EUROPEAN SPACE AGENCY
DIRECTORATE OF OPERATIONS AND INFRASTRUCTURE
OPS-GI

CONFIGURATION AND INSTALLATION GUIDE

EGOS Data Dissemination System

Reference: EGOS-GEN-EDDS-CIG-1001
Version: 25.0
Date: 2025-03-07

Document Title:	Configuration and Installation Guide		
Document Reference:	EGOS-GEN-EDDS-CIG-1001		
Document Version:	25.0	**Date:**	2025-03-07
Abstract			

ESOC Approval Table:

Action	Name	Function	Signature	Date
Prepared by:	Manu Mohan	EDDS Team		2025-03-07
Verified by:	Florian Cleyssac	Quality Assurance Manager		2025-01-10
Approved by:	Rui Santos	ESA Technical Officer		2025-01-10

Authors and Contributors:

Name	Contact	Description	Date
Michael Hawkshaw	michael.hawkshaw@cgi.com	Author	2011-11-14
Rauno Ots	rauno.ots@cgi.com	Contributor	2011-09-09
Merlin Lobjakas	merlin.lobjakas@cgi.com	Contributor	2011-09-09
Kamill Panitzek	kamill.panitzek@cgi.com	Contributor	2016-02-04
Rokibul Uddin	rokibul.uddin@c-ssystems.de	Contributor	2017-05-05 2018-02-16 2018-09-14
Abhishek Annadanappa Chandrashekar	abhishek.annadanappa@c-ssystems.de	Contributor	2020-09-30 2021-05-31
Teresa Noviello	teresa.noviello@c-ssystems.de	Contributor	2020-09-30
Alessandro Afloarei	alessandro.afloarei@c-ssystems.de	Contributor	2021-12-14
Manu Mohan	manu.mohan@c-ssystems.de	Contributor	2022-10-31 2023-02-09 2023-09-18 2023-11-01 2024-02-27
Diwakar Kushwaha	diwakar.kushwaha@csgroup.de	Contributor	2024-05-08

Distribution List:

ESA

Document Change Log

Issue	Date	Description
1.0	2010-05-28	First Issue
1.1	2010-10-11	Issue for CDR
1.2	2011-03-18	Final Acceptance Issue
2.0	2011-09-09	Issue for EDDS v1.1.0i1
2.1	2011-10-24	Issue for EDDS v1.1.0i2
2.2	2011-11-14	Issue for EDDS v1.1.0i3
2.3	2012-02-13	Issue for EDDS v1.1.1i1
2.4	2012-04-04	Issue for EDDS v1.1.1i2
2.5	2012-06-25	Issue for EDDS v1.1.2i1
2.6	2012-09-17	Issue for EDDS v1.2.0i1
2.7	2012-09-27	Issue for EDDS v1.2.0i2
2.8	2012-10-12	Issue for EDDS v1.2.0i3
2.9	2013-03-21	Issue for EDDS v1.2.1i1
2.10	2013-04-10	Issue for EDDS v1.2.1i2
3.0	2013-06-21	Issue for EDDS v1.2.2i1
4.0	2013-12-03	Issue for EDDS v1.3.0i1
5.0	2014-05-28	Issue for EDDS v1.4.0i1
6.