package name) of Singleton Session Bean.

(b) mappedName attribute

Type
`String`

Description
Specify the optional name of Singleton Session Bean.

Default value
""

(c) description attribute

Type
`String`

Description
Specify the description of Singleton Session Bean.

Default value
""

2.4.24 @Startup

(1) Description

Specify this annotation when Singleton Session Bean starts concurrently with application start up. Specify this annotation in the Singleton Session Bean class.

(2) Attribute

`@Startup` attribute does not exist.

2.4.25 @Stateful

(1) Description

Set in Stateful Session Bean class.

(2) Element

The following table lists the elements of `@Stateful`:

Element name	Function
`name`	Specify Stateful Session Bean name.
`mappedName`	You can specify the element, but you cannot run elements on Cosminexus because Cosminexus does not support elements.
`description`	Specify the description of Stateful Session Bean.

The details of each element are as follows:

(a) name element

Type
: String

Description
: Specify Stateful Session Bean name.

Default value
: Class name excluding Stateful Session Bean package

(b) mappedName element

Type
: String

Description
: Specify the alias of Stateful Session Bean.

Default value
: None

(c) description element

Type
: String

Description
: Specify the description of Stateful Session Bean.

Default value
: ""

2.4.26 @Stateless

(1) Description

Set in a Stateless Session Bean class.

(2) Element

The following table lists the elements of `@Stateless`:

Element name	Function
`name`	Specify the Stateless Session Bean name.

Element name	Function
`mappedName`	You can specify the element, but you cannot run elements on Cosminexus because Cosminexus does not support elements.
`description`	Specify the description of Stateless Session Bean.

The details of each element are as follows:

(a) name element

Type
    String
Description
    Specify the Stateless Session Bean name.
Default value
    Class name excluding the Stateless Session Bean package

(b) mappedName element

Type
    String
Description
    Specify the alias of Stateless Session Bean.
Default value
    None

(c) description element

Type
    String
Description
    Specify the description of Stateless Session Bean.
Default value
    ""

2.4.27 @Timeout

(1) Description

Set in the timeout method that is called back when using the TimerService.

(2) Element

`@Timeout` does not have any elements.

2.4.28 @TransactionAttribute

(1) Description

Specify transaction attributes when Enterprise Bean operates in CMT. You can specify transaction attributes in class and method.

(2) Attribute

The following table lists the `@TransactionAttribute`:

Attribute name	Functionality
`value`	Set the transaction attributes.

Details of each attribute are as follows:

(a) value attribute

Type
`TransactionAttributeType`

Description
Set the transaction attributes.

Default value
`REQUIRED`

2.4.29 @TransactionManagement

(1) Description

Set the transaction management type of the Enterprise Bean.. Note that you can specify multiple resources only in a class.

(2) Element

The following table lists the elements of `@TransactionManagement`:

Element name	Function
`value`	Specify the transaction management type.

The details of each element are as follows:

(a) value element

Type
TransactionManagementType

Description
Specify the transaction management type.

Default value
`CONTAINER`

2.5 javax.faces.bean package

The following table lists the annotations included in the `javax.faces.bean` package:

Annotation list

Annotation name	Functionality
`@ManagedBean`	Specify the Managed Bean used by JSF.

For annotations other than this, see the JSF specification document.

2.5.1 @ManagedBean

(1) Description

Specify the Managed Bean used by JSF.

(2) Attribute

The following table lists the `@ManagedBean` attributes:

Attribute name	Functionality
`eager`	Specify whether a Managed Bean is to be generated when the Web application starts.
`name`	Specify the name of ManagedBean.

Details of each attribute are as follows:

(a) eager attribute

Type

`boolean`

Description

Specify whether a Managed Bean is to be generated when Web application starts. If `true` is set, Bean scope should be within the application scope.

Default value

`false`

(b) name attribute

Type

`String`

Description

Specify the name of ManagedBean.

If unspecified or Null, use the class name starting with a lower case character for the class that specifies annotation.

Example : For `java.examlpes.Bean`, the name becomes bean.

Default value

`""`

2.6 javax.interceptor package

The following table lists the annotations included in the `javax.interceptor` package:

List of annotations

Annotation name	Function
`@AroundInvoke`	Specify invocation of the business method in a method that intercepts.
`@ExcludeClassInterceptors`	Set in a method to which the class interceptor is not applied.
`@ExcludeDefaultInterceptors`	Specify in a class to which a default interceptor is not applied, and method.
`@Interceptors`	Specify the interceptor class to be applied. Note that you can specify the interceptor class in a class and method.

2.6.1 @AroundInvoke

(1) Description

Specify invocation of the business method in a method that intercepts.

(2) Element

`@AroundInvoke` does not have any elements.

2.6.2 @ExcludeClassInterceptors

(1) Description

Set in a method to which the class interceptor is not applied.

(2) Element

`@ExcludeClassInterceptors` does not have any elements.

2.6.3 @ExcludeDefaultInterceptors

(1) Description

Specify in a class to which a default interceptor is not applied, and method.

(2) Element

`@ExcludeDefaultInterceptors` does not have any elements.

2.6.4 @Interceptors

(1) Description

Specify the interceptor class to be applied. Note that you can specify the interceptor class in a class and method.

(2) Element

The following table lists the elements of `@Interceptors`:

Element name	Function
`value`	Specify the interceptor class to be applied.

The details of each element are as follows:

(a) value element

Type

Class[]

Description

Specify the interceptor class to be applied.

Default value

None

2.7 javax.persistence package

This section describes the list of annotations included in the `javax.persistence` package and the precautions to be taken when specifying annotations.

You can also specify the mapping information in an O/R mapping file instead of the annotations. For details on the correspondence between the annotations and the O/R mapping files, see *2.7.64 Correspondence between the annotations and O/R mapping*.

Precautions when specifying an annotation

- With the Cosminexus JPA, the annotations included in the `javax.persistence` package are not supported in the attributes related to the DDL output functionality.
- When specifying the same column name more than once in an annotation, arrange the upper case and lower case characters.
- If field names or method names are allocated in the column name, character strings are considered as upper case characters strings and used with Cosminexus JPA. If you want to specify a column name in the supported annotation, use upper case characters.
- The access type is decided according to the location at which the annotation is provided. However, if the access type exists in both, the field and property, the settings of the field will be enabled.
- The property name is decided as follows depending on the character string acquired by removing `get` or `set` (`is`) from the access method:
  - If the first two characters are in upper case, the string is used as it is.
  - If the first two characters are not in upper case, the first character is converted into lower case, and the string is used.
  - For a single character, the first character is converted into lower case, and the string is used.

List of annotations

Annotation classification	Annotation name	Overview
Entity annotation	`@Entity`	Indicates that the class is an entity.
Annotations related to the tables or columns	`@Column`	Specifies the mapping between the persistence field or the persistence property, and the columns of the database.
	`@JoinColumn`	Specifies the external key column for the binding table or a column of the binding--destination table that is referenced from the external key column by correlating the entity classes.
	`@JoinColumns`	Used when multiple `@JoinColumns` are coded concurrently.
	`@JoinTable`	This annotation specifies the binding table set up in the following classes: • Owner side class when ManyToMany relationship is specified. • Class with single-sided OneToMany relationship.
	`@PrimaryKeyJoinColumn`	Specifies the column used as the external key, when binding with other tables.
	`@PrimaryKeyJoinColumns`	Used when multiple `@PrimaryKeyJoinColumns` are coded concurrently.
	`@SecondaryTable`	Specifies a secondary table in the entity class.

Annotation classification	Annotation name	Overview
Annotations related to the tables or columns	`@SecondaryTables`	Used when multiple `@SecondaryTables` are coded concurrently.
	`@Table`	Specifies a primary table in the entity class.
	`@UniqueConstraint`	If you want to generate CREATE sentences for the primary table or secondary table, include the unique constraints, and then specify. Note that this annotation is not supported with Cosminexus JPA provider CJPA provider.
Annotations related to the ID	`@EmbeddedId`	Specifies the compound primary key of a class that can be embedded.
	`@GeneratedValue`	Specifies the method for automatically generating and allotting a unique value to the primary key column.
	`@Id`	Specifies the properties or fields of the primary key of the entity class.
	`@IdClass`	Specifies the compound primary key class mapped to multiple fields or properties of the entity class.
	`@SequenceGenerator`	Specifies the settings of the sequence generator that creates the primary key.
	`@TableGenerator`	Specifies the settings of the generator that creates the primary key.
Lock annotation	`@Version`	Specifies the `version` field or the `version` property for using the optimistic lock functionality.
Annotations related to mapping	`@Basic`	Indicates the type of mapping to the simplest database column.
	`@Embeddable`	Specifies an embedded class.
	`@Embedded`	Specifies the persistence property or the persistence field indicating the instance value of the embedded class within the entity class at the embedding destination.
	`@Enumerated`	Specifies the persistence field or the persistence property as the enumeration type.
	`@Lob`	Specifies the persistence field or the persistence property of the `large` object type supported by the database.
	`@MapKey`	Specifies the map key used for object identification within the map, when a non--owner entity class is indicated by the `java.util.Map` type in the `OneToMany` relationship or the `ManyToMany` relationship.
	`@OrderBy`	Specifies the order in which the collection is evaluated when the entity information is acquired.
	`@Temporal`	Specifies in the persistence property or persistence field having the type that expresses the time (`java.util.Date` and `java.util.Calendar`).

Annotation classification	Annotation name	Overview
Annotations related to mapping	`@Transient`	Specifies the field or property of a non--persisting entity class, mapped superclass, or embedded class.
Annotations related to the relationship	`@ManyToMany`	Indicates that the specified class has a `ManyToMany` relationship, and also specifies the multiple relationships from the owner entity class to the non--owner entity class.
	`@ManyToOne`	Indicates that the specified class has the `ManyToOne` relationship, and also specifies the relationship to the non--owner entity class.
	`@OneToMany`	Indicates that the specified class has the `OneToMany` relationship, and also specifies the multiple relationships from the owner entity class to the non--owner entity class.
	`@OneToOne`	Indicates that the specified class has the `OneToOne` relationship, and also specifies the single relationship between entity classes.
Annotations related to inheritance and overriding	`@AssociationOverride`	Overrides the settings used in the `ManyToOne` relationship or the `OneToOne` relationship specified in a mapped superclass and embedded class.
	`@AssociationOverrides`	Used when multiple `@AssociationOverrides` are coded concurrently.
	`@AttributeOverride`	Overrides the following mapping information: • Properties or fields specified by `@Basic` (or applied by default) • Properties or fields specified by `@Id`
	`@AttributeOverrides`	Used when multiple `@ElementOverrides` are coded concurrently.
	`@DiscriminatorColumn`	Specifies the column used for identification in the SINGLE_TABLE strategy or JOINED strategy. This annotation is added to an entity class that becomes a superclass by inheriting an entity class.
	`@DiscriminatorValue`	Specifies the value of the column used for identification in the SINGLE_TABLE strategy or JOINED strategy.
	`@Inheritance`	Specifies the inheritance mapping strategy used in the entity class hierarchy.
	`@MappedSuperclass`	Specifies a mapped superclass.
Annotations related to queries	`@ColumnResult`	Specifies the column for mapping the query results of an SQL to the entity class.
	`@EntityResult`	Specifies the entity class in which the query results of the SQL are to be mapped.
	`@FieldResult`	Specifies the field in which the query results of the SQL are to be mapped.

Annotation classification	Annotation name	Overview
Annotations related to queries	`@NamedNativeQueries`	Used when multiple `@NamedNativeQueries` are coded concurrently.
	`@NamedNativeQuery`	Specifies a named query in the SQL.
	`@NamedQueries`	Used when multiple `@NamedQueries` are coded concurrently.
	`@NamedQuery`	Specifies a named query of JPQL.
	`@QueryHint`	Specifies a database--specific query hint.
	`@SqlResultSetMapping`	Specifies the result set mapping of an SQL query.
	`@SqlResultSetMappings`	Used when multiple `@SqlResultSetMappings` are coded concurrently.
Annotations related to event callback[#]	`@EntityListeners`	Specifies the callback listener class used in the entity class or mapped superclass.
	`@ExcludeDefaultListeners`	This annotation excludes the default listener for the following classes: • Entity class • Mapped superclass • Subclass of the entity class or mapped superclass
	`@ExcludeSuperclassListeners`	This annotation excludes the superclass listener for the following classes: • Entity class • Mapped superclass • Subclass of the entity class or mapped superclass
	`@PostLoad`	This annotation indicates the callback method invoked after the SELECT statement is issued in the database.
	`@PostPersist`	This annotation indicates the callback method invoked after the INSERT statement is issued in the database.
	`@PostRemove`	This annotation indicates the callback method invoked after the DELETE statement is issued in the database.
	`@PostUpdate`	This annotation indicates the callback method invoked after the UPDATE statement is issued in the database.
	`@PrePersist`	This annotation indicates the callback method invoked before the INSERT statement is issued in the database.
	`@PreRemove`	This annotation indicates the callback method invoked before the DELETE statement is issued in the database.
	`@PreUpdate`	This annotation indicates the callback method invoked before the UPDATE statement is issued in the database.

Annotation classification	Annotation name	Overview
Annotations related to the reference of EntityManager and EntityManagerFactory	`@PersistenceContext`	Defines the container--managed EntityManager.
	`@PersistenceContexts`	Used when multiple `@PersistenceContexts` are coded concurrently.
	`@PersistenceProperty`	Sets up properties in the container--managed EntityManager.
	`@PersistenceUnit`	Defines the persistence unit for the EntityManagerFactory.
	`@PersistenceUnits`	Used when multiple `@PersistenceUnits` are coded concurrently.

# For details on the callback method, see *6.15 How to specify the callback method* in the *uCosminexus Application Server Common Container Functionality Guide*.

2.7.1 @AssociationOverride

(1) Description

This annotation overrides the settings used in the `ManyToOne` relationship or the `OneToOne` relationship specified in a mapped superclass and an embedded class.

When `@AssociationOverride` is not specified, the external key column is mapped in the same way as the original mapping.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@AssociationOverride`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the field or property having the related mapping that is to be overridden.
`joinColumns`	Required	This element specifies an array of `@JoinColumn`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the field or property having the related mapping that is to be overridden.

Default value

None

(b) joinColumns element

Type

JoinColumn[]

Description

This element specifies an array of `@JoinColumn`.

The definition of the mapped superclass or embedded class is applied as the mapping type.

You can specify the value within the specifiable range of the arrays of `@JoinColumn`. For details, see *2.7.24 @JoinColumn*.

Default value

None

2.7.2 @AssociationOverrides

(1) Description

This annotation is specified when multiple `@AssociationOverrides` are coded concurrently.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@AssociationOverrides`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@AssociationOverride`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

AssociationOverride[]

Description

This element specifies an array of `@AssociationOverride`.

You can specify the value within the specifiable range of the arrays of `@AssociationOverride`. For details, see *2.7.1 @AssociationOverride*.

Default value

None

2.7.3 @AttributeOverride

(1) Description

This annotation overrides the following mapping information:

- Properties or fields specified by `@Basic` (applied by default)
- Properties or fields applied by default
- Properties or fields specified by `@Id`

To override the settings of `@Column` defined in the mapped superclass and embedded class, apply the field or property of the entity class and embedded class in which the mapped superclass is inherited.

If `@AttributeOverride` is not specified, the column is mapped with the original mapping before override.

If `@AttributeOverride` is defined in the entity class of a unit that does not have an inheritance relationship, the operation is performed; however, the operation cannot be guaranteed.

The applicable targets are class, method, and field.

(2) Element

The following table lists the `@AttributeOverride` attributes.

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the field or property in which the mapping is overridden.
`column`	Required	This element specifies the `@Column` to be overridden.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
String

Description
This element specifies the name of the field or property in which mapping is overridden.

Default value
None

(b) column element

Type
Column

Description
This element specifies the `@Column` to be overridden.
The definition of the embeddable class or mapped superclass is applied as the mapping type.
You can specify the value within the specifiable range of `@Column`. For details, see *2.7.6 @Column*.

Default value
None

2.7.4 @AttributeOverrides

(1) Description

The annotation to be specified when you want to code multiple `@AttributeOverride` concurrently.

Applicable elements are class, method, and field.

(2) Element

The following table lists the `@AttributeOverrides` attributes:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@ElementOverride`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
`AttributeOverride[]`

Description

Attribute that specifies the array of `@AttributeOverride`.

Specifiable values are within the range of the specifiable values for `@AttributeOverride` array. For details, see *2.7.3 @AttributeOverride*.

Default value

None

2.7.5 @Basic

(1) Description

This annotation indicates the type of mapping to the simplest database column.

This annotation can be applied to the properties or instance variables of the following persistence types:

- Java primitive type
- Primitive type wrapper class
- `java.lang.String`
- `java.math.BigInteger`
- `java.math.BigDecimal`
- `java.util.Date`
- `java.util.Calendar`
- `java.sql.Date`
- `java.sql.Time`
- `java.sql.Timestamp`
- `byte[]`
- `Byte[]`
- `char[]`
- `Character[]`
- `enums`
- User--defined serialize type

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@Basic`:

Element name	Optional/Required	Element description
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
optional	Optional	This element specifies whether or not a null value can be used in the field or property. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) fetch element

Type

FetchType

Description

This element specifies the specification value of the fetch strategy.

`FetchType.EAGER` or `FetchType.LAZY` can be specified.

Furthermore, the `fetch` attribute is ignored in Cosminexus JPA provider CJPA provider, and the default `FetchType.EAGER` is usually applied. For details on the `fetch` attribute, see *6.4.5 Synchronization with the database* in the *uCosminexus Application Server Common Container Functionality Guide*.

Default value

`FetchType.EAGER`

2.7.6 @Column

(1) Description

This annotation specifies the mapping between the persistence field or persistence property, and the columns of the database.

Even when `@Column` is not specified explicitly in the persistence property or persistence field, the persistence property or persistence field is handled as if `@Column` were specified. In such a case, the default values will be applied in each element value of `@Column`.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@Column`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the column name.
`unique`	Optional	This element specifies whether or not the property is a unique key. Note that Cosminexus JPA provider does not support this attribute.
`nullable`	Optional	This element specifies whether or not a null value can be specified in the database column. Note that Cosminexus JPA provider does not support this attribute.
`insertable`	Optional	This element specifies whether or not to include the column specified by `@Column` in the INSERT statement of the SQL.
`updatable`	Optional	This element specifies whether or not to include the column specified by `@Column` in the UPDATE statement of the SQL.
`columnDefinition`	Optional	This element is used to describe the constraints added to the column in the DDL, when the CREATE statement is output. Note that Cosminexus JPA provider does not support this attribute.
`table`	Optional	This element specifies the table name that includes the column.
`length`	Optional	This element specifies the length of a column. Note that Cosminexus JPA provider does not support this attribute.
`precision`	Optional	This element specifies the accuracy of a column. This element is specified when the column is `numeric` type. Note that Cosminexus JPA provider does not support this attribute.
`scale`	Optional	This element specifies the scale of a column. This element is specified when the column is `numeric` type. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the column name.

The column name that can be specified depends on the database specifications.

Default value

Property name or field name in which this annotation is specified

(b) insertable element

Type

boolean

Description

This element specifies whether or not to include the column specified by `@Column` in the INSERT statement of the SQL. You can specify either `true` or `false`.

These values imply the following meaning:

`true`: The column specified by `@Column` is included in the INSERT statement of the SQL.

`false`: The column specified by `@Column` is not included in the INSERT statement of the SQL.

Default value

`true`

(c) updatable element

Type

boolean

Description

This element specifies whether or not to include the column specified by `@Column` in the UPDATE statement of the SQL. You can specify either `true` or `false`.

These values imply the following meaning:

`true`**:** The column specified by `@Column` is included in the UPDATE statement of the SQL.

`false`**:** The column specified by `@Column` is not included in the UPDATE statement of the SQL.

Default value

`true`

(d) table element

Type

String

Description

This element specifies the table name that includes the column.

The table name that can be specified depends on the database specifications.

Default value

Primary table name

2.7.7 @ColumnResult

(1) Description

This annotation specifies the column for mapping the query results of an SQL to the entity class

The applicable targets are the columns of `@SqlResultSetMapping`.

(2) Element

The following table lists the elements of `@ColumnResult`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name or optional name of the columns of SELECT clause.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name or optional name of the columns of SELECT clause.

The column name that can be specified depends on the database specifications.

Default value

None

2.7.8 @DiscriminatorColumn

(1) Description

This annotation specifies the column for identification used in the SINGLE_TABLE strategy or JOINED strategy. This annotation is added to an entity class that becomes a superclass by inheriting an entity class.

The applicable target is class.

(2) Element

The following table lists the elements of `@DiscriminatorColumn`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the name of the column for identification.
`discriminatorType`	Optional	This element specifies the type of the column for identification.
`columnDefinition`	Optional	This element is used to describe the constraints added to the column for identification in the DDL, when the CREATE statement is output. Note that Cosminexus JPA provider does not support this attribute.
`length`	Optional	This element specifies the length when the column for identification is a character string. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the column for identification.

The column name that can be specified depends on the database specifications.

Specify the same value (matching the upper case and lower case characters) for the `name` element as that for the `name` element of `@Column`.

Default value

`"DTYPE"`

(b) discriminatorType element

Type

DiscriminatorType

Description

This element specifies the type of the column for identification.

You can specify the following values:

- DiscriminatorType.STRING
- DiscriminatorType.CHAR
- DiscriminatorType.INTEGER

Default value

`DiscriminatorType.STRING`

2.7.9 @DiscriminatorValue

(1) Description

This annotation specifies the value of the column for identification used in the SINGLE_TABLE strategy or JOINED strategy. You can specify this annotation in a superclass or subclass.

The applicable target is class.

Note the following points:

- The settings of `@DiscriminatorValue` are not inherited. `@DiscriminatorValue` must be set up in each entity class.
- The settings of `@DiscriminatorValue` must match the type specified in `discriminatorType` and length specified in `length` of `@DiscriminatorColumn`.
- If the `discriminatorType` of `@DiscriminatorColumn` is `INTEGER`, make note of the following points:
  - In @DiscriminatorValue, specify only an integer that does not include `0` or a blank at the beginning.
  - You cannot omit `@DiscriminatorValue`. If omitted, the operation will not be guaranteed.
- If the `discriminatorType` of `@DiscriminatorColumn` is other than `INTEGER`, you can omit `@DiscriminatorValue`. In such a case, the operation is performed by assuming that the value specified in `value` is the class name of the entity.

(2) Element

The following table lists the elements of `@DiscriminatorValue`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies the value to be set up in the column for identification.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

String

Description

This element specifies the value to be set up in the column for identification.

The value that can be specified depends on the type of the column for identification.

Default value

Entity name

2.7.10 @Embeddable

(1) Description

This annotation indicates an embedded class.

An embedded class is a class that can be embedded as a field within the entity class.

The applicable target is class.

(2) Element

`@Embeddable` does not have attributes.

2.7.11 @Embedded

(1) Description

This annotation specifies the persistence property or persistence field indicating the instance value of the embedded class within the entity class.

If you want to override the column mapping declared within the embedded class, use either `@ElementOverride` or `@ElementOverrides`.

The applicable targets are method and field.

(2) Element

`@Embedded` does not have attributes.

2.7.12 @EmbeddedId

(1) Description

This annotation specifies the compound primary key of an embedded class.

This annotation is added in the persistence property or persistence field of an embeddable class owned by the entity.

When using `@EmbeddedId`, you cannot specify multiple `@EmbeddedIds` or specify `@Id` besides `@EmbeddedId`.

When you add `@Transient` to a field of the embedded class, the compound primary key will not be applicable for that field.

The applicable targets are method and field.

(2) Element

`@EmbeddedId` does not have attributes.

2.7.13 @Entity

(1) Description

This annotation specifies that the class is an entity.

The class name of the entity class does not include the package name. Note the following points during specification:

- Make sure that the entity name is a unique name within the persistence unit.
- You cannot set up the reserved characters of JPQL. If you set up the reserved characters, the operation will not be guaranteed.

The applicable target is class.

(2) Element

The following table lists the elements of `@Entity`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies a logical name for the entity class. It becomes an abstract schema name in JPQL.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
    String

Description
    This element specifies a logical name for the entity class. It becomes an abstract schema name in JPQL.
    The value that can be specified depends on the specifications of JPQL.

Default value
    Class name of the class in which `@Entity` is specified

2.7.14 @EntityListeners

(1) Description

This annotation specifies the callback listener class used in the entity class or mapped superclass.

The applicable target is class.

(2) Element

The following table lists the elements of `@EntityListeners`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies the callback listener class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
    Class[]

Description

This element specifies the callback listener class.

The value that can be specified is class.

Default value

None

2.7.15 @EntityResult

(1) Description

This annotation specifies the entity class in which the query results of the SQL are to be mapped.

The applicable target is the `entities` element of `@SqlResultSetMapping`.

(2) Element

The following table lists the elements of `@EntityResult`:

Element name	Optional/Required	Element description
`entityClass`	Required	This element specifies the result class.
`fields`	Optional	This element specifies the arrays of `@FieldResult`.
`discriminatorColumn`	Optional	This element specifies the name or optional name of the column for identification within the SELECT clause that determines the type of the entity instance.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) entityClass element

Type

Class

Description

This element specifies the result class.

The value that can be specified is the class name.

Default value

None

(b) fields element

Type

FieldResult[]

Description

This element specifies an array of `@FieldResult`.

You can specify the value within the specifiable range of the arrays of `@FieldResult`. For details, see *2.7.19 @FieldResult*.

Default value

Blank array

(c) discriminatorColumn element

Type

String

Description

This element specifies the name or optional name of the column for identification within the SELECT clause that determines the type of the entity instance.

The value that can be specified is the name or optional name of the column specified in the table.

Default value

Blank array

2.7.16 @Enumerated

(1) Description

This annotation specifies the persistence field or persistence property as the enumeration type.

This annotation can be used along with `@Basic`. You can specify `ORDINAL` (numeric type) and `STRING` (character string type) in the enumeration type.

In the following cases, `ORDINAL` (numeric type) is specified as the enumeration type:

- When the enumeration type is not specified in the `value` element
- When `@Enumerated` is not specified

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@Enumerated`:

Element name	Optional/Required	Element description
`value`	Optional	This element specifies the type used for mapping the enumeration type.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

EnumType

Description

This element specifies the type used for mapping the enumeration type.

You can specify either of the following values:

- `EnumType.ORDINAL`: Numeric type
- `EnumType.STRING`: Character string type

Default value

`EnumType.ORDINAL`

2.7.17 @ExcludeDefaultListeners

(1) Description

This annotation excludes the default listener for the following classes:

- Entity class
- Mapped superclass
- Subclass of the entity class or mapped superclass

Note that the default listener can be specified only in the XML descriptor.

The applicable target is class.

(2) Element

`@ExcludeDefaultListeners` does not have attributes.

2.7.18 @ExcludeSuperclassListeners

(1) Description

This annotation excludes the superclass listener for the following classes:

- Entity class
- Mapped superclass
- Subclass of the entity class or mapped superclass

The applicable target is class.

(2) Element

`@ExcludeSuperclassListeners` does not have attributes.

2.7.19 @FieldResult

(1) Description

This annotation specifies the field in which the query results of the SQL are to be mapped.

The applicable target is the `field` element of `@EntityResult`.

(2) Element

The following table lists the elements of `@FieldResult`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the persistence field or persistence property of the class.
`column`	Required	This element specifies the name or optional name of the column of SELECT clause.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the persistence field or persistence property of the class.

Default value

None

(b) column element

Type
: String

Description
: This element specifies the name or optional name of the column of SELECT clause.
: The column name or optional name that can be specified depends on the database specifications.

Default value
: None

2.7.20 @GeneratedValue

(1) Description

This annotation specifies the method for automatically generating and allotting a unique value to the primary key column. This annotation is applicable to the field or property of the primary key of entity class or mapped superclass containing `@Id`.

The primary key value is generated by the following four methods. Depending on the generation method selected, the base table and database sequence object must be prepared beforehand. For details on each of the generation methods, see the description about the *strategy element*.

- GenerationType.AUTO
- GenerationType.IDENTITY
- GenerationType.SEQUENCE
- GenerationType.TABLE

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@GeneratedValue`:

Element name	Optional/Required	Element description
`strategy`	Optional	This element specifies the method for generating the primary key value of the entity class.
`generator`	Optional	This element specifies the `name` element set up in `@SequenceGenerator` or `@TableGenerator` to be used.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) strategy element

Type
: GenerationType

Description
: This element specifies the method for generating the primary key value of the entity class.
: The following four types of values can be specified:

- **GenerationType.AUTO**
  For generating the primary key value, select the most appropriate procedure in each database.
  When Oracle or HiRDB is used as the database, the processing is same as `GenerationType.TABLE`.
- **GenerationType.IDENTITY**
  The primary key value is generated using the `identity` column of the database.

If Oracle is used as the database, the processing is same as `GenerationType.SEQUENCE`.

If HiRDB is used as the database, the processing is same as `GenerationType.TABLE`.

- **GenerationType.SEQUENCE**
  The primary key value is generated using the database sequence object.
  If HiRDB is used as the database, the processing is same as `GenerationType.TABLE`.
- **GenerationType.TABLE**
  The primary key value is generated using a table for maintaining the primary key value.

Default value

`GenerationType.AUTO`

(b) generator element

Type

String

Description

This element specifies the `name` element set up in `@SequenceGenerator` or `@TableGenerator` to be used.

Default value

The following names are assumed depending on the value of the strategy element:

- **In the case of GenerationType.AUTO**
  `"SEQ_GEN"`
- **In the case of GenerationType.SEQUENCE**
  `"SEQ_GEN_SEQUENCE"`
- **In the case of GenerationType.TABLE**
  `"SEQ_GEN_TABLE"`

2.7.21 @Id

(1) Description

This annotation specifies the properties or fields of the primary key of entity class.

`@Id` is applicable in the entity class or mapped superclass.

The column of the database mapped to the field or property in which `@Id` is specified is assumed as the primary key column of the primary table. When the column name of the primary key column is not specified using `@Column`, the column name of the primary key column becomes the name of the field or property in which `@Id` is specified.

Note that if `@Version` is specified in a field in which `@Id` is specified, `@Id` becomes invalid.

The applicable targets are method and field.

(2) Element

`@Id` does not have attributes.

2.7.22 @IdClass

(1) Description

This annotation specifies the compound primary key class mapped to multiple fields or properties of the entity class.

This annotation is applicable to the mapped superclass or entity class.

The name and type of the field or property of the compound primary key class must match with that of the field or property of the primary key of entity class. The name and type specified in this annotation must correspond to the name and type of the property or field of primary key of the entity in which `@Id` is added.

The applicable target is class.

(2) Element

The following table lists the elements of `@IdClass`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies the compound primary key class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
  Class

Description
  This element specifies the compound primary key class.
  The value that can be specified is the class name.

Default value
  None

2.7.23 @Inheritance

(1) Description

This annotation specifies the inheritance mapping strategy used in the inheritance hierarchy of an entity.

`@Inheritance` is specified in the parent entity class of inheritance hierarchy.

The following are two types of inheritance mapping strategy available with Cosminexus JPA provider:

- **SINGLE_TABLE** (single table for each class hierarchy)
- **JOINED (**binding subclass strategy**)**

For details on the inheritance mapping strategy, see *6.13.2 Inheritance mapping strategy* in the *uCosminexus Application Server Common Container Functionality Guide*.

The applicable target is class.

(2) Element

The following table lists the elements of `@Inheritance`:

Element name	Optional/Required	Element description
`strategy`	Optional	This element specifies the type of inheritance mapping strategy.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) strategy element

Type
  InheritanceType

Description

This element specifies the type of inheritance mapping strategy used in an entity.

The following two types of values can be specified:

- `InheritanceType.SINGLE_TABLE`: This strategy is used to map all classes in the inheritance hierarchy to a single table.
- `InheritanceType.JOINED`: This strategy is used to map the top most (parent class) of the inheritance hierarchy to a single table, and map each subclass with a subclass--specific mapping.

Default value

`InheritanceType.SINGLE_TABLE`

2.7.24 @JoinColumn

(1) Description

This annotation specifies the external key column for the binding table, or the column name of the binding--destination table that is referenced from the external key column, when the entity classes are correlated. Always specify the column that acts as the primary key of the binding--destination table.

When multiple external key columns exist, use `@JoinColumns`, and specify `@JoinColumn` for each relation. When multiple `@JoinColumns` are specified, specify the `name` element and `referencedColumnName` element in each annotation.

When `@JoinColumn` is not specified explicitly, it is assumed that a single external key column is specified in the persistence property or persistence field that specifies the relation. Also, the default value is applied to each element value of `@JoinColumn`.

Furthermore, if the changes made in a single column of the field and the changes made by correlating the cascade operation are performed concurrently, consistency might not be achieved. Therefore, when the column specified in the `name` element and the `referencedColumnName` element is defined in a field of the entity, the `insertable` element and the `updatable` element must be set to `false`. With this, only the changes made in the field will be applied to the database, but the changes made due to the correlation of the cascade operation will not be applied to the database.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@JoinColumn`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the external key column name used to bind the target tables.
`referencedColumnName`	Optional	This element specifies the column name of the binding--destination table that is referenced from the external key column specified in the `name` element.
`unique`	Optional	This element specifies whether or not the property is a unique key. Note that Cosminexus JPA provider does not support this attribute.
`nullable`	Optional	This element specifies whether or not a null value can be specified in the database column. Note that Cosminexus JPA provider does not support this attribute.
`insertable`	Optional	This element specifies whether or not to include the column specified by `@JoinColumn` in the INSERT statement of the SQL.
`updatable`	Optional	This element specifies whether or not to include the column specified by `@JoinColumn` in the UPDATE statement of the SQL.
`columnDefinition`	Optional	This element is used to describe the constraints added to the external column in the DDL, when the CREATE statement is output.

Element name	Optional/Required	Element description
`columnDefinition`	Optional	Note that Cosminexus JPA provider does not support this attribute.
`table`	Optional	This element specifies the table name that includes the external key column.

The details of attributes that are supported in Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the external key column name used to bind the target tables.

The location of existence of the external key column is different for each type of the entity relationship. The location of existence of the external key column for each type of the entity relationship is as follows:

- **In the case of OneToOne relationship or ManyToOne relationship**
  Within the local entity table
- **In the case of ManyToMany relationship**
  Within the binding table of `@JoinTable`

The values that can be specified depend on the specifications of the database column name.

Default value

- **When a single external key column is specified in the local entity, and nothing is specified in the value of the name element**
  name-of-the-related-property-or-field-within-the-local-entity_name--of-the-referenced--primary--key--column
- **When the related property and field that is being referenced does not exist (example: when @JoinTable is used)**
  name-of-the-referenced-entity_name-of-the-referenced-primary-key-column

(b) referencedColumnName element

Type

String

Description

This element specifies the column name of the binding--destination table that is referenced by the external key column specified in the `name` element.

The column name of the binding--destination table exists at the following locations:

- **When the relationship annotation is used**
  Within the referenced table
- **When @JoinTable is used**
  Within the entity table of the owner entity

Note

When binding is defined as a part of reverse binding, the location will be within the table of the non--owner entity class.

The column names that can be specified depend on the database specifications.

Default value

Column name of the primary key of the table referenced from the external key

Note

If a single external key column is specified, the default value will be applied.

(c) insertable element

Type

boolean

Description

This element specifies whether or not to include the column specified by `@JoinColumn` in the INSERT statement of the SQL. You can specify either `true` or `false`.

These values imply the following meaning:

`true`: The column specified by `@JoinColumn` is included in the INSERT statement of the SQL.

`false`: The column specified by `@JoinColumn` is not included in the INSERT statement of the SQL.

Default value

`true`

(d) updatable element

Type

boolean

Description

This element specifies whether or not to include the column specified by `@JoinColumn` in the UPDATE statement of the SQL. You can specify either `true` or `false`.

These values imply the following meaning:

`true`: The column specified by `@JoinColumn` is included in the UPDATE statement of the SQL.

`false`: The column specified by `@JoinColumn` is not included in the UPDATE statement of the SQL.

Default value

`true`

(e) table element

Type

String

Description

This element specifies the table name that includes the external key column.

The table name that can be specified depends on the database specifications.

Default value

Primary table name

2.7.25 @JoinColumns

(1) Description

This annotation is specified when multiple `@JoinColumns` that indicate the same relationship are coded concurrently. `@JoinColumns` defines the mapping of the compound external key.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@JoinColumns`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@JoinColumn`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

JoinColumn[]

Description

This element specifies an array of `@JoinColumn`.

The values can be specified within the specifiable range of the arrays of `@JoinColumn`.

Default value

None

2.7.26 @JoinTable

(1) Description

This annotation specifies the binding table set up in the following classes:

- Owner class when the `ManyToMany` relationship is specified
- Class containing a single-direction `OneToMany` relationship

When the `name` element is not specified, the name of the binding table becomes as follows:

owner-table--name_non--owner-table-name

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@JoinTable`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the name of the binding table.
`catalog`	Optional	This element specifies the catalog name of the binding table. Note that Cosminexus JPA provider does not support this attribute.
`schema`	Optional	This element specifies the schema name of the binding table.
`joinColumns`	Optional	This element specifies the external key column of the binding table that references the primary table of the owner entity.
`inverseJoinColumns`	Optional	This element specifies the external key column of the binding table that references the primary table of the non--owner entity.
`uniqueConstraints`	Optional	This element specifies the unique constraints of the table. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the binding table.

The table name that can be specified depends on the database specifications.

Default value

owner-table-name_non-owner-table-name

(b) schema element

Type

String

Description

This element specifies the schema name of the table.

The value that can be specified depends on the specifications of the database schema name.

Default value

Default schema of the database used

(c) joinColumns element

Type

JoinColumn[]

Description

This element specifies the external key column of the binding table that references the primary table of the owner entity. This element specifies an array of `@JoinColumn`. The external key column name of the binding table is specified in the `name` element, while the referenced column name of the owner is specified in the `referencedColumnName` element of `@JoinColumn`.

The column names that can be specified depend on the database specifications.

Default value

External key of `@JoinColumn`

(d) inverseJoinColumns element

Type

JoinColumn[]

Description

This element specifies the external key column of the binding table that references the primary table of the non--owner entity. This element specifies an array of `@JoinColumn`. The external key column name of the binding table is specified in the `name` element, while the column of the binding--destination table that is referenced by the external key column is specified in the `referencedColumnName` element of `@JoinColumn`.

The values that can be specified depend on the specifications of the database column name.

Default value

External key column of `@JoinColumn`

2.7.27 @Lob

(1) Description

This annotation specifies the persistence field or persistence property of the `large` object type supported by the database. This annotation can be used together with `@Basic`.

`@Lob` contains the binary type (`Blob`) and character type (`Clob`). The type of `@Lob` is determined based on the type of the persistence field or persistence property. For a character string and character type, the type of `@Lob` is `Clob`, and in other cases, the type is `Blob`.

The applicable targets are method and field.

(2) Element

`@Lob` does not have attributes.

2.7.28 @ManyToMany

(1) Description

This annotation specifies the multiple relationships from an owner entity class having a `ManyToMany` relationship to a non-owner entity class.

The `ManyToMany` relationship includes the owner and non-owner, irrespective of bi-direction or single direction. If the relationship is bi-directional, the binding table can be specified in any direction.

If the `Collection` element type is specified using `Generics`, the non-owner entity class is not required to be specified. In other cases, make sure to specify it.

Furthermore, when you specify `@ManyToMany`, note the settings of the following annotations:

- The elements of the same annotations for `@OneToMany` are same as that of `@ManyToMany`.
- If the properties or fields in which `@ManyToMany` is defined have the same name in the owner and non-owner classes, do not use the default settings (when the `joinColumns` element and `inverseJoinColumns` element is not specified) of `@JoinTable`.
- For the bi--directional relationship, the value of the binding table is updated based on the information of the owner. Even if the mapping information is changed in the entity class in which the `mappedBy` element is specified, the information will not be applied in the binding table.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@ManyToMany`:

Element name	Optional/Required	Element description
`targetEntity`	Optional	This element specifies the non--owner entity class.
`cascade`	Optional	This element specifies the operations to be cascaded.
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
`mappedBy`	Optional	This element specifies the name of the field or property that maintains a relationship in the owner entity class, when added in the elements of the non--owner entity class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) targetEntity element

Type

Class

Description

This element specifies the non-owner entity class.

The specification of this element is optional when the collection property is defined using `Generics`. In other cases, you must always specify this element.

Default value

The type in which the collection contains parameters

# Set up only when the collection property is defined using `Generics`.

(b) cascade element

Type

CascadeType[]

Description

This element specifies the operations to be cascaded.

The following table describes the specifiable values:

- `CascadeType.ALL`**: The** persist, remove, merge, and refresh operations of the owner entity class are cascaded to the related destination.
- `CascadeType.MERGE`**:** The merge operation of the owner entity class is cascaded to the related destination.
- `CascadeType.PERSIST`**:** The persist operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REFRESH`**:** The refresh operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REMOVE`**:** The remove operation of the owner entity class is cascaded to the related destination.

Default value

Not to be cascaded

(c) fetch element

Type

FetchType

Description

This attribute defines the fetch strategy of data from the database. For details on the fetch strategy, see *6.4.5 Synchronization with the database* in the *uCosminexus Application Server Common Container Functionality Guide*.

The following two types of values can be specified:

- `EAGER` **strategy:** Requests in which the data must be fetched eagerly
- `LAZY` **strategy:** Requests in which data is fetched lazily when accessed for the first time

Default value

`FetchType.LAZY`

(d) mappedBy element

Type

String

Description

This element specifies the name of the field or property that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class.

When this element is specified, the relationship becomes bi-directional. For a bi-directional relationship, the value of the binding table is updated based on the information of the owner. Even when the mapping information is changed in the non--owner entity class (the entity class in which the `mappedBy` element is specified), the information will not be applied in the binding table.

Default value

None

2.7.29 @ManyToOne

(1) Description

This annotation indicates that the class in which `@ManyToOne` is specified has a `ManyToOne` relationship, and also specifies the relationship from the owner entity class to the non--owner entity class.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@ManyToOne`:

Element name	Optional/Required	Element description
`targetEntity`	Optional	This element specifies the non-owner entity class.
`cascade`	Optional	This element specifies the operations to be cascaded.
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
`optional`	Optional	This element specifies whether or not a null value can be set up for all non--primitive type fields and properties.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) targetEntity element

Type

Class

Description

This element specifies the non--owner entity class.

Default value

Type of the field and property in which the annotation is added

(b) cascade element

Type

CascadeType[]

Description

This element specifies the operations to be cascaded.

The following table describes the specifiable values:

- `CascadeType.ALL`**:** The persist, remove, merge, and refresh operations of the owner entity class are cascaded to the related destination.
- `CascadeType.MERGE`**:** The merge operation of the owner entity class is cascaded to the related destination.
- `CascadeType.PERSIST`**:** The persist operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REFRESH`**:** The refresh operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REMOVE`**:** The remove operation of the owner entity class is cascaded to the related destination.

Default value

Not to be cascaded

(c) fetch element

Type

FetchType

Description

This attribute defines the fetch strategy of data from the database. For details on the fetch strategy, see *6.4.5 Synchronization with the database* in the *uCosminexus Application Server Common Container Functionality Guide*.

The following two types of values can be specified:

- `EAGER` **strategy:** Requests in which the data must be fetched eagerly

- `LAZY` **strategy:** Requests in which data is fetched lazily when accessed for the first time

Default value
`FetchType.EAGER`

(d) optional element

Type
boolean

Description
This element specifies whether or not a null value can be set up for all non-primitive type fields and properties. The following values can be specified:

- `true`**:** A null value can be set up for all non-primitive type fields and properties.
- `false`**:** A null value cannot be specified for all non-primitive type fields and properties.

Default value
`true`

2.7.30 @MapKey

(1) Description

This annotation specifies the map key used for object identification within the map when the non--owner entity class is indicated by the `java.util.Map` type, in the `OneToMany` relationship or `ManyToMany` relationship.

When the `name` element is not specified, the primary key of the correlated entity is used as the map key.

If mapping is done as `@IdClass` when the primary key is a compound primary key, the compound primary key is used as the map key.

If a persistence field or persistence property other than the primary key is used as the map key, the unique key constraints related to the map key can be included.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@MapKey`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the name of the persistence field or persistence property of the non-owner entity class that is used as the map key.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
String

Description
This element specifies the name of the persistence field or persistence property of the non-owner entity class that is used as the map key.

Default value
Name of the primary key field or property of the non-owner entity class

2.7.31 @MappedSuperclass

(1) Description

This annotation specifies a mapped superclass.

A mapped superclass is used for inheritance. Hence, there are no tables corresponding to this class. Except for mapping to a subclass, and the inheritance of the related mapping information, the mapped superclass is mapped to the table in the same way as an entity.

The mapping information can be overridden in the subclass using `@ElementOverride`.

The applicable target is class.

(2) Element

`@MappedSuperclass` does not have attributes.

2.7.32 @NamedNativeQueries

(1) Description

This annotation is specified when multiple `@NamedNativeQueries` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@NamedNativeQueries`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@NamedNativeQuery`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
    NamedNativeQuery[]

Description
    This element specifies an array of `@NamedNativeQuery`.
    The values can be specified within the specifiable range of the arrays of `@NamedNativeQuery`. For details, see *2.7.33 @NamedNativeQuery*.

Default value
    None

2.7.33 @NamedNativeQuery

(1) Description

This annotation specifies a named query in the SQL. This annotation can be applied to an entity class and mapped superclass.

The applicable target is class.

(2) Element

The following table lists the elements of `@NamedNativeQuery`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the named query.
`query`	Required	This element specifies the SQL string.
`hints`	Optional	This element specifies an array of `@QueryHint`.
`resultClass`	Optional	This element specifies the class in which the SQL results are applied.
`resultSetMapping`	Optional	This element specifies the name indicated in the `name` element of `@SqlResultSetMapping`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
String

Description
This element specifies the name of the named query.
The value that can be specified is a character string.

Default value
None

(b) query element

Type
String

Description
This element specifies the SQL string.
The SQL that can be specified depends on the specifications of the database used.

Default value
None

(c) hints element

Type
QueryHint[]

Description
This element specifies an array of `@QueryHint`.
You can specify the value within the specifiable range of the arrays of `@QueryHint`. For details, see *2.7.53 @QueryHint*.

Default value
Blank array

(d) resultClass element

Type
Class

Description
This element specifies the class in which the SQL results are applied.

The `resultClass` element is specified when the class, in which you want to map the execution results of the query, exists. Do not specify the `resultClass` element and `resultSetMapping` element concurrently.

The value that can be specified is the class name.

Default value

`void.class`

(e) resultSetMapping element

Type

String

Description

This element specifies the name indicated in the `name` element of `@SqlResultSetMapping` in which the result set is defined.

This element is specified when the SQL results are to be mapped to any result set.

Do not specify the `resultClass` element and `resultSetMapping` element concurrently.

You can specify the value within the specifiable range of the `name` element of `@SqlResultSetMapping`. For details, see *2.7.57(2)(a) name element*.

Default value

Null character string

2.7.34 @NamedQueries

(1) Description

This annotation is specified when multiple `@NamedQueries` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@NamedQueries`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@NamedQuery`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

NamedQuery[]

Description

This element specifies an array of `@NamedQuery`.

You can specify the value within the specifiable range of the arrays of `@NamedQuery`. For details, see *2.7.35 @NamedQuery*.

Default value

None

2.7.35 @NamedQuery

(1) Description

This annotation specifies a named query of JPQL. This annotation can be applied to an entity class and mapped superclass.

The applicable target is class.

(2) Element

The following table lists the elements of `@NamedQuery`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the named query.
`query`	Required	This element specifies the query string of JPQL.
`hints`	Optional	This element specifies an array of `@QueryHint`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the named query.

The value that can be specified is a character string.

Default value

None

(b) query element

Type

String

Description

This element specifies the query string of JPQL.

The value that can be specified depends on the specifications of JPQL.

Default value

None

(c) hints element

Type

QueryHint[]

Description

This element specifies an array of `@QueryHint`.

You can specify the value within the specifiable range of the arrays of `@QueryHint`. For details, see *2.7.53 @QueryHint*.

Default value

Blank array

2.7.36 @OneToMany

(1) Description

This annotation specifies the multiple relationships from an owner entity class having the `OneToMany` relationship to a non-owner entity class.

The elements of the same annotations for `@OneToMany` are same as that of `@ManyToMany`.

If the `Collection` element type is specified using `Generics`, the non-owner entity class is not required to be specified. In other cases, make sure to specify it.

Furthermore, to achieve a bi-directional relationship, always specify the `mappedBy` element at the non-owner side.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@OneToMany`:

Element name	Optional/Required	Element description
`targetEntity`	Optional	This element specifies the non-owner entity class.
`cascade`	Optional	This element specifies the operations to be cascaded.
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
`mappedBy`	Optional	This element specifies the name of the field or property that maintains a relationship in the owner entity class, when added in the elements of the non--owner entity class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) targetEntity element

Type

Class

Description

This element specifies the non-owner entity class.

The specification of this element is optional when a collection property is defined using `Generics`. In other cases, you must always specify this element.

Default value

The type in which the collection contains parameters

# Set up only when the collection property is defined using `Generics`.

(b) cascade element

Type

CascadeType[]

Description

This element specifies the operations to be cascaded.

The following table describes the specifiable values:

- `CascadeType.ALL`: The persist, remove, merge, and refresh operations of the owner entity class are cascaded to the related destination.
- `CascadeType.MERGE`: The merge operation of the owner entity class is cascaded to the related destination.
- `CascadeType.PERSIST`: The persist operation of the owner entity class is cascaded to the related destination.

- `CascadeType.REFRESH`**:** The refresh operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REMOVE`**:** The remove operation of the owner entity class is cascaded to the related destination.

Default value
Not to be cascaded

(c) fetch element

Type
FetchType

Description
This attribute defines the fetch strategy of data from database. For details on the fetch strategy, see *>6.4.5 Synchronization with the database* in the *uCosminexus Application Server Common Container Functionality Guide*.

The following two types of values can be specified:

- `EAGER` **strategy:** Requests in which the data must be fetched eagerly
- `LAZY` **strategy:** Requests in which data is fetched lazily when accessed for the first time

Default value
`FetchType.LAZY`

(d) mappedBy element

Type
String

Description
This element specifies the name of the field or property that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class.

When this element is specified, the relationship becomes bi-directional.

Default value
None

2.7.37 @OneToOne

(1) Description

This annotation indicates that the specified class has `OneToOne` relationship, and also specifies the single relationship between entity classes.

Furthermore, to achieve a bi-directional relationship, always specify the `mappedBy` element at the non--owner side.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@OneToOne`:

Element name	Optional/Required	Element description
`targetEntity`	Optional	This element specifies the non--owner entity class.
`cascade`	Optional	This element specifies the operations to be cascaded.
`fetch`	Optional	This element specifies the specification value of the fetch strategy.

Element name	Optional/Required	Element description
`optional`	Optional	This element specifies whether or not a null value can be set up for all non-primitive type fields and properties.
`mappedBy`	Optional	This element specifies the name of the field that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) targetEntity element

Type

Class

Description

This element specifies the non--owner entity class.

Default value

Type of the field and property in which the annotation is added

(b) cascade element

Type

CascadeType[]

Description

This element specifies the operations to be cascaded.

The following table describes the specifiable values:

- `CascadeType.ALL`**:** The persist, remove, merge, and refresh operations of the owner entity class are cascaded to the related destination.
- `CascadeType.MERGE`**:** The merge operation of the owner entity class is cascaded to the related destination.
- `CascadeType.PERSIST`**:** The persist operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REFRESH`**:** The refresh operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REMOVE`**:** The remove operation of the owner entity class is cascaded to the related destination.

Default value

Not to be cascaded

(c) fetch element

Type

FetchType

Description

This attribute defines the fetch strategy of data from the database. For details on the fetch strategy, see *6.4.5 Synchronization with the database* in the *uCosminexus Application Server Common Container Functionality Guide*.

The following two types of values can be specified:

- `EAGER` **strategy:** Requests in which the data must be fetched eagerly
- `LAZY` **strategy:** Requests in which data is fetched lazily when accessed for the first time

Default value

`FetchType.EAGER`

(d) optional element

Type

boolean

Description

This element specifies whether or not a null value can be set up for all non-primitive type fields and properties. The following values can be specified:

- `true`**: A null** value can be set up for all non-primitive type fields and properties.
- `false`**:** A null value cannot be specified for all non-primitive type fields and properties.

Default value

`true`

(e) mappedBy element

Type

String

Description

This element specifies the name of the field that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class. When this element is specified, the relationship becomes bi-directional.

Default value

None

2.7.38 @OrderBy

(1) Description

This annotation specifies the order in which the information is maintained in the collection, when the entity information is acquired.

The applicable targets are method and field.

(2) Element

The following table lists the elements `@OrderBy`:

Element name	Optional/Required	Element description
`value`	Optional	This element is specified when the entities are to be acquired in an order based on the fields or properties other than the primary key.

The details of attribute for mapping with Cosminexus JPA provider are as follows:

(a) value element

Type

String

Description

This element is specified when the entities are to be acquired in an order based on the fields or properties other than the primary key. The fields or properties for which the order is to be specified are demarcated by comma (,).

The order of collection is specified after the fields or properties. The following values can be specified. If the order is not specified, the ascending order is assumed.

- `ASC`**:** Ascending order
- `DESC`**:** Descending order

In the fields or properties specified in the `value` element, specify the column that stores the values for which you can perform the comparative calculation.

Default value

Ascending order based on the primary key of the entity class

2.7.39 @PersistenceContext

(1) Description

This annotation defines the reference of a container-managed EntityManager. This annotation is added to the class to be looked up.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@PersistenceContext`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the lookup name of the EntityManager.
`unitName`	Optional	This element specifies the name of the persistence unit defined in the `persistence.xml` file.
`type`	Optional	This element specifies the type of lifecycle of the persistence context.
`properties`	Optional	This element specifies the vendor--dependent properties specified in `@PersistenceProperty`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the lookup name of the EntityManager.

You are not required to specify this element when using a DI.

Default value

Null character string

(b) unitName element

Type

String

Description

This element specifies the name of the persistence unit defined in the `persistence.xml` file.

When the `unitName` element is specified, set the same name for the persistence unit used by EntityManagerFactory that can be accessed by the JNDI name space.

Default value

Null character string

(c) type element

Type

PersistenceContextType

Description

This element specifies the type of lifecycle of the persistence context.

The following two types of values can be specified:

- `TRANSACTION`: Persistence context of the transaction scope
- `EXTENDED`: Extended persistence context

Default value

`TRANSACTION`

(d) properties element

Type

PersistenceProperty[]

Description

This element specifies the vendor-dependent properties of the JPA Provider specified in `@PersistenceProperty`.

You can specify the value within the specifiable range of the arrays of `@PersistenceProperty`. For details, see *2.7.41 @PersistenceProperty*.

When the `properties` element is specified, the properties that cannot be recognized are ignored.

Default value

Blank array

2.7.40 @PersistenceContexts

(1) Description

This annotation is specified when multiple `@PersistenceContexts` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@PersistenceContexts`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@PersistenceContext`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

PersistenceContext[]

Description

This element specifies an array of `@PersistenceContext`.

You can specify the value within the specifiable range of the arrays of `@PersistenceContext`. For details, see *2.7.39 @PersistenceContext*.

Default value

None

2.7.41 @PersistenceProperty

(1) Description

This annotation sets up properties in the container-managed EntityManager .

Currently, no properties can be used.

The applicable target is the `properties` element of `@PersistenceContext`.

(2) Element

The following table lists the elements of `@PersistenceProperty`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the property.
`value`	Required	This element specifies the value of the property.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the property.

Default value

None

(b) value element

Type

String

Description

This element specifies the value of the property.
You can specify the value depends on the specifications of the properties specified in the `name` element.

Default value

None

2.7.42 @PersistenceUnit

(1) Description

This annotation defines the reference of the EntityManagerFactory. This annotation is added to the class to be looked up.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@PersistenceUnit`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the lookup name of the EntityManagerFactory.

Element name	Optional/Required	Element description
`unitName`	Optional	This element specifies the name of the persistence unit defined in the `persistence.xml` file.

The details of attributes supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the lookup name of the EntityManagerFactory. This element specifies the name of the EntityManagerFactory to be registered in the JNDI name space.

The value that can be specified is a character string.

You are not required to specify this element when using a DI.

Default value

Null character string

(b) unitName element

Type

String

Description

This element specifies the name of the persistence unit defined in the `persistence.xml` file.

When the `unitName` element is specified, set the same name for the persistence unit used by EntityManagerFactory that can be accessed by the JNDI name space.

Default value

Null character string

2.7.43 @PersistenceUnits

(1) Description

This annotation is specified when multiple `@PersistenceUnits` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@PersistenceUnits`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies the arrays of `@PersistenceUnit`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

PersistenceUnit[]

Description

This element specifies the arrays of `@PersistenceUnit`.

You can specify the value within the specifiable range of the arrays of `@PersistenceUnit`. For details, see *2.7.42 @PersistenceUnit*.

Default value

None

2.7.44 @PostLoad

(1) Description

This annotation indicates the callback method invoked after an entity is read from the cache or after the SELECT statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PostLoad` does not have attributes.

2.7.45 @PostPersist

(1) Description

This annotation indicates the callback method invoked after the INSERT statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PostPersist` does not have attributes.

2.7.46 @PostRemove

(1) Description

This annotation indicates the callback method invoked after the DELETE statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PostRemove` does not have attributes.

2.7.47 @PostUpdate

(1) Description

This annotation indicates the callback method invoked after the UPDATE statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PostUpdate` does not have attributes.

2.7.48 @PrePersist

(1) Description

This annotation indicates the callback method invoked before the INSERT statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PrePersist` does not have attributes.

2.7.49 @PreRemove

(1) Description

This annotation indicates the callback method invoked before the DELETE statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PreRemove` does not have attributes.

2.7.50 @PreUpdate

(1) Description

This annotation indicates the callback method invoked before the UPDATE statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PreUpdate` does not have attributes.

2.7.51 @PrimaryKeyJoinColumn

(1) Description

This annotation specifies the column used as the external key when binding with another table. This annotation is used in the following cases:

- When the names of the primary key of the superclass and the primary key of the subclass of an entity are different in the JOINED strategy of the inheritance mapping strategy
- When the primary table and secondary table are to be bound in `@SecondaryTable`#
- When the primary key of the non-owner entity class is used as an external key in the `OneToOne` relationship

#

Here, this annotation is used within `@SecondaryTable`.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@PrimaryKeyJoinColumn`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the column name for binding the target tables.
`referencedColumnName`	Optional	This element specifies the column name of the primary key of binding--destination table that is referenced by the column specified in the `name` element.
`columnDefinition`	Optional	This element is used to describe the constraints added to the column in the DDL, when the CREATE statement is output. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the column name for binding the target tables.

The column name that can be specified depends on the database specifications.

Default value

- **When the JOINED strategy is used**
  Column name of the primary key of primary table of the superclass
- **When @SecondaryTable is used**
  Column name of the primary key of primary table
- **When the OneToOne relationship is used**
  Column name of the primary key of target entity table

(b) referencedColumnName element

Type

String

Description

This element specifies the column name of the primary key of binding-destination table that is referenced by the column specified in the `name` element. Specify the same value as the character string of the name element of `@Column`. Arrange the upper case and lower case characters in the character string to be specified.

The column name that can be specified depends on the database specifications.

Even when the unique key constraints are used instead of specifying the primary key in the column in the `OneToOne` relationship, the operation will continue, but will not be guaranteed.

Default value

- **When the JOINED strategy is used**
  Column name of the primary key of primary table of the superclass
- **When @SecondaryTable is used**
  Column name of the primary key of primary table
- **When the OneToOne relationship is used**
  Column name of the primary key of target entity table

2.7.52 @PrimaryKeyJoinColumns

(1) Description

This annotation is specified when multiple `@PrimaryKeyJoinColumns` are coded concurrently.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@PrimaryKeyJoinColumns`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@PrimaryKeyJoinColumn`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

PrimaryKeyJoinColumn[]

Description

This element specifies an array of `@PrimaryKeyJoinColumn`.

You can specify the value within the specifiable range of `@PrimaryKeyJoinColumn`. For details, see *2.7.51 @PrimaryKeyJoinColumn*.

Default value

None

2.7.53 @QueryHint

(1) Description

This annotation specifies a database--specific query hint.

You can set up a pessimistic lock and the cache functionality of the entity.

The applicable target is the `hints` element of `@NamedQuery` or `@NamedNativeQuery`.

(2) Element

The following table lists the elements of `@QueryHint`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the hint.
`value`	Required	This element specifies the value of the hint.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the hint to be used. The following value can be specified:

cosminexus.jpa.pessimistic-lock

This is the name of the hint that specifies whether or not to use a pessimistic lock.

Default value

None

(b) value element

Type

String

Description

This element specifies the value of the hint. The following values are specified based on the name of the hint specified in the `name` element:

Specification value when `cosminexus.jpa.pessimistic-lock` is specified in the `name` element

- `NoLock`: Specified when the pessimistic lock is not used.
- `Lock`: Specified when the pessimistic lock is used. If the target table is already locked, unlocking is awaited. The SQLs issued at this point are specified as follows, for each used database:
  In Oracle: `SELECT.... FOR UPDATE`
  In HiRDB: `SELECT....WITH EXCLUSIVE LOCK`
- LockNoWait: Specified when the pessimistic lock is used. If the target table is already locked, an exception occurs. The SQLs issued at this point are specified as follows, for each used database:
  In Oracle: `SELECT.... FOR UPDATE NO WAIT`
  In HiRDB: `SELECT....WITH EXCLUSIVE LOCK NO WAIT`

Default value

When `cosminexus.jpa.pessimistic-lock` is specified in the `name` element

NoLock

2.7.54 @SecondaryTable

(1) Description

This annotation specifies the secondary table in the entity class.

This annotation is specified when the entity class is mapped in multiple tables of the database.

When `@SecondaryTable` is not specified within the entity class, all persistence properties or persistence fields of the entity class will be mapped to the tables specified in the primary table.

The applicable target is class.

(2) Element

The following table lists the elements of `@SecondaryTable`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the secondary table name.
`catalog`	Optional	This element specifies the catalog name of the secondary table. Note that Cosminexus JPA provider does not support this attribute.
`schema`	Optional	This element specifies the schema name of the secondary table.
`pkJoinColumns`	Optional	This element specifies the external key column used to bind the secondary table to the primary table.
`uniqueConstraint s`	Optional	This element specifies the unique key constraints in the table.

Element name	Optional/Required	Element description
`uniqueConstraints`	Optional	Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
String

Description
This element specifies the secondary table name.
The table name that can be specified depends on the database specifications.

Default value
None

(b) schema element

Type
String

Description
This element specifies the schema name of the secondary table.
The schema name that can be specified depends on the database specifications.

Default value
Default schema name of the database used

(c) pkJoinColumns element

Type
PrimaryKeyJoinColumn[]

Description
This element specifies the external key column of the secondary table. This annotation is specified in the arrays of `@PrimaryKeyJoinColumn`.
When this element is not specified, the external key column of the secondary table has the same name and type as the primary key column of the primary table. Therefore, the secondary table references the primary key column of the primary table.

Default value
You can specify the value within the specifiable range of the arrays of `@PrimaryKeyJoinColumn`. For details, see *2.7.51 @PrimaryKeyJoinColumn*.

2.7.55 @SecondaryTables

(1) Description

This annotation is specified when multiple `@SecondaryTables` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@SecondaryTables`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@SecondaryTable`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

SecondaryTable[]

Description

This element specifies an array of `@SecondaryTable`.

You can specify the value within the specifiable range of the arrays of `@SecondaryTable`. For details, see *2.7.54 @SecondaryTable*.

Default value

None

2.7.56 @SequenceGenerator

(1) Description

This annotation specifies the settings of the sequence generator that creates the primary key. The following settings are required when using `@SequenceGenerator`:

- Specify `GenerationType.SEQUENCE` in the `strategy` element of `@GeneratedValue`.
- Set up the name specified in the `generator` element of `@GeneratedValue` to the `name` element of `@SequenceGenerator`.

The sequence generator is specified in the fields or properties of the entity class or primary key. The scope of the sequence generator name is enabled in the persistence unit.

When creating a sequence object, specify a positive integer in the increment interval (INCREMENT BY) between sequential numbers, and in the initial value (START WITH) of the generated sequential number. When `1` is specified in the initial value (START WITH), the primary key is generated from `1`. The operation will not be guaranteed if a negative value is specified.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@SequenceGenerator`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name specified in the `generator` element of the `@GeneratedValue` annotation.
`sequenceName`	Optional	This element specifies the name of the database sequence object for acquiring an existing primary key value, or an already defined primary key value.
`initialValue`	Optional	This element specifies the initial value when the generation of the primary key value by the sequence object is started. Note that Cosminexus JPA provider does not support this attribute. Ignore when the value is specified.
`allocationSize`	Optional	This element specifies the size of allocating the primary key value from the sequence.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name specified in the `generator` element of the `@GeneratedValue` annotation.

The value that can be specified is a character string.

Default value

None

(b) sequenceName element

Type

String

Description

This element specifies the name of the database sequence object for acquiring an existing primary key value, or an already defined primary key value.

The sequence object name that can be specified depends on the database specifications.

Default value

Specified value of the `generator` element of `@GeneratedValue`

(c) allocationSize element

Type

int

Description

This element specifies the allocation size of the primary key value from the sequence. The sequence object name that can be specified depends on the database specifications.

The size that can be specified is a numeric value that is at least one more than the `int` type. Specify a value same as the increment interval of the sequence object. The operation will not be guaranteed if you specify a different value.

Note that in this element, you can specify the maximum value used during execution. If you specify a large value for acquiring the management area of the sequence number, the `java.lang.OutOfMemoryError` exception will occur during the execution.

Default value

`50`

2.7.57 @SqlResultSetMapping

(1) Description

This annotation specifies the result set mapping of an SQL query.

The applicable target is class.

(2) Element

The following table lists the elements of `@SqlResultSetMapping`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the result set mapping.
`entities`	Optional	This element specifies an array of `@EntityResult`.

Element name	Optional/Required	Element description
`columns`	Optional	This element specifies an array of `@ColumnResult`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the result set mapping.

The value that can be specified is a character string.

Default value

None

(b) entities element

Type

EntityResult[]

Description

This element specifies an array of `@EntityResult`.

You can specify the value within the specifiable range of the arrays of `@EntityResult`. For details, see *2.7.15 @EntityResult*.

Default value

Blank array

(c) columns element

Type

ColumnResult[]

Description

This element specifies an array of `@ColumnResult`.

You can specify the value within the specifiable range of the arrays of `@ColumnResult`. For details, see *2.7.7 @ColumnResult*.

Default value

Blank array

2.7.58 @SqlResultSetMappings

(1) Description

This annotation is specified when multiple `@SqlResultSetMappings` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@SqlResultSetMappings`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@SqlResultSetMapping`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

SqlResultSetMapping[]

Description

This element specifies an array of `@SqlResultSetMapping`.

You can specify the value within the specifiable range of the arrays of `@SqlResultSetMapping`. For details, see *2.7.57 @SqlResultSetMapping*.

Default value

None

2.7.59 @Table

(1) Description

This annotation specifies the primary table in the entity class.

Even when `@Table` is not specified explicitly in the entity class, the entity class is handled as if `@Table` were specified. In such a case, the default value will be applied in each element of `@Table`.

If more than one table is specified for mapping the entities, use either `@SecondaryTable` or `@SecondaryTables`.

The applicable target is class.

(2) Element

The following table lists the elements of `@Table`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the table name.
`catalog`	Optional	This element specifies the catalog name of the table. Note that Cosminexus JPA provider does not support this attribute.
`schema`	Optional	This element specifies the schema name of the table.
`uniqueConstraints`	Optional	This element specifies the unique key constraints in the table. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the table name.

The table name that can be specified depends on the database specifications.

Default value

Entity name

(b) schema element

Type

String

Description

This element specifies the schema name of the table.

The schema name that can be specified depends on the database specifications.

Default value

Default schema name of the database used

2.7.60 @TableGenerator

(1) Description

This annotation specifies the settings of the generator that creates the primary key.

The following settings are required when using `@TableGenerator`:

- Specify `GenerationType.TABLE` in the `strategy` element of `@GeneratedValue`.
- Set up the name specified in the `generator` element of `@GeneratedValue` to the `name` element of `@TableGenerator`.

The table generator is specified in the fields or properties of the entity class or primary key. The scope of the generator name is enabled in the persistence unit.

Use the rows of the generator table when generating the primary key value in an entity.

When creating a table for managing the sequence, specify a positive integer in the initial value. If `0` is specified in the initial value, the primary key will be generated from `1`.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@TableGenerator`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the generator name for the primary key value.
`table`	Optional	This element specifies the name of the table that maintains the generated primary key values.
`catalog`	Optional	This element specifies the catalog name of the table that maintains the generated primary key values. Note that Cosminexus JPA provider does not support this attribute.
`schema`	Optional	This element specifies the schema name of the table that maintains the generated primary key values.
`pkColumnName`	Optional	This element specifies the primary key column name of the table that maintains the generated primary key values.
`valueColumnName`	Optional	This element specifies the column name that maintains the final generated value.
`pkColumnValue`	Optional	This element specifies the primary key value of the table that maintains the generated primary key values.
`initialValue`	Optional	This element specifies the value used for initializing the column that maintains the recent generated values. Note that Cosminexus JPA provider does not support this attribute. Ignore when the value is specified.

Element name	Optional/Required	Element description
`allocationSize`	Optional	This element specifies the size of allocating the primary key value from the generator.
`uniqueConstraints`	Optional	This element specifies the unique key constraints in the table that maintains the generated primary key values. Note that Cosminexus JPA provider does not support this attribute. Ignore when the value is specified.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
String

Description
This element specifies the generator name for the primary key value.
The value that can be specified is a character string.

Default value
None

(b) table element

Type
String

Description
This element specifies the name of the table that maintains the generated primary key values.
The table name that can be specified depends on the database specifications.

Default value
`"SEQUENCE"`

(c) schema element

Type
String

Description
This element specifies the schema name of the table that maintains the generated primary key values.
The schema name that can be specified depends on the database specifications.

Default value
Default schema name of the database used

(d) pkColumnName element

Type
String

Description
This element specifies the primary key column name of the table that maintains the generated primary key values.
The column name that can be specified depends on the database specifications.

Default value
`"SEQ_NAME"`

(e) valueColumnName element

Type

String

Description

This element specifies the column name that maintains the final generated value.

The column name that can be specified depends on the database specifications.

Default value

`"SEQ_COUNT"`

(f) pkColumnValue element

Type

String

Description

This element specifies the primary key value of the table that maintains the generated primary key values.

The value that can be specified depends on the type of column of the generated primary key.

Default value

Character string specified in the `name` element

(g) allocationSize element

Type

int

Description

This element specifies the allocation size of the primary key value from the generator.

The value that can be specified is a numeric value that is at least one more than the `int` type.

Note that you can specify the maximum value used during the execution in this element. If you specify a large value for acquiring the management area of the sequence number, the `java.lang.OutOfMemoryError` exception will occur during the execution.

Default value

`50`

2.7.61 @Temporal

(1) Description

This annotation is specified in the persistence property or persistence field having the type that expresses the time (`java.util.Date` and `java.util.Calendar`). This annotation can be used along with `@Basic`.

However, `@Version` and `@Temporal` cannot be specified concurrently. Specify either of these annotations.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@Temporal`:

Element name	Optional/Required	Element description
`value`	Required	This element is specified in the `TemporalType` enumeration type corresponding to the database type.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

TemporalType

Description

This element is specified in the `TemporalType` enumeration type corresponding to the database type.

The following three types of values can be specified:

- `TemporalType.DATE`**: Same as** `java.sql.Data`.
- `TemporalType.TIME`**: Same as** `java.sql.Time`.
- `TemporalType.TIMESTAMP`**: Same as** `java.sql.Timestamp`.

Default value

None

2.7.62 @Transient

(1) Description

This annotation specifies the fields or properties of the following non--persisting classes:

- Entity class
- Mapped superclass
- Embedded class

The applicable targets are method and field.

The value of a field, in which `@Transient` is defined, is not persisted. However, since this value is stored in the persistence context, you can acquire the setup value. Howeachou cannot acquire the value from another EntityManager.

(2) Element

`@Transient` does not have attributes.

2.7.63 @Version

(1) Description

This annotation specifies the `version` field or `version` property used to make use of the optimistic lock functionality.

The following types are supported by the `version` field or `version` property:

- `int`
- `java.lang.Integer`
- `short`
- `java.lang Short`
- `long`
- `java.lang Long`
- `java.sql.Timestamp`

Make note of the following points when using this annotation:

- You cannot specify `@Version` and `@Temporal` concurrently. Specify either of these annotations.

- Do not specify the `version` property in a table other than the primary table.
- In some applications, the field or property specified in `@Version` must not be updated.
- For bulk update, when multiple records are updated at once using SQL, the `version` field or `version` property is not updated automatically. Therefore, when you use the optimistic lock for performing bulk update, you must reference and update manually.
- You can set up only a single `version` field or `version` property for an entity class. If you set up multiple `version` fields or `version` properties, only a single will be enabled. The sequence for enabling the settings is not fixed.

The applicable targets are method and field.

(2) Element

`@Version` does not have attributes.

2.7.64 Correspondence between the annotations and O/R mapping

The following table describes the correspondence between the annotations and O/R mapping files:

Table 2-30: Correspondence between the annotations and O/R mapping files

Annotation	O/R mapping elements
`@AssociationOverride`	<association-override>
`@AssociationOverrides`	--
`@ElementOverride`	<element-override>
`@ElementOverrides`	--
`@Basic`	<basic>
`@Column`	<column>
`@ColumnResult`	<column-result>
`@DiscriminatorColumn`	<discriminator-column>
`@DiscriminatorValue`	<discriminator-value>
`@Embeddable`	<embeddable>
`@Embedded`	<embedded>
`@EmbeddedId`	<embedded-id>
`@Entity`	<entity>
`@EntityListeners`	<entity-listeners>
`@EntityResult`	<entity-result>
`@Enumerated`	<enumerated>
`@ExcludeDefaultListeners`	<exclude-default-listeners>
`@ExcludeSuperclassListeners`	<exclude-superclass-listeners>
`@FieldResult`	<field-result>
`@GeneratedValue`	<generated-value>
`@Id`	<id>
`@IdClass`	<id-class>

Annotation	O/R mapping elements
`@Inheritance`	<inheritance>
`@JoinColumn`	<join-column>
`@JoinColumns`	--
`@JoinTable`	<join-table>
`@Lob`	<lob>
`@ManyToMany`	<many-to-many>
`@ManyToOne`	<many-to-one>
`@MapKey`	<map-key>
`@MappedSuperclass`	<mapped-superclass>
`@NamedNativeQueries`	--
`@NamedNativeQuery`	<named-native-query>
`@NamedQueries`	--
`@NamedQuery`	<named-query>
`@OneToMany`	<one-to-many>
`@OneToOne`	<one-to-one>
`@OrderBy`	<order-by>
`@PersistenceContext`	--#
`@PersistenceContexts`	--#
`@PersistenceProperty`	--#
`@PostLoad`	<post-load>
`@PostPersist`	<post-persist>
`@PostRemove`	<post-remove>
`@PostUpdate`	<post-update>
`@PrePersist`	<pre-persist>
`@PreRemove`	<pre-remove>
`@PreUpdate`	<pre-update>
`@PrimaryKeyJoinColumn`	<primary-key-join-column>
`@PrimaryKeyJoinColumns`	--
`@QueryHint`	<hint>
`@SecondaryTable`	<secondary-table>
`@SecondaryTables`	--
`@SequenceGenerator`	<sequence-generator>
`@SqlResultSetMapping`	<sql-result-set-mapping>
`@SqlResultSetMappings`	--
`@Table`	<table>

Annotation	O/R mapping elements
`@TableGenerator`	<table-generator>
`@Temporal`	<temporal>
`@Transient`	<transient>
`@UniqueConstraint`	<unique-constraint>
`@Version`	<version>

Legend:

--: Not applicable.

#

Not an annotation for O/R mapping.

2.8 javax.servlet.annotation package

This section describes the list of annotations included in the `javax.servlet.annotation` package.

Annotation list

Annotation name	Functionality
`@HandlesTypes`	This annotation specifies the class type that deals with the implementation class of the `ServletContainerInitializer` interface.
`@HttpConstraint`	This annotation specifies the default security constraint.
`@HttpMethodConstraint`	This annotation specifies the security constraints of the HTTP method.
`@MultipartConfig`	This annotation specifies the settings for the Servlet that deals with multipart/form-data requests.
`@ServletSecurity`	This annotation specifies the Servlet security constraints.
`@WebInitParam`	This annotation specifies the initial parameters of Servlet or filter.
`@WebFilter`	This annotation specifies the filer.
`@WebListener`	This annotation specifies the listener.
`@WebServlet`	This annotation specifies the Servlet.

2.8.1 @HandlesTypes

(1) Description

This annotation specifies the class type that deals with the Implementation class of the `ServletContainerInitializer` interface.

(2) Attributes

The following table lists the `@HandlesTypes` attributes:

Attribute name	Functionality
`value`	This attribute specifies the type of class or annotation that deals with the Implementation class of the `ServletContainerInitializer` interface. The class list that is attached with the class that extends or implements the specified class or annotation, is passed to the Implementation class of the `ServletContainerInitializer` interface.

Details of each attribute are as follows:

(a) value attribute

Type

`Class[]`

Description

This attribute specifies the type of class or annotation that deals with the implementation class of the `ServletContainerInitializer` interface. The class list that is attached with the class that extends or implements the specified class or annotation, is passed to the Implementation class of the `ServletContainerInitializer` interface.

Default value

`{}`

2.8.2 @HttpConstraint

(1) Description

This annotation specifies the default security constraints.

(2) Attribute

The following table lists the `@HttpConstraint` attributes:

Attribute name	Functionality
`value`	This attribute specifies the behavior when role is not specified.
`rolesAllowed`	This attribute specifies the list of user names used for authentication.
`transportGuarantee`	This attribute specifies the method to communicate between the client and server.

Details of each attribute are as follows:

(a) value attribute

Type

`ServletSecurity.EmptyRoleSemantic`

Description

This attribute specifies the behavior when the role is not specified.

Default value

```
javax.servlet.annotation.ServletSecurity.EmptyRoleSemantic.
PERMIT
```

(b) rolesAllowed attribute

Type

`String[]`

Description

This attribute specifies the list of user names used for authentication.

Default value

`{}`

(c) transportGuarantee attribute

Type

`ServletSecurity.TransportGuarantee`

Description

This attribute specifies the method to communicate between the client and server.

Default value

```
javax.servlet.annotation.ServletSecurity.
TransportGuarantee.
NONE
```

2.8.3 @HttpMethodConstraint

(1) Description

This annotation specifies the security constraints of the HTTP method.

(2) Attribute

(a) The following table lists the `@HttpMethodConstraint` attributes.

Attribute name	Functionality
`value`	This attribute specifies the HTTP method that applies security constraints.
`emptyRoleSemantic`	This attribute specifies the behavior when the role is not specified.
`rolesAllowed`	This attribute specifies the list of user names used for authentication.
`transportGuarantee`	This attribute specifies the method to communicate between the client and server.

Details of each attribute are as follows:

(b) value attribute

Type
`String`

Description
This attribute specifies the HTTP method that applies security constraints.

Default value
`None`

(c) emptyRoleSemantic attribute

Type
`ServletSecurity.EmptyRoleSemantic`

Description
This attribute specifies the behavior when the role is not specified.

Default value
```
javax.servlet.annotation.ServletSecurity.
EmptyRoleSemantic.
PERMIT
```

(d) rolesAllowed attribute

Type
`String[]`

Description
This attribute specifies the list of user names used for authentication.

Default value
`{}`

(e) transportGuarantee attribute

Type
`ServletSecurity.TransportGuarantee`

Description

This attribute specifies the method to communicate between the client and server.

Default value

`javax.servlet.annotation.ServletSecurity.`
`TransportGuarantee.`
`NONE`

2.8.4 @MultipartConfig

(1) Description

This annotation performs settings for the Servlet that deals with multipart/form-data requests.

The following table lists the `@MultipartConfig` attributes:

Attribute name	Functionality
`fileSizeThreshold`	This attribute sets the threshold size of the uploaded file written on the disk.
`location`	This attribute specifies the directory that stores the uploaded file.
`maxFileSize`	This attribute specifies the maximum size of the uploaded file.
`maxRequestSize`	This attribute specifies the maximum size of multipart/form-data requests.

Details of each attributes are as follows:

(2) Attributes

(a) fileSizeThreshold attributes

Type

`int`

Description

This attribute specifies the threshold size of the uploaded file written on disk.

Default value

`0`

(b) location attribute

Type

`String`

Description

This attribute specifies the directory that stores the uploaded file.

Default value

`""`

(c) maxFileSize attribute

Type

`long`

Description

This attribute specifies the maximum size of the uploaded file.

Default value

`-1L` (limitless)

(d) maxRequestSize attribute

Type
`long`

Description
This attribute specifies the maximum size of multipart/form-data requests.

Default value
`-1L` (limitless)

2.8.5 @ServletSecurity

(1) Description

This annotation specifies the Servlet security constraints.

(2) Attribute

The following table lists the `@ServletSecurity` attributes:

Attribute name	Functionality
`httpMethodConstraints`	This attribute specifies the security constraints for each Servlet HTTP method.
`value`	This attribute specifies the default security constraints of the Servlet.

Details of each attribute are as follows:

(a) httpMethodConstraints attributes

Type
`HttpMethodConstraint[]`

Description
This attribute specifies the security constraints for each Servlet HTTP method.

Default value
`{}`

(b) value attribute

Type
`HttpConstraint`

Description
This attribute specifies the default security constraints of Servlet.

Default value
`@javax.servlet.annotation.HttpConstraint`

2.8.6 @WebInitParam

(1) Description

This annotation specifies the initial parameters of Servlet or filter.

(2) Attribute

The following table lists the `@WebInitParam` attributes:

Attribute name	Functionality
`description`	This attribute specifies the parameter description.
`name`	This attribute specifies the parameter name.
`value`	This attribute specifies the parameter value.

Details of each attributes are as follows:

(a) description attributes

Type

`String`

Description

This attribute specifies the parameter description.

Default value

""

(b) name attribute

Type

`String`

Description

This attribute specifies the parameter name.

Default value

""

(c) value attribute

Type

`String`

Description

This attribute specifies the parameter value.

Default value

`None`

2.8.7 @WebFilter

(1) Description

This annotation specifies the filter.

(2) Attributes

The following table lists the `@WebFilter` attributes:

Attribute Name	Functionality
`description`	This attribute specifies the filter description.
`dispatcherTypes`	This attribute specifies the filter adjustment conditions.

Attribute Name	Functionality
`displayName`	This attribute specifies the display name.
`filterName`	This attribute specifies the filter name.
`initParams`	This attribute specifies the initial parameters for filter.
`largeIcon`	This attribute specifies the large icons used on the GUI tool.
`servletNames`	This attribute specifies the Servlet name of Servlet that performs mapping.
`smallIcon`	This attribute specifies the small icons used on the GUI tool.
`urlPatterns`	This attribute specifies the URL patterns to be mapped.
`value`	This attribute specifies the URL patterns to be mapped. Ignored if specified concurrently with urlPatterns.

Details of each attribute are as follows:

(a) description attribute

Type

`String`

Description

This attribute specifies the filter description.

Default Value

`""`

(b) dispatcherTypes attribute

Type

`DispatcherType[]`

Description

This attribute specifies the filter adjustment conditions.

Default value

`javax.servlet.DispatcherType.REQUEST`

(c) displayName attribute

Type

`String`

Description

This attribute specifies the display name.

Default value

`""`

(d) filterName attribute

Type

`String`

Description

This attribute specifies the filter name.

Default value

`""`

(e) initParams attribute

Type
`WebInitParam[]`

Description
This attribute specifies the initial parameters for filter.

Default value
`{}`

(f) largeIcon attribute

Type
`String`

Description
This attribute specifies the large icons used on the GUI tool.

Default value
`""`

(g) servletNames attribute

Type
`String[]`

Description
This attribute specifies the Servlet name of Servlet that performs mapping.

Default value
`{}`

(h) smallIcon attribute

Type
`String`

Description
This attribute specifies the small icons used on the GUI tool.

Default value
`""`

(i) urlPatterns attribute

Type
`String[]`

Description
This attribute specifies URL patterns to be mapped.

Default Value
`{}`

(j) value attribute

Type
`String[]`

Description
This attribute specifies the URL patterns to be mapped. Ignored if specified concurrently with `urlPatterns`.
Default Value
`{}`

2.8.8 @WebListener

(1) Description

This annotation specifies the listener.

(2) Attributes

The following table lists the `@WebListener` attributes:

Attribute Name	Functionality
`value`	This attribute specifies the description of listener.

(a) value attributes

Type

`String`

Description

This attribute specifies the description of listener.

Default Value

""

2.8.9 @WebServlet

(1) Description

This annotation specifies Servlet.

(2) Attribute

The following table lists the `@WebServlet` attributes:

Attribute Name	Functionality
`description`	This attribute specifies the Servlet description.
`displayName`	This attribute specifies the display name.
`initParams`	This attribute specifies the initial parameters for Servlet.
`largeIcon`	This attribute specifies the large icons used on GUI tool.
`loadOnStartup`	This attribute specifies the start order of Servlet.
`name`	This attribute specifies the Servlet name.
`smallIcon`	This attribute specifies the small icons used on the GUI tool.
`urlPatterns`	This attribute specifies the URL patterns to be mapped.
`value`	This attribute specifies the URL patterns to be mapped. Ignored if specified concurrently with urlPatterns.

Details of each attribute are as follows:

(a) description attributes

Type

`String`

Description

This attribute specifies the Servlet description.

Default value

`""`

(b) displayName attributes

Type

`String`

Description

This attribute specifies the display name.

Default value

`""`

(c) initParams attributes

Type

`WebInitParam[]`

Description

This attribute specifies the initial parameters for Servlet.

Default value

`{}`

(d) largeIcon attributes

Type

`String`

Description

This attribute specifies the large icons used on the GUI tool.

Default value

`""`

(e) loadOnStartup attributes

Type

`int`

Description

This attribute specifies the start order of Servlet.

Default value

`-1`

(f) name attributes

Type

`String`

Description

This attribute specifies the Servlet name.

Default value

`""`

(g) smallIcon attributes

Type
: `String`

Description
: This attribute specifies the small icons used on the GUI tool.

Default value
: `""`

(h) urlPatterns attributes

Type
: `String[]`

Description
: This attribute specifies the URL patterns to be mapped.

Default value
: `{}`

(i) value attributes

Type
: `String[]`

Description
: This attribute specifies the URL patterns to be mapped. Ignored if specified concurrently with `urlPatterns`.

Default value
: `{}`

2.9 Dependency Injection supported on Cosminexus Application Server

The Dependency Injection (DI) is a functionality for the EJB container to automatically set the reference to EJB and resource by specifying annotations (`@EJB`, `@Resource` and `@Inject`) in a field or the `set` method of the target class.

Among the classes running on the EJB container, following are the classes that become target classes:

- Enterprise Bean
- Interceptor

Among the classes running on the Web container, following are the classes that become target classes:

- Servlet
- Filter
- Listener
- Tag handler

When executing DI for referencing the Enterprise Bean home interface or business interface, specify `@EJB`.

When specifying `@Resource`, you can execute DI for the types of resources described in the following table.

Table 2-31: Resource types for which DI can be executed with @Resource

Resource type	Permission of DI#1
`java.lang.String`#2	N
`java.lang.Character`#2	N
`java.lang.Integer`#2	N
`java.lang.Boolean`#2	N
`java.lang.Double`#2	N
`java.lang.Byte`#2	N
`java.lang.Short`#2	N
`java.lang.Long`#2	N
`java.lang.Float`#2	N
`javax.xml.rpc.Service`	N
`javax.xml.ws.Service`	N
`javax.jws.WebService`	N
`javax.sql.DataSource`#3	Y
`javax.jms.ConnectionFactory`	Y
`javax.jms.QueueConnectionFactory`#4	Y
`javax.jms.TopicConnectionFactory`	Y
`javax.mail.Session`	Y
`java.net.URL`	N
`javax.resource.cci.ConnectionFactory`#5	Y

Resource type	Permission of DI[#1]
`org.omg.CORBA_2_3.ORB`	Y[#6]
`javax.jms.Queue`[#3, #7]	Y
`javax.jms.Topic`[#7]	Y
`javax.resource.cci.InteractionSpec`	N
`javax.transaction.UserTransaction`	Y[#8]
`javax.ejb.EJBContext`	Y[#9]
`javax.ejb.SessionContext`	Y[#9]
`javax.ejb.TimerService`	Y[#9, #10]
`JavaBeans resource`	Y
Interface unique to the object to be managed	Y

Legend:

Y: Can be used

N: Cannot be used

#1

The correlation to the object to be managed is established with the `mappedName` element irrespective of the Java Type. Use `!#` to demarcate the display name of the resource adapter and the name of the object to be managed.

#2

You cannot specify in `<env-entry--value>`, therefore, you cannot specify a value acquired in `DI`, `lookup`.

#3

Applicable to DB Connector.

#4

Applicable to TP1/Message Queue-Access, Cosminexus RM.

#5

Applicable to uCosminexus TP1 Connector.

#6

Runs considering `true` is specified for ORB `shareable` element. Note that the ORB object to be injected is a shared instance that is used even with other components.

#7

When using a resource adapter conforming to Connector 1.5, the object to be managed (`javax.jms.Destination` interface or sub interface) that is defined in JMS is specified in the `<connector>-<resourceadapter>-<adminobject>-<adminobject--interface>` tag of the standard DD (`ra.xml`) of resource adapter.

#8

You cannot use in Enterprise Beans or interceptors that run on CMT.

#9

You cannot use in a class that runs on the Web container.

#10

You cannot use in a Stateful Session Bean and an interceptor applied to Stateful Session Bean.

3 APIs Used in the Web Container

This chapter describes the APIs used in the Web Container. This section describes the exception classes unique to the Web container of Application Server.

3.1 Exception classes

Among the APIs of the Web container, this section describes the exception classes provided by Application Server.

The following table describes the exception classes of the Web container:

Table 3-1: Exception classes of the Web container

Exception name	Contents
`com.hitachi.software.web.dbsfo.DatabaseAccessException`	This exception class reports that an attempt to access the database using the database session failover functionality has failed. When this exception is output, make sure that the database is operating properly, and there is no problem in the communication path between the database and the J2EE server. After that take the following countermeasures: When an error occurs in the database Take action against the cause according to the database recovery procedure. When there is a problem in the communication path between the database and the J2EE server Resolve the problem in the communication path. If a problem occurs in the communication path, the mutual exclusion of the database might not be released. Before restarting a business, check the disabled connections, and release the unreleased mutual exclusion. When the database session failover functionality is disabled Do not operate the `HttpSession` object in a request process where the database session failover functionality is disabled due to extension or URI. This exception is thrown if you operate an HTTP session with a URL for which the database session failover functionality is disabled when `true` is specified in the `webserver.dbsfo.exception_type_backcompat` property of the J2EE server. If this exception occurs when there is no problem in the database or the communication path, check whether the HTTP session is operated by the URL for which the database session failover functionality is disabled. The `DatabaseAccessException` class inherits the `java.lang.IllegalStateException` class.
`HttpSessionLimitExceededException class`	This exception class reports that the HttpSession object has exceeded the upper limit. This exception is applicable to the J2EE server mode in which you can specify the upper limit for the number of HttpSession objects. This exception class is not applicable in the servlet engine mode in which you cannot specify the upper limit for the number of HttpSession objects. This exception class is also not applicable to the exception that occurs when the number of global sessions in an SFO server exceeds the upper limit. If you are using the `com.hitachi.software.web.session.HttpSessionLimitExceededException` class, add the *CosminexusApplication Server-installation-directory*`\CC\lib\ejbserver.jar` to the class path, and compile the Java program. The `HttpSessionLimitExceededException` class inherits the `java.lang.IllegalStateException` class.
`com.hitachi.software.web.dbsfo.SessionOperationException`	This is an exception that reports the status in which you cannot operate the HttpSession. This exception is thrown in the following cases: • When the database session failover disable functionality is disabled depending on the extension or URL, you cannot operate the `HttpSession` object in the request processing in which the database session failover disable functionality is disabled. This exception is thrown if you invoke `javax.servlet.http.HttpServletRequest#getSession()` or `getSession(boolean create)` to acquire the `HttpSession` object. • You cannot disable the HTTP session in the reference specific request of the HTTP session. If you invoke `javax.servlet.http.HttpSession#invalidate()` in the reference request, this exception is thrown. • If you perform settings to return error 503 by using the pending queue of the number of the concurrent threads for the Web application unit, you cannot create or disable the HTTP session on an error page specified in DD (`web.xml`). If you create an HTTP session on an

Exception name	Contents
`com.hitachi.software.web.dbsfo.SessionOperationException`	error page specified in DD (`web.xml`) or invoke `javax.servlet.http.HttpSession#invalidate()`, this exception is thrown. If this exception is thrown, check the following points: • If you are using disabling of the database session failover disable functionality, check whether there is a problem in the settings of the extension or URL that is disabled. If there is no problem in the settings, check the Web applications, and check whether the HTTP session operation is performed by the URL that is disabled for the database session failover disable functionality. • If you are using the reference specific request definition functionality of the HTTP session, check whether there is any problem in the URL of the reference request of the HTTP session. If there is no problem in the settings, check the Web application and check whether the HTTP session is disabled in the reference specific request. • If you have done a setting to return error 503 by using a pending queue, check whether the HTTP session is created or disabled in error page specified in DD (`web.xml`). SessionOperationException class is inherited from the `java.lang.IllegalStateException` class.
`com.hitachi.software.web.eadssfo.SessionOperationException`	This exception reports that the operations of HttpSession cannot be performed. This exception is thrown in the following cases: • When you use the EADs session failover disable functionality, you cannot operate the HttpSession object in the request processing in which the session failover disable functionality is disabled. The system throws this exception, when you invoke `javax.servlet.http.HttpServletRequest#getSession()` or `getSession(boolean create)` for getting the HttpSession object. • You cannot disable the HTTP session in the reference specific requests. The system throws this exception if you invoke `javax.servlet.http.HttpSession#invalidate()` in the reference specific request. Confirm the following, when this exception is thrown: • When you use the EADs session failover disable functionality, confirm that there is no problem with the settings of the URL pattern to be disabled. When there is no problem with the settings, check the Web application and check whether the HTTP session operates with the URL that is the target for the EADs session failover disable functionality. • When you use the reference specific request definition functionality, confirm that there is no problem with the settings of the URL pattern of the reference specific request. When there is no problem with the settings, check the Web application and confirm that the HTTP session is not disabled in the reference specific request. The `SessionOperationException` class inherits the `java.lang.IllegalStateException` class.

4 APIs Used by EJB Client Applications

This chapter describes the APIs and exception classes used by EJB client applications.

4.1 List of APIs used by EJB client applications

The APIs used by EJB client applications include the APIs that set security functions and communication timeouts. The following table lists the APIs.

Table 4-1: List of APIs used by EJB client applications

Class name	Function
`EJBClientInitializer` Class	Initializes J2EE services for an EJB client.
`LoginInfoManager` class	Sets the security functionality. For details on this API, see *17.1LoginInfoManager class* in the *uCosminexus Application Server Security Management Guide*.
`RequestTimeoutConfigFactory` Class	Acquires the RequestTimeoutConfig object required for setting the RMI-IIOP timeout.
`RequestTimeoutConfig` Class	Sets the RMI-IIOP timeout.
`UserTransactionFactory` class	Acquires the `UserTransaction` object required for using a transaction in an EJB client.

4.2 EJBClientInitializer Class

Description

Initializes J2EE services for an EJB client.

The package name of the `EJBClientInitializer` class is `com.hitachi.software.ejb.ejbclient.EJBClientInitializer`.

List of methods

Method name	Function
`initialize` method	Initializes J2EE services for an EJB client.

Initialize method

Description

This method initializes J2EE services for an EJB client application. When an EJB client is stopped during a transaction, this method restarts the EJB client and starts the recovery of a global transaction.

Invoke the `initialize` method from the user code of an EJB client, immediately after starting the EJB client process.

Note that if the `javax.naming.InitialContext` is generated or if the `getUserTransaction` method of the `UserTransactionFactory` class is invoked before invoking the `initialize` method, the initialization process is performed at that time.

Format

```
public static void initialize()
throws InitializeFailedException;
```

Parameters

None

Exceptions

`com.hitachi.software.ejb.ejbclient.InitializeFailedException`:

An attempt to initialize services failed.

Return value

None

Caution

If an exception occurs while initializing the services, the EJB client runtime system properties may not have been set properly. Take an action based on the exception message.

4.3 RequestTimeoutConfigFactory Class

Description

This is a factory for acquiring the `RequestTimeoutConfig` object that sets the RMI-IIOP communication timeout. Use the `getRequestTimeoutConfig` method to acquire the `RequestTimeoutConfig` object, and then use the method of the `RequestTimeoutConfig` object to set the timeout.

The package name of the `RequestTimeoutConfigFactory` class is `com.hitachi.software.ejb.ejbclient`.

List of methods

Method name	Function
`getRequestTimeoutConfig` method	Acquires the `RequestTimeoutConfig` object.

getRequestTimeoutConfig method

Description

Acquires the `RequestTimeoutConfig` object.

Format

```
public static RequestTimeoutConfig getRequestTimeoutConfig();
```

Parameters

None

Exceptions

None

Return value

`RequestTimeoutConfig`:

This method returns the `RequestTimeoutConfig` object.

4.4 RequestTimeoutConfig Class

Description

This object sets an RMI-IIOP communication timeout.

The package name of the `RequestTimeoutConfig` class is `com.hitachi.software.ejb.ejbclient`.

List of methods

Method name	Function
`setRequestTimeout` method (format 1)	Sets an RMI-IIOP communication timeout. Sets a timeout for the object.
`setRequestTimeout` method (format 2)	Sets an RMI-IIOP communication timeout. Sets a timeout for the thread.
`unsetRequestTimeout` method	Resets the RMI-IIOP communication timeout set by the `setRequestTimeout` method (format 2) to its default settings.

setRequestTimeout method (format 1)

Description

Sets an RMI-IIOP communication timeout. This method generates a copy of the `obj` parameter and returns an object in which the `sec` parameter is set as the timeout value. The timeout set by this method is valid for the returned object.

Format

```
public java.rmi.Remote setRequestTimeout(java.rmi.Remote obj,
                                         int sec)
  throws IllegalArgumentException,
         IllegalStateException;
```

Parameters

`obj`:

Specify the object (`EJBHome` or `EJBObject`) for which the timeout is set.

`sec`:

Specify an integer in the range of 0 to 86400 for the timeout period (unit: seconds). The timeout is not set when `0` is specified.

Exceptions

`java.lang.IllegalArgumentException`:

This exception is thrown when an invalid object is specified as the target for setting timeout or an invalid value is specified for timeout period.

`java.lang.IllegalStateException`:

An attempt to set the timeout failed.

Return value

This method returns the object that is set with a timeout value.

Caution

When you set a timeout with this method, it takes more time for processing as compared to the time taken to set the timeout by using the `setRequestTimeout` method (format 2).

setRequestTimeout method (format 2)

Description

Sets an RMI-IIOP communication timeout. This method sets the parameter `sec` as the timeout value for running threads. The timeout set by this method is valid for threads that are currently being executed. At the end of the processing, make sure to cancel the timeout settings using the `unset` method. When this method is invoked more than once in the same thread, the value set for timeout gets overwritten.

Format

```
public void setRequestTimeout(int sec)
  throws IllegalArgumentException,
         IllegalStateException;
```

Parameters

sec:

Specify an integer in the range of 0 to 86400 for the timeout period (unit: seconds). The timeout is not set when `0` is specified.

Exceptions

`java.lang.IllegalArgumentException`:

This exception is thrown when an invalid object is specified as the target for setting timeout or an invalid value is specified for timeout period.

`java.lang.IllegalStateException`:

An attempt to set the timeout failed.

Return value

None

Caution

When you set the timeout by this method, make sure to invoke the `unsetRequestTimeout` method and cancel the timeout settings at the end of the processing. If you do not cancel the timeout settings and if the corresponding thread is used while invoking from other clients, an unexpected communication timeout may occur for these clients.

unsetRequestTimeout method

Description

Cancel the RMI-IIOP communication timeout. Cancel the timeout set by the `setRequestTimeout` (format 2), corresponding to the thread that is being executed. Note that when the timeout is set in the thread with the `setRequestTimeout` (format 2) method; make sure to cancel the timeout settings using this method at the end of processing. When this method is invoked without invoking the `setRequestTimeout` (format 2), or even when this method is invoked more than once in the same thread, an exception does not occur.

Format

```
public void unsetRequestTimeout()
  throws IllegalStateException;
```

Parameters

None

Exceptions

`java.lang.IllegalStateException`:

An attempt to cancel timeout failed.

Return value

None

4.5 UserTransactionFactory class

Description

This factory acquires the `UserTransaction` object to use a transaction in an EJB client.

The package name of the `UserTransactionFactory` class is `com.hitachi.software.ejb.ejbclient.UserTransactionFactory`.

List of methods

Method name	Function
`getUserTransaction` method	Acquires the `UserTransaction` object.

getUserTransaction method

Description

Acquires the `UserTransaction` object.

Format

```
public static UserTransaction getUserTransaction();
```

Exceptions

`java.lang.IllegalStateException`:

The API is published from a client other than the EJB client or an attempt to acquire the `UserTransaction` object failed.

Return value

`javax.transaction.UserTransaction` object

4.6 Exception Class

This subsection describes the classes provided by Application Server from among the exception classes used by the APIs of EJB client applications.

The following table describes the exception classes used by the APIs of EJB client applications:

Table 4-2: Exception classes used by the APIs of EJB client applications

Exception name	Contents
`com.hitachi.software.ejb.security.base.authentication.NotFoundServerException`	This exception is output when you cannot connect to the J2EE server to be logged in and when you try to log in using the `login` method of the `LoginInfoManager` class. Check whether the J2EE server name specified in the `ejbserver.serverName` property is the same as that of the J2EE server name to be logged in. Also, confirm whether the J2EE server to be logged in is running.
`com.hitachi.software.ejb.security.base.authentication.InvalidUserNameException`	This exception is thrown when you try to log in using the `login` method of the `LoginInfoManager` class and the user name is invalid. Confirm whether the user name is correct.
`com.hitachi.software.ejb.security.base.authentication.InvalidPasswordException`	This exception is thrown when you try to log in using the `login` method of the `LoginInfoManager` class and the password is invalid. Confirm whether the password is correct.

5 APIs Used When Using the TP1 Inbound Adapter to Link with OpenTP1(INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

5.1 (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

6 APIs Used in the Asynchronous Parallel Processing of Threads

This chapter describes the APIs used in the asynchronous parallel processing of threads. This chapter also describes Cosminexus Application Server APIs that differ in operations from the APIs defined by the specifications of Timer and Work Manager for Application Servers.

6.1 List of Cosminexus APIs that differ in operation from Timer and Work Manager for Application Servers specifications

The following table describes the names and operations of Cosminexus Application Server APIs that differ in operation from the APIs defined by the specifications of Timer and Work Manager for Application Servers.

Table 6-1: List of Cosminexus Application Server APIs with operations that differ from the Timer and Work Manager for Application Servers specifications

Class name	Method name	Operations in Cosminexus Application Server
`commonj.timers.TimerManager` class	`schedule(TimerListener listener,Date time)` method	Returns `IllegalArgumentException`, when the listener inherits `javax.ejb.EnterpriseBean`.
	`schedule(TimerListener listener,long delay)` method	
	`schedule(TimerListener listener,Date firstTime,long period)` method	
	`schedule(TimerListener listener,long delay,long period)` method	
	`scheduleAtFixedRate(TimerListener listener,Date firstTime,long period)` method	
	`scheduleAtFixedRate (TimerListener listener,long delay,long period)` method	
`commonj.work.WorkManager` class	`schedule(Work work)` method	Throws `WorkException`, when `work` is `null`.
	`schedule(Work work,WorkListener wl)` method	Returns `WorkException`, when `work` is `null`. Returns `IllegalArgumentException`, when `WorkListener` inherits `javax.ejb.EnterpriseBean`.

7 APIs Used in the User Log Functionality

This chapter describes the APIs used in the user log functionality.

7.1 List of APIs used in the user log functionality

The following table lists and describes the APIs used when logs (user logs) are output by J2EE applications, batch applications, or EJB client applications in the Hitachi trace common library format.

Table 7-1: List of APIs used in user log functionality

Class name	Function
`CJLogRecord` Class	Adds the `MsgID` and `AppName` parameter in the `LogRecord` class. You can also output the field value of the `MsgID` and `AppName` with the value specified during execution by passing the `LogRecord` object (`CJLogRecord` object), created using the methods of this class, to the `Logger.log` method.

7.2 CJLogRecord Class

Description

This class adds the `MsgID` and `AppName` parameters in the `java.util.logging.LogRecord` class. This class provides a static method for creating the `LogRecord` object (hereafter, called `CJLogRecord`), when the `MsgID` and `AppName` are specified.

The package name of the `CJLogRecord` class is `com.hitachi.software.ejb.application.userlog`.

List of methods

Method name	Function
create Method (Format 1)	Passes the `Level`, `Message`, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 2)	Passes the `Level`, `Message`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 3)	Passes the `Level`, `Message`, `Object`, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 4)	Passes the `Level`, `Message`, `Object`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 5)	Passes the `Level`, `Message`, `Thrown`, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 6)	Passes the `Level`, `Message`, `Thrown`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 7)	Passes the `Level`, `Message`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 8)	Passes the `Level`, `Message`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 9)	Passes the `Level`, `Message`, `Thrown`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.
create Method (Format 10)	Passes the `Level`, `Message`, `Thrown`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 1)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 2)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 3)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Object`, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 4)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Object`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 5)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Thrown`, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 6)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Thrown`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Method name	Function
createp Method (Format 7)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 8)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 9)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Thrown`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.
createp Method (Format 10)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Thrown`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 1)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `message` and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 2)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 3)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Object`, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 4)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Object`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 5)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Thrown`, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 6)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Thrown`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 7)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 8)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 9)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Thrown`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.
createrb Method (Format 10)	Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Thrown`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Note that the `LogRecord` class that is the source of inheritance of the `CJLogRecord` class and the `Level` to be specified in the parameters of each method are classes that belong to the `java.util.logging` package.

create Method (Format 1)

Description

Passes the `Level`, `Message`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 2)

Description

Passes the `Level`, `Message`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 String appName,
                                 String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 3)

Description

Passes the `Level`, `Message`, `Object`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object param1,
                                 String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`param1`:

Specify an object to be set in the `LogRecord`.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 4)

Description

Passes the `Level`, `Message`, `Object`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object param1,
                                 String appName,
                                 String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`param1`:

Specify an object to be set in the `LogRecord`.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 5)

Description

Passes the `Level`, `Message`, `Thrown`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 6)

Description

Passes the `Level`, `Message`, `Thrown`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 String appName,
                                 String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`) `msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 7)

Description

Passes the `Level`, `Message`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object[] params,
                                 String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 8)

Description

Passes the `Level`, `Message`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                Object[] params,
                                String appName,
                                String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 9)

Description

Passes the `Level`, `Message`, `Thrown`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                String msg,
                                Throwable thrown,
                                Object[] params,
                                String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

create Method (Format 10)

Description

Passes the `Level`, `Message`, `Thrown`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 Object[] params,
                                 String appName,
                                 String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 1)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 2)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  String appName,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 3)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Object`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object param1,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`param1`:

Specify an object to be set in the `LogRecord`.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 4)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Object`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object param1,
                                  String appName,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`param1`:

Specify an object to be set in the `LogRecord`.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 5)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Thrown`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 6)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Thrown`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  String appName,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 7)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object[] params,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 8)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object[] params,
                                  String appName,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 9)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Thrown`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  Object[] params,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createp Method (Format 10)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), `Message`, `Thrown`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  Object[] params,
                                  String appName,
                                  String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 1)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `message` and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                        String sourceMethod,
                                        String bundleName,
                                        String msg,
                                        String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 2)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                        String sourceClass,
                                        String sourceMethod,
                                        String bundleName,
                                        String msg,
                                        String appName,
                                        String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 3)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Object`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Object param1,
                                   String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`param1`:

Specify an object to be set in the `LogRecord`.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 4)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Object`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Object param1,
                                   String appName,
                                   String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`param1`:

Specify an object to be set in the `LogRecord`.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 5)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Thrown`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                        String bundleName,
                                        String msg,
                                        Throwable thrown,
                                        String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 6)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Thrown`, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                        String sourceClass,
                                        String sourceMethod,
                                        String bundleName,
                                        String msg,
                                        Throwable thrown,
                                        String appName,
                                        String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 7)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Object[] params,
                                   String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 8)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Object[] params,
                                   String appName,
                                   String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 9)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Thrown`, `Object` array, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Throwable thrown,
                                   Object[] params,
                                   String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

createrb Method (Format 10)

Description

Passes the `Level`, source class name (`sourceClass`), source method name (`sourceMethod`), resource bundle name (`bundleName`), `Message`, `Thrown`, `Object` array, `AppName`, and `MsgID`, and creates the `CJLogRecord` object.

Format

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   Throwable thrown,
                                   Object[] params,
                                   String appName,
                                   String msgID);
```

Parameters

`level`:

Specify a message level identifier (for example, `SEVERE`)

`sourceClass`:

Specify a class name that publishes a logging request.

`sourceMethod`:

Specify a method name that publishes a logging request.

`bundleName`:

Specify a resource bundle name to regionalize the `msg`.

`msg`:

Specify a string message or key of the message catalog.

`thrown`:

Specify an exception object to be set in the `LogRecord`.

`params`:

Specify a user-specific `Object` array (planned to be passed directly to an `Object` array of the `Logger.log` method) used by the user.

`appName`:

Specify a value (application distinguished name) to be output to the `AppName` field.

`msgID`:

Specify a value (message string) to be output to the `MsgID` field.

Return value

This method returns the `CJLogRecord` object.

8 APIs Used to Output Audit Logs (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

8.1 (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

9 APIs Used in Performance Analysis Trace

This chapter describes the APIs used in the functionality of performance analysis trace for acquiring root application information.

9.1 List of APIs used in performance analysis trace

The following table lists the APIs used in performance analysis trace.

Table 9-1: List of APIs used in performance analysis trace functionality

Class name	Function
`CprfTrace` Class	Provides functionality related to performance analysis trace.

9.2 CprfTrace Class

Description

Provides functionality related to performance analysis trace.

The package name of the `CprfTrace` class is `com.hitachi.software.ejb.application.prf`.

List of methods

Method name	Function
`getRootApInfo` Method	Returns a character string expression for the root application information of performance analysis trace stored by an existing thread.

Caution

When using this class, compile the class by specifying the following JAR files in the class path:

- In Windows
  application-server-installation-directory`\CC\lib\ejbserver.jar`
- In UNIX
  `/opt/Cosminexus/CC/lib/ejbserver.jar`

getRootApInfo Method

Description

Returns a character string expression for the root application information of performance analysis trace stored by an existing thread.

The character string expression of the root application information contains the IP address, process ID, and communication number configuring the root application information demarcated with a forward slash (/) (maximum length is 45).

```
Example "10.209.15.130/1234/0x0000000000000001"
```

By registering the character string expression of the root application information in the log file, you can compare it with the performance analysis trace file at any time and consequently use it for troubleshooting.

See *7.4 Implementation for collection of root application information of trace based performance analysis* in the *uCosminexus Application Server Maintenance and Migration Guide* for acquiring the root application information trace based performance analysis. For checking the log by using the root application information, see*7.7.7 Investigating the Log Using the Root Application Information* in the *uCosminexus Application Server Maintenance and Migration Guide*.

Format

```
public static final String getRootApInfo();
```

Parameters

None

Exceptions

None

Return value

A character string expression of the root application information.

This method returns `null` in the following cases:

- When Cosminexus Performance Tracer is not installed, and root application information with current threads is not available.
- When invoked from outside the EJB container and Web container
- When invoked from the EJB client

10 APIs Used with JavaVM

This chapter describes the APIs used in the product JavaVM (hereafter called *JavaVM*).

JavaVM is compliant to Java SE 6. For details, see the manual *uCosminexus Application Server Overview*. For details on APIs available in JDK 6, see the JDK 6 documentation provided by Oracle Corporation.

10.1 List of APIs used with JavaVM

The following table lists the APIs used with JavaVM.

Table 10-1: List of APIs used by JavaVM

Class name	Function
`BasicExplicitMemory` Class	This is a class that indicates the Explicit memory block. Note that you cannot use this class for other JDK products.
`ExplicitMemory` Class	This is an abstract class that indicates the Explicit memory block. This class defines the contents to be processed in the BasicExplicitMemory class that is an inherited class. Note that you cannot use this class for other JDK products.
`MemoryArea` Class	This is an abstract class that indicates the Explicit memory block or the Java heap. Note that you cannot use this class for other JDK products.
`MemoryInfo` Class	Acquires the memory information of garbage collection. Note that you cannot use this class for other JDK products.
`Exception` classes	This is an exception class that expresses exceptions that occur in the APIs used by JavaVM. Note that you cannot use this class for other JDK products.

Omission of the package name

Note that in this chapter, the class names of the classes belonging to the `java.lang` and `MemoryArea` packages are mentioned only with the class names instead of the complete names.

Example:

For `java.lang.Object`: `Object`

10.2 BasicExplicitMemory Class

Description

This is a class that indicates the Explicit memory block created by the Application Server. This class inherits the `ExplicitMemory` class.

The package of the `BasicExplicitMemory` class is `JP.co.Hitachi.soft.jvm.MemoryArea`.

List of constructor and methods

Constructor and method name	Function
`BasicExplicitMemory constructor` (Format 1)	Initializes the Explicit memory block.
`BasicExplicitMemory constructor` (Format 2)	Initializes the Explicit memory block, and then sets up a name.
`getName` method	Returns the Explicit memory block name.

BasicExplicitMemory constructor (Format 1)

Description

Initializes the Explicit memory block. Through initialization, the object can be arranged in the Explicit heap.

After initializing the Explicit memory block, the instance name is set to `BasicExplicitMemory-`*Explicit-memory-block-ID*. However, the memory area of the Explicit memory block is not reserved.

For the following conditions, disable the Explicit memory block:

- When the option `HitachiUseExplicitMemory` is `OFF` (when `-XX:+HitachiUseExplicitMemory` is not specified)
- When the maximum number of Explicit memory blocks is exceeded

Format

```
public BasicExplicitMemory();
```

Parameters

None

Exceptions

None

Return value

None

BasicExplicitMemory constructor (Format 2)

Description

Initializes the Explicit memory block, and simultaneously sets up a name for the Explicit memory block.

The name of the instance will become the parameter name. However, if the parameter name is `null`, the instance name will become `BasicExplicitMemory-`*Explicit-memory-block-ID*.

For the following conditions, disable the Explicit memory block:

- When the option `HitachiUseExplicitMemory` is `OFF` (when `-XX:+HitachiUseExplicitMemory` is not specified)
- When the maximum number of Explicit memory blocks is exceeded

Format

```
public BasicExplicitMemory(String name)
```

Parameters

name:
Specifies the String that indicates the name.

Exceptions

None

Return value

None

getName method

Description

Returns the name of the Explicit memory block indicated by this object.

Format

```
public String getName();
```

Parameters

None

Exceptions

None

Return value

This method acquires the name of the Explicit memory block indicated by the object from the instance field that indicates the name of the object, and then returns the reference.

Caution

The uniqueness of the Explicit memory block ID that is added to the name set up by default has been assured. However, when the instance that contains this ID is discarded, the same Explicit memory block ID will be reused. In such cases, the different instances that do not coexist have the same default name.

10.3 ExplicitMemory Class

Description

This is an abstract class that indicates the Explicit memory block. This class defines the processing in `BasicExplicitMemory`.

The package of the `ExplicitMemory` class is `JP.co.Hitachi.soft.jvm.MemoryArea`.

List of methods

Method name	Function
`countExplicitMemories` Method	Returns the number of Explicit memory blocks.
`freeMemory` method	Returns the usable size of Explicit memory blocks.
`getMemoryUsage` Method	Returns the usage state of the Explicit heap.
`isActive` Method	Returns whether or not the Explicit memory block can be processed.
`isReclaimed` Method	Returns whether or not the Explicit memory block is in the reserved for release state or released state.
`newArray` method (format 1)	Creates array objects in Explicit memory blocks.
`newArray` method (format 2)	
`newInstance` method (format 1)	Creates objects in Explicit memory blocks.
`newInstance` method (format 2)	
`newInstance` method (format 3)	
`reclaim` Method (Format 1)	Reserves the Explicit memory blocks for release.
`reclaim` Method (Format 2)	
`reclaim` Method (Format 3)	
`reclaim` Method (Format 4)	
`setName` method	Sets up the Explicit memory block name in `name`.
`toString` method	Returns the Explicit memory block name.
`totalMemory` method	Returns the reserved size of Explicit memory blocks.
`usedMemory` method	Returns the used memory size of Explicit memory blocks.

countExplicitMemories Method

Description

Returns the number of Explicit memory blocks in the Explicit heap. If an Explicit memory block is already released or is disabled, that Explicit memory block is not counted.

Format

```
public static int countExplicitMemories();
```

Parameters

None

Exceptions

None

Return value

This method counts the number of Explicit memory blocks in the Explicit heap, and then returns the number in the `int` type.

Caution

- Even when the Explicit memory blocks are not operated explicitly for counting the number of blocks that are already reserved for release, the number might change over time.
- This method is not used to count the number of ExplicitMemory instances, but instead it counts the actual number of Explicit memory blocks.

freeMemory method

Description

Returns the memory size that can be used by Explicit memory blocks.

Format

```
public long freeMemory();
```

Parameters

None

Exceptions

InaccessibleMemoryAreaException:
  This functionality is not supported.

Return value

The common error check is performed and then this method returns either of the following values. For details on common error check, see *10.6 Error check (common error check) of the process that controls the Explicit memory block*.

0:
  Returned when the API processing cannot be performed.

The memory size (number of bytes) that can be used for the Explicit memory block indicated by this object:
  When you can perform API processing, the memory size that can be used for the Explicit memory block, indicated by the object, is returned in the `long` type.

getMemoryUsage Method

Description

Returns the usage state of the Explicit heap.

Format

```
public static java.lang.management.MemoryUsage getMemoryUsage();
```

Parameters

None

Exceptions

None

Return value

This method returns the reference to the `java.lang.management.MemoryUsage` instance that maintains the usage state of the Explicit heap as a field, in the form of the following values:

init:

This is the initial value of the Explicit heap. This value is always `0`.

used:

This is the memory size (number of bytes) being used in the Explicit heap.

committed:

This is the reserved size (number of bytes) of the Explicit heap.

max:

This is the value (number of bytes) of the maximum Explicit heap size specified by `-XX:HitachiExplicitHeapMaxSize`. However, when the option `HitachiUseExplicitMemory` is `OFF` (when `-XX:+HitachiUseExplicitMemory` is specified), `0` is returned.

Caution

The value contained in the `MemoryUsage` instance is the value at the point of time when `getMemoryUsage()` is invoked. This value might be different from the actual value when each field is read out from the `MemoryUsage` instance.

isActive Method

Description

Returns whether or not the Explicit memory block indicated by the object can be processed.

Format

```
public boolean isActive();
```

Parameters

None

Exceptions

None

Return value

This method returns the state of the Explicit memory block indicated by the object, in the following `Boolean` types:

true:

The Explicit memory block can be processed. This value is returned for an enabled Explicit memory block, when the substate is `Enable`.

false:

The Explicit memory block cannot be processed. This value is returned in either of the following cases:

- A disabled Explicit memory block
- An enabled Explicit memory block, when the substate is `Disable`

Caution

Once the ExplicitMemory is disabled, it cannot be enabled again.

isReclaimed Method

Description

Returns whether or not the Explicit memory block indicated by the object is in the reserved for release state or the released state.

Format

```
public boolean isReclaimed();
```

Parameters

None

Exceptions

None

Return value

This method returns the state of the Explicit memory block indicated by the object, in the following `boolean` types:

true:

This value is returned when the Explicit memory block indicated by the object is either in the reserve for release state or the released state.

false:

This value is returned when the Explicit memory block indicated by the object is in the enabled state.

newArray method (format 1)

Description

Directly creates an array instance of the length specified in the parameter `length` of the class that is specified in the parameter `type` of the Explicit memory block indicated by the object.

Format

```
public Object newArray(Class type,  int length);
```

Parameters

type:

This parameter specifies the class of the array instance to be created directly.

length:

This parameter specifies the length of the array instance to be created directly.

Exceptions

NullPointerException:

The parameter `type` is `null`.

NegativeArraySizeException:

The parameter `length` is `0` or less than `0`.

IllegalArgumentException:

An array class for which the parameter `length` is more than `0` and the parameter `type` is more than 255 dimensions or the array class having the type `Void.TYPE`.

InaccessibleMemoryAreaException:

This functionality is not supported.

Return value

This method directly creates an array of the type `type` in the Explicit memory block indicated by the object, and then returns the reference. The length of the array is as specified in `length`.

If it is judged that processing cannot be executed by performing the common error check, invoke `java.lang.reflect.Array.newInstance(Class<?>,componentType,int length)` by the parameter type and parameter length, and return that result. For details on common error check, see *10.6 Error check (common error check) of the process that controls the Explicit memory block* .

newArray method (format 2)

Description

Directly creates an array instance with a dimension `dimensions.length` in the Explicit memory block indicated by the object. Note that the number of elements of the nth dimension of the class indicated by the parameter `type` is `dimensions[n-1]`.

Format

```
public Object newArray(Class type,  int[] dimensions);
```

Parameters

type:

This parameter specifies the class of the array instance to be created directly.

dimensions:

This parameter specifies the number of dimensions and elements of the array instance to be created directly.

Exceptions

NullPointerException:

The value of either one or both the parameters `dimensions` and `type` is `null`.

NegativeArraySizeException:

The parameter `dimension has` an element with a negative value.

IllegalArgumentException:

This exception is thrown in any of the following cases:

- When `dimensions.length` of the parameter `dimensions` is less than `0` or more than `255`
- When the total of the number of dimensions of the parameter `type` and `dimensions.length` of the parameter `dimensions` is more than `255`
- When the parameter `type` is `Void.TYPE`

InaccessibleMemoryAreaException:

This functionality is not supported.

Return value

This method directly creates an array object with the dimension `dimensions.length` and the type `type`, wherein the number of elements of the nth dimension is `dimensions[n-1]`, in the Explicit memory block indicated by the object, and then returns the reference.

If it is judged that processing cannot be executed by performing the common error check, invoke `java.lang.reflect.Array.newInstance (Class<?> componentType,int[] dimensions)` by the parameter type and parameter dimensions, and return that result. For details on common error check, see *10.6 Error check (common error check) of the process that controls the Explicit memory block* .

newInstance method (format 1)

Description

Directly creates the instances of the class indicated by the parameter `type` in the Explicit memory block indicated by the object. Only the instances of the class specified in the parameter are created in the Explicit memory block. The objects created through the initialization by the constructor of the instances of the class specified in the parameter are created in the Java heap. `type` is same as `Class.newInstance()` for this object, however some part is different.

Format

```
public Object newInstance(Class type);
```

Parameters

type:

This is the class of the array instance to be created directly.

Exceptions

NullPointerException:

Either the parameter `type` or the class indicated by the parameter `type` is `null`.

SecurityException:

This exception is thrown when SecurityManager exists, and when any of the following conditions hold true:

- The invocation of `s.checkMemberAccess(type, Member.PUBLIC)` does not allow access to this constructor.
- The class loader at the calling side is different.
- The invocation of the class loader higher than the current class loader and also the invocation of `s.checkPackageAccess()` do not allow access to the package of this class.

NoSuchMethodException:

No constructor, without a public parameter, exists in either the parameter `type` or the class indicated by the parameter `type`.

ExceptionInInitializerError:

An attempt to initialize the parameter `type` or the class indicated by the parameter `type` has failed.

InstantiationException:

The parameter `type` or the class indicated by the parameter `type` is either an abstract class or an interface.

InvocationTargetException:

An exception occurred during the execution of the parameter `type` or the constructor of the class indicated by the parameter `type`.

IllegalAccessException:

The class or its `nullary` constructor cannot be accessed.

InaccessibleMemoryAreaException:

This functionality is not supported.

Return value

This method returns the reference to the instances created in the Explicit memory block indicated by the object.

If it is judged that processing cannot be executed by performing the common error check, invoke `the Class.newInstance()` method by considering the parameter type as a receiver, and return that result. For details on common error check, see *10.6 Error check (common error check) of the process that controls the Explicit memory block* .

Caution

We recommend that you add a public class in the parameter `type`.

newInstance method (format 2)

Description

Directly creates the instances of the class indicated by the parameter `type` in the Explicit memory block. The value specified in the parameter `args` is passed as an argument of the constructor that creates the instances. The objects created through the initialization by the constructor of the instances of the class specified in the parameter are created in the Java heap.

Format

```
public Object newInstance(Class type,  Object... args);
```

Parameters

type:

This is the class of the array instance to be created directly.

args:

This is a parameter that is passed to the constructor.

Exceptions

NullPointerException:

The value of either one of the parameters or both the parameters `type` and `args` is `null`.

SecurityException:

This exception is thrown when SecurityManager exists, and when any of the following conditions hold true:

- The invocation of `s.checkMemberAccess(type, Member.PUBLIC)` does not allow access to this constructor.
- The class loader at the calling side is different.
- The invocation of the class loader is higher than the current class loader, and also the invocation of `s.checkPackageAccess()` does not allow access to the package of this class.

NoSuchMethodException:

A public constructor having a parameter of the same type as the elements of the parameter `args` does not exist in the class indicated by the parameter `type`.

ExceptionInInitializerError:

An attempt to initialize the parameter `type` or the class indicated by the parameter `type` has failed.

InstantiationException:

The parameter `type` or the class indicated by the parameter `type` is either an abstract class or an interface.

InvocationTargetException:

An exception occurred during the execution of the parameter `type` or the constructor of the class indicated by the parameter `type`.

InaccessibleMemoryAreaException:

This functionality is not supported.

IllegalAccessException:

The base constructor cannot be accessed because Java language access control is executed in the `Constructor` object.

IllegalArgumentException:

This exception is thrown in each of the conditions:

- When the number of real parameters and virtual parameters is different
- When the lap release conversion of the primitive arguments fails
- When the parameter value cannot be converted to the corresponding virtual parameter type after lap release of the primitive argument
- When the constructor is related to the enumeration type

Return value

This method returns the reference to the instances created in the Explicit memory block indicated by the object.

If it is judged that processing cannot be executed after performing the common error check, acquire the `java.lang.reflect.Constructor` instance as `type.getConstructor(arg_types#)`. In this case, consider `java.lang.reflect.Constructor` as a receiver, `parameter args` as a parameter, invoke the `java.lang.reflect.Constructor.newInstance(Object... initargs)` method, and return the result. For details on common error check, see *10.6 Error check (common error check) of the process that controls the Explicit memory block* .

#

`arg_types` is a class array in which the results of invocation of `Object.getClass()` as the object are assumed as the elements of the parameter `args`.

Caution

We recommend that you add a public class in the parameter `type`.

You cannot invoke a constructor in which the primitive type is assumed as an argument. To invoke a constructor in which the primitive type is assumed as an argument, use the `newInstance` method (format 3). A coding example in which the `newInstance` method (format 3) is used is as follows:

```
import JP.co.Hitachi.soft.jvm.MemoryArea.*;
import java.lang.reflect.*;
public class test1 {
  public static void main(String[] args) throws Exception {
    ExplicitMemory em = new BasicExplicitMemory();
    TheClass obj = null;

    Constructor cons = TheClass.class.getConstructor(new Class[]{int.class});
    obj = (TheClass)em.newInstance(cons, 1);            // Execution successful

    obj = (TheClass)em.newInstance(TheClass.class, 1);  //
NoSuchMethodException is thrown
  }
}

public class TheClass {
  public TheClass(int i){}
}
```

newInstance method (format 3)

Description

Executes the constructor indicated by the parameter `cons` in the parameter `args`, and then directly creates the instance in the Explicit memory block indicated by the object. Only the instances of the class specified in the parameter are created in the Explicit memory block. The objects created through initialization by the constructor of the instances of class specified in the parameter are created in the Java heap.

Format

```
public Object newInstance(java.lang.reflect.Constructor cons,  Object... args);
```

Parameters

cons:
: This parameter specifies the constructor of the array instance to be created directly.

args:
: This is a parameter that is passed to the constructor.

Exceptions

NullPointerException:
: The value of either one or both the parameters `cons` and `args` is `null`.

ExceptionInInitializerError:
: An attempt to initialize the class with the constructor indicated by the parameter `cons` has failed.

InstantiationException:
: The constructor indicated by the parameter `cons` is an abstract class.

IllegalArgumentException:
: The parameter of the constructor indicated by the parameter `cons` does not match the parameter `args`.

InvocationTargetException:
: An exception occurred during the execution of the constructor indicated by the parameter `cons` or parameter `args`.

InaccessibleMemoryAreaException:
: This functionality is not supported.

Return value

This method returns the reference to the instances created in the Explicit memory block indicated by the method.

If it is judged that processing cannot be executed by performing the common error check, invoke parameter `cons` of `java.lang.reflect.Constructor.newInstance(Object... initargs)` as this object, parameter `args` as a parameter, and return that result. For details on common error check, see *10.6 Error check (common error check) of the process that controls the Explicit memory block* .

Caution

We recommend that you add the constructor of a public class in the parameter `cons`.

reclaim Method (Format 1)

Description

Reserves the release processing for all the elements of the parameter `areas`.

When the parameter `areas` is other than `null`, execute the same processing that is executed when `ExplicitMemory.reclaim(ExplicitMemory area)` is invoked assuming the elements as the parameters, for all the elements of the parameter `areas`. The order of the elements, for which the processing is performed, is not defined. If an exception occurs during the processing for any element, that exception will be thrown. The processing is not executed for the elements that are not processed until the exception is thrown.

Format

```
public static void reclaim(ExplicitMemory... areas);
```

Parameters

areas:
: This parameter specifies the array containing the Explicit memory block for which the release processing is to be reserved in the elements.

Exceptions

NullPointerException:
: The parameter `areas` is `null`.

InaccessibleMemoryAreaException:
: This functionality is not supported.

Return value

None

Caution

This method only reserves the release processing, and does not actually perform the release processing.

When the option `HitachiExplicitMemoryAutoReclaim` is `ON` (`-XX:+HitachiExplicitMemoryAutoReclaim` is specified), the Explicit memory block after the automatic release will execute the same operation as is executed by the Explicit memory block that is generated by the explicit memory management automatic deployment settings file. If you do not want this operation to be performed, set the option `HitachiExplicitMemoryAutoReclaim` to `OFF` (`-XX:+HitachiExplicitMemoryAutoReclaim` is not specified).

reclaim Method (Format 2)

Description

If it is judged by the common error check that the processing can be executed when the parameter `area` is other than `null`, execute the exclusion processing in the parameter `area`, and then reserve the Explicit memory block indicated by the parameter `area` for the release.

The processing is not executed in the following cases:

- When the parameter `area` is `null`
- When it is judged by the common error check that the processing cannot be executed

Format

```
public static void reclaim(ExplicitMemory area);
```

Parameters

area:
This parameter specifies the Explicit memory block for which the release processing is to be reserved.

Exceptions

InaccessibleMemoryAreaException:
This functionality is not supported.

Return value

None

Caution

This method only reserves the release processing, and does not actually perform the release processing.

When the option `HitachiExplicitMemoryAutoReclaim` is `ON` (`-XX:+HitachiExplicitMemoryAutoReclaim` is specified), the Explicit memory block after the automatic release will execute the same operation as is executed by the Explicit memory block that is generated by the explicit memory management automatic deployment settings file. If you do not want this operation to be performed, set the option `HitachiExplicitMemoryAutoReclaim` to `OFF` (`-XX:+HitachiExplicitMemoryAutoReclaim` is not specified).

reclaim Method (Format 3)

Description

Reserves the release processing for the Explicit memory blocks indicated by the parameter `area0` and `area1`.

Execute the processing that is executed when `ExplicitMemory.reclaim(ExplicitMemory area)` is invoked using the parameter `area0` and `area1` as the parameters. The processing order of the parameter `area0` and `area1` is not defined. If an exception occurs during the processing for one parameter, that exception will be thrown. If the other parameter is unprocessed, the processing will not be executed.

Format

```
public static void reclaim(ExplicitMemory area0,  ExplicitMemory area1);
```

Parameters

area0:
: This parameter specifies the Explicit memory block 1 for which the release processing is to be reserved.

area1:
: This parameter specifies the Explicit memory block 2 for which the release processing is to be reserved.

Exceptions

InaccessibleMemoryAreaException:
: This functionality is not supported.

Return value

None

Caution

This method only reserves the release processing, and does not actually perform the release processing.

When the option `HitachiExplicitMemoryAutoReclaim` is `ON` (`-XX:+HitachiExplicitMemoryAutoReclaim` is specified), the Explicit memory block after the automatic release will execute the same operation as is executed by the Explicit memory block that is generated by the explicit memory management automatic deployment settings file. If you do not want this operation to be performed, set the option `HitachiExplicitMemoryAutoReclaim` to OFF (`-XX:+HitachiExplicitMemoryAutoReclaim` is not specified).

reclaim Method (Format 4)

Description

Reserves the release processing for all the elements of the parameter `areas`.

When the parameter `areas` is other than `null`, execute the same processing that is executed when `ExplicitMemory.reclaim(ExplicitMemory area)` is invoked assuming the elements as the parameters, for all the elements of the parameter `areas`. The order of the elements for which the processing is performed is not defined. If an exception occurs during the processing for any element, that exception will be thrown. Processing is not executed for the elements that are not processed until the exception is thrown.

Format

```
public static void reclaim(Iterable<ExplicitMemory> areas);
```

Parameters

areas:
: This parameter specifies the iterator of the Explicit memory block for which the release processing is reserved.

Exceptions

NullPointerException:
: The value of the parameter `areas` is `null`.

InaccessibleMemoryAreaException:
: This functionality is not supported.

Return value

None

Caution

This method only reserves the release processing, and does not actually perform the release processing.

When the option `HitachiExplicitMemoryAutoReclaim` is `ON` (`-XX:+HitachiExplicitMemoryAutoReclaim` is specified), the Explicit memory block after the automatic release will execute the same operation as is executed by the Explicit memory block that is generated by the explicit memory management automatic deployment settings file. If you do not want this operation to be performed, set the option `HitachiExplicitMemoryAutoReclaim` to `OFF` (`-XX:+HitachiExplicitMemoryAutoReclaim` is not specified).

setName method

Description

Sets up a name for the Explicit memory block. The parameter `name` is set up in the instance field that indicates the name of this object, as the name of the Explicit memory block indicated by the object.

This name is mainly set up for the purpose of debugging. The specified value is displayed in the event log or the thread dump.

Format

```
public void setName(String name);
```

Parameters

name:
: This parameter specifies the string that indicates the name to be set up.

Exceptions

NullPointerException:
: The value of the parameter `name` is `null`.

Return value

None

Caution

The names are not unique because the same name can be set up in multiple `ExplicitMemory`.

toString method

Description

Returns the string expression of the object. This method invokes `this.getName()`, and then returns the results.

Format

```
public String toString();
```

Parameters

None

Exceptions

None

Return value

This method returns the reference to the `String` type object that indicates the string expression of the object.

totalMemory method

Description

Returns the total reserved size of the Explicit memory block.

Format

```
public long totalMemory();
```

Parameters

None

Exceptions

InaccessibleMemoryAreaException:
  This functionality is not supported.

Return value

The common error check is performed, and then this method returns either of the following values:

0:
  Returned when the API processing cannot be performed.

Total reserved memory size (number of bytes) of the Explicit memory block indicated by the object:
  When the API processing can be performed, the memory size that can be used by the Explicit memory block indicated by the object is returned to the `long` type.

For details on common error check, see *10.6 Error check (common error check) of the process that controls the Explicit memory block* .

usedMemory method

Description

Returns the used memory size of the Explicit memory block.

Format

```
public long usedMemory();
```

Parameters

None

Exceptions

InaccessibleMemoryAreaException:

This functionality is not supported.

Return value

The used memory size (number of bytes) of the Explicit memory block indicated by the object is returned to the `long` type.

When it is judged by the common error check that the processing cannot be executed, `0` is returned. For details on common error check, see *10.6 Error check (common error check) of the process that controls the Explicit memory block* .

10.4 MemoryArea Class

Description

This is an abstract class that indicates the Explicit memory block or the Java heap. The package of the `MemoryArea` class is `JP.co.Hitachi.soft.jvm.MemoryArea`.

The methods included in the `MemoryArea` class are the abstract methods, and therefore, do not undergo any processing. For details on the processing of each method, see methods of the inherited classes with the same signature. The following table describes the format of each method and the references:

Format of methods and list of methods with the same signature

Method name	Format	Method with the same signature
`freeMemory` method	`public abstract long freeMemory();`	*freeMemory method*
`getName` method	`public abstract String getName();`	*getName method*
`newArray` method (format 1)	`public abstract Object newArray(Class type, int number);`	*newArray method (format 1)*
`newArray` method (format 2)	`public abstract Object newArray(Class type, int[] dimensions);`	*newArray method (format 2)*
`newInstance` method (format 1)	`public abstract Object newInstance(Class type);`	*newInstance method (format 1)*
`newInstance` method (format 2)	`public abstract Object newInstance(Class type, Object... args);`	*newInstance method (format 2)*
`newInstance` method (format 3)	`public abstract Object newInstance(java.lang.reflect.Constructor cons, Object... args);`	*newInstance method (format 3)*
`setName` method	`public abstract void setName(String name);`	*setName method*
`toString` method	`public abstract String toString();`	*toString method*
`totalMemory` method	`public abstract long totalMemory();`	*totalMemory method*
`usedMemory` method	`public abstract long usedMemory();`	*usedMemory method*

10.5 MemoryInfo Class

Description

You can acquire the memory information of garbage collection directly from a Java program.

For example, the space that is being currently used is calculated with the following expression:

`getXXXTotalMemory()-getXXXFreeMemory()`

The package of the `MemoryInfo` class is `JP.co.Hitachi.soft.jvm`.

List of methods

Method name	Function
`getEdenFreeMemory` Method	Acquires the available space in the Eden area.
`getEdenMaxMemory` Method	Acquires the maximum space used by the Eden area.
`getEdenTotalMemory` Method	Acquires the available space in the Eden area.
`getPermFreeMemory` Method	Acquires the available space in the Permanent area.
`getPermMaxMemory` Method	Acquires the maximum space used by the Permanent area.
`getPermTotalMemory` Method	Acquires the available space in the Permanent area.
`getSurvivorFreeMemory` Method	Acquires the available space in the Survivor area.
`getSurvivorMaxMemory` Method	Acquires the maximum space used by the Survivor area.
`getSurvivorTotalMemory` Method	Acquires the available space in the Survivor area.
`getTenuredFreeMemory` Method	Acquires the available space in the Tenured area.
`getTenuredMaxMemory` Method	Acquires the maximum space used by the Tenured area.
`getTenuredTotalMemory` Method	Acquires the available space in the Tenured area.

Usage example

The examples of method usage for acquiring memory information are as follows:

For obtaining the free size of Perm area

```
free_memory = JP.co.Hitachi.soft.jvm.MemoryInfo.getPermFreeMemory()
```

For obtaining the currently used Eden area

```
use_memory = JP.co.Hitachi.soft.jvm.MemoryInfo.getEdenTotalMemory()-
JP.co.Hitachi.soft.jvm.MemoryInfo.getEdenFreeMemory()
```

getEdenFreeMemory Method

Description

Acquires the available space in the Eden area.

Format

```
getEdenFreeMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the available space (number of bytes) in the Eden area as a long type.

getEdenMaxMemory Method

Description

Acquires the maximum space used by the Eden area.

Format

```
getEdenMaxMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the maximum space (number of bytes) used by the Eden area as a long type.

getEdenTotalMemory Method

Description

Acquires the available space in the Eden area.

Format

```
getEdenTotalMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the available space (number of bytes) in the Eden area as a long type.

getPermFreeMemory Method

Description

Acquires the available space in the Permanent area.

Format

```
getPermFreeMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the available space (number of bytes) in the Permanent area as a long type.

getPermMaxMemory Method

Description

Acquires the maximum space used by the Permanent area.

Format

```
getPermMaxMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the maximum space used by the Permanent area (number of bytes) as a long type.

getPermTotalMemory Method

Description

Acquires the available space in the Permanent area.

Format

```
getPermTotalMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the available space (number of bytes) in the Permanent area as a long type.

getSurvivorFreeMemory Method

Description

Acquires the available space in the Survivor area.

Format

```
getSurvivorFreeMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the available space (number of bytes) in the Survivor area as a long type.

getSurvivorMaxMemory Method

Description

Acquires the maximum space used by the Survivor area.

Format

```
getSurvivorMaxMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the maximum space (number of bytes) used by the Survivor area as a long type.

getSurvivorTotalMemory Method

Description

Acquires the available space in the Survivor area.

Format

```
getSurvivorTotalMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the available space (number of bytes) in the Survivor area as a long type.

getTenuredFreeMemory Method

Description

Acquires the available space in the Tenured area.

Format

```
getTenuredFreeMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the available space (number of bytes) in the Tenured area as a long type.

getTenuredMaxMemory Method

Description

Acquires the maximum space used by the Tenured area.

Format

```
getTenuredMaxMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the maximum space (number of bytes) used by the Tenured area as a long type.

getTenuredTotalMemory Method

Description

Acquires the available space in the Tenured area.

Format

```
getTenuredTotalMemory();
```

Parameters

None

Exceptions

None

Return value

This method returns the available space (number of bytes) in the Tenured area as a long type.

10.6 Error check (common error check) of the process that controls the Explicit memory block

When an enabled Explicit memory block is not specified, many APIs that operate the Explicit heap cannot be processed. In such a case, an error check routine common to all the APIs is defined, and it is determined whether the processing of an API can be executed or not. The common error check is used to determine whether or not an API can be processed depending on the state of the Explicit memory blocks that are processed by each API . The values returned by the common error check are as follows:

true:

Determines that the processing of an API can be continued. This value is returned when the state of the Explicit memory block that is to be processed is enabled.

false:

Determines that an API cannot be processed. This value is returned when the state of the Explicit memory block, that is to be processed, is disabled.

InaccessibleMemoryAreaException (exception class):

This exception has been thrown when an attempt was made to execute an unsupported functionality. For details on `InaccessibleMemoryAreaException` classes, see *10.7 Exception classes*.

This exception has been thrown when the Explicit memory block to be processed is in the following states:

- Released
- Reserved for release or explicitly reserved for automatic release
- State other than enabled, disabled, released, and reserved for release

10.7 Exception classes

The exception class that expresses the exceptions occurring in the APIs used in JavaVM is described below.

The following table describes a list of exception classes:

Table 10-2: Exception classes occurring in APIs used with JavaVM

Exception class name	Description
JP.co.Hitachi.soft.jvm.MemoryArea.MemoryManagementException	This is the base class of the exception classes defined in the `JP.co.Hitachi.soft.jvm.MemoryArea` package. This exception class is not assumed to be created or thrown in a Java program. The inherited class is `InaccessibleMemoryAreaException`.
JP.co.Hitachi.soft.jvm.MemoryArea.InaccessibleMemoryAreaException	This exception class has been thrown when an attempt was made to execute an unsupported functionality, for an instance of the `MemoryArea` class. An example is the delete operation executed for an instance of the `ExplicitMemory` class that is either already reserved for deletion or is already deleted. This exception class is not assumed to be created or thrown in a Java program. The base class is `MemoryManagementException`.

11 Properties that can be Used During Application Development

This chapter describes the properties that you can use while developing applications.

11.1 Properties that can be used in a batch application

This section describes the properties that you can use while developing the batch applications.

ejbserver.batch.currentdir property

Description

This property acquires the absolute path of the current directory in which the batch execution command (`cjexecjob`) is executed.

Use this property in the batch applications.

Usage example

The following is a usage example:

```
File f = new File(System.getProperty("ejbserver.batch.currentdir") +
System.getProperty("file.separator") + "DataFile.txt");
```

Appendixes

A. JavaAPI Classes in which Leakage of the Java Heap Memory Occurs Easily

When you use a JavaAPI class, the instances of the JavaAPI class are created in the Java heap memory. However, when the methods of the JavaAPI classes, described in Table A-1 and Table A-2, are used and the conditions described in the tables are fulfilled, the instances other than the JavaAPI classes will also be also created in the Java heap memory.

The created instances of the user-created classes are stored in the Java heap memory until the JavaAPI classes are deleted. Therefore, before you realize it, the usage of the Java heap memory might increase and leakage of the Java heap memory might occur.

However, as for the JavaAPI classes described in Table A-2, you can also delete the instances of the user-created classes.

The classes in which leakage of the Java heap memory occurs easily are listed in separate tables below, depending on the existence of the methods for deleting the instances. Table A-2 also describes the methods for deleting the user-created instances.

Reference note

Some changes might be made in the version 08-00 and later versions regarding the JavaAPI classes in which leakage of the Java heap memory occurs easily, and the conditions of leakage and the methods for deleting the instances of the user-created classes.

Table A-1: List of classes in which leakage of the Java heap memory occurs easily (When no deletion method exists)

JavaAPI class name	Conditions for creating instances of the user-created classes in the Java heap memory
`java.lang.ClassLoader`	When `ProtectionDomain`, specified in an argument of the `defineClass()` method, is correlated to a user-created class.
`java.net.URL`	When the `URLStreamHander` class, created by `URLStreamHandlerFactory` during the creation of the URL class instances, is a user-created class.
`java.text.DateFormat`	When `super()` is invoked by the constructor of a user-created class that inherits the `DateFormat` class.
`java.util.logging.Level`	When `super()` is invoked by the constructor of a user-created class that inherits the `Level` class.
`java.util.logging.LogManager`	When a user-created class, that inherits the `Logger` class in the argument of the `addLogger()` method, is specified.
`java.util.logging.Logger`	When the `getAnonymousLogger()` method and the `setParent()` method are invoked by a user-created class that inherits the `Logger` class.
`javax.accessibility.AccessibleBundle`	When a user class object is set up in a value of a pair that consists of a key and a value maintained in a user-implemented class corresponding to the resource bundle name specified in the argument of the `toDisplayString()` method.
`javax.print.attribute.standard.MediaSize`	When `super()` is invoked by the constructor of a user-created class that inherits the `MediaSize` class.
`java.rmi.server.UnicastRemoteObject`	When a user-created class is specified in the arguments (`RMIClientSocketFactory` and `RMIServerSocketFactory`) of the `exportObject()` method.

Table A-2: List of classes in which leakage of the Java heap memory occurs easily (When a deletion method exists)

JavaAPI class name	Conditions for creating instances of the user-created classes in the Java heap memory	Method for deleting the instances of the user-created classes
`java.beans.Introspector`	When a user-created class is specified in an argument (`beanClass`) of the `getBeanInfo()` method.	Execute the `flushCaches()` method and the `flushFromCaches()` method.

JavaAPI class name	Conditions for creating instances of the user-created classes in the Java heap memory	Method for deleting the instances of the user-created classes
`java.beans.PropertyEditorManager`	When a user-created class is specified in an argument (`editorClass`) of the `registerEditor()` method.	Specify `null` in `editorClass` of the `registerEditor()` method.
`java.util.logging.Logger`	When a user-created class that inherits the `Handler` class is specified in an argument of the `addHandler()` method.	Execute the `removeHandler()` method.
`javax.imageio.ImageReader`	When a user-created class that inherits the `IIOReadWarningListener` class is specified in an argument of the `addIIOReadWarningListener()` method.	Execute the `removeIIOReadWarningListener()` method.
	When a user-created class that inherits the `IIOReadProgressListener` class is specified in an argument of the `addIIOReadProgressListener()` method.	Execute the `removeIIOReadProgressListener()` method.
	When a user-created class that inherits the `IIOReadUpdateListener` class is specified in an argument of the `addIIOReadUpdateListener()` method.	Execute the `removeIIOReadUpdateListener()` method.
`javax.imageio.ImageWriter`	When a user-created class that inherits the `IIOWriteProgressListener` class is specified in an argument of the `addIIOWriteProgressListener()` method.	Execute the `removeIIOWriteProgressListener()` method.
	When a user-created class that inherits the `IIOWriteWarningListener` class is specified in an argument of the `addIIOWriteWarningListener()` method.	Execute the `removeIIOWriteWarningListener()` method.
`javax.naming.InitialContext`	When a user-created class is specified in an argument (`propVal`) of the `addToEnvironment()` method.	Execute the `removeFromEnvironment()` method.
`javax.naming.spi.NamingManager`	When a user-created class is specified in an argument of the `getContinuationContext()` method.	Implement `close()` of the acquired Context, and then execute `super()`.
`javax.print.attribute.HashAttributeSet`	When a user-created class is specified in an argument of the `add()` method.	Execute the `remove(Attribute)` method and `remove(Class)` method.

B. JavaAPI Classes that Implicitly Generate Threads inside JavaVM

Typically, Java SE APIs generate threads when the `java.lang.Thread` class or a thread-related class of the `java.util.concurrent` package is used.

However, some Java SE APIs implicitly generate threads. If you are using such APIs, you must take care because the number of threads might increase unintentionally, and the C heap area might be consumed. This section describes the APIs and functions that generate threads inside the JavaVM of Java SE 6.0.

B.1 Thread generation process list

This subsection describes threads that are generated inside JavaVM by the following APIs and functions:

- GUI related APIs
- JMX related APIs
- JNDI related APIs
- RMI related APIs
- Other APIs and functions

(1) GUI related APIs

The following threads are generated by GUI related APIs.

No specific conditions

If you use the GUI functionality of `AWT` or `Swing`, threads are generated. The GUI related APIs generate maximum six threads in a Java process.

`java.awt.EventQueue` class

One thread is generated for each `EventQueue` instance.

`java.awt.FileDialog` class

If you invoke the `show ()` method, one thread is generated. A maximum of one thread is generated for each `FileDialog` instance, when this method is invoked.

`java.awt.image.renderable.RenderableImageProducer` class

If you invoke the `startProduction()` method, one thread is generated.

`java.awt.print.PrinterJob` class

If you invoke the following methods, one thread is generated:

- `print()` (both types)
- `printDialog()` (both types)
- `pageDialog()` (both types)

`java.awt.TrayIcon` class

If the `java.awt.TrayIcon` instance is generated in a UNIX environment, one thread is generated for each instance.

`javax.swing.JEditorPane` class

One thread is generated for each `JEditorPane` instance.

`javax.swing.JFileChooser` class

One thread is generated for each `javax.swing.JFileChooser` instance.

`javax.swing.JTable` class

If you invoke `print()` (all five types), one thread is generated.

`javax.swing.`Timer

If you invoke the `start()` or `restart()` method, one thread is generated. A maximum of one thread is generated for each `Timer` instance, when this method is invoked.

`javax.swing.text.LayoutQueue` Class

If you invoke the `addTask()` method, one thread is generated. A maximum of one thread is generated for each `LayoutQueue` instance when this method is invoked.

`javax.swing.text.JTextComponent` class

When you invoke `print()` (all three types), one thread is generated.

`javax.swing.text.AsyncBoxView` class

If you invoke the following methods, a `LayoutQueue` instance is created inside the API, and a thread is generated in its extension:

- `preferenceChanged()`
- `replace()`
- `setSize()`

`javax.swing.text.html.FormView` class

If you invoke the `submitData()` method, one thread is generated.

Using `Applet`

If you use `Applet` in a UNIX environment and a warning icon is displayed, one thread is generated.

Using the `Input` method

If you use the `Input` method provided in `java.awt.im` and `java.awt.im.spi`, threads are generated. There can be a maximum of one such thread in a Java process.

(2) JMX related APIs

The following threads are generated by APIs related with JMX.

Resource monitoring in SNMP (Simple Network Management Protocol)

If you monitor or manage resources by using SNMP, maximum nine threads are generated.

javax.management.remote.rmi. RMIConnection interface

One thread is generated for each instance of the class that implements the RMIConnection interface.

javax.management.remote.rmi.RMIConnector class

If you invoke the `connect()` (both types) method, two threads are generated for one established connection.

(3) JNDI related APIs

The following threads are generated by APIs related with JNDI.

No specific conditions

Two threads are generated for one JNDI context by using the naming and directory operations.

`javax.naming.event.EventContext` interface

If you invoke the `addNamingListene()` (both types) method of the class that implements the Event Context interface, one thread is generated. Maximum one thread is generated for each directory context when this method is invoked.

(4) RMI related APIs

The following threads are generated by RMI related APIs:

RMI server-side

The following threads are generated inside the JavaVM.

- Maximum six threads are generated, when there are no specific conditions. These threads are maintained until the end of the JavaVM process.

- When waiting for connections from the RMI client, generate the threads equal to the number of TCP ports that have exported remote objects.
- If there is a method invocation from the RMI client, one thread is generated to control that method.
- One thread is generated for invoking the `unreferenced()` method of the remote object that implements the `java.rmi.server.Unreferenced` interface.

RMI client side

Threads equal to the number of connected RMI servers are generated.

(5) Other APIs and functions

The following threads are created by other APIs and functions:

HTTP/HTTPS communication

Two threads are generated to control `Keep Alive` in HTTP/HTTPS communication. There can be a maximum of two such threads in a Java process.

DNS communication

If you execute the DNS communication in a Windows environment, one thread is generated.

There can be a maximum of one such thread in a Java process.

Explicit execution of finalize

If you invoke the `runFinalization()` method of the `java.lang.System` class or `java.lang.Runtime` class, one thread is generated.

Creating an external processes

If you create external processes by using the `java.lang.ProcessBuilder` class in UNIX environment, one thread is generated. There can be a maximum of one such thread for each `java.lang.Process` instance.

`java.nio.channels.Selector` class

If you use the `Selector` class in Windows environment, one thread is generated each time the number of registrations of the channel to the `Selector` instance increases by 1,024.

`java.util.prefs.Preferences` class

If you use the `Preferences` class, one thread is created. There can be a maximum of one such thread in a Java process.

`javax.print.PrintServiceLookup` class

When you use `javax.print.PrintServiceLookup`, one thread is generated. There can be a maximum of one such thread in a Java process.

`sun.security.pkcs11.SunPKCS11` class

One thread is created for each `sun.security.pkcs11.SunPKCS11` instance.

Index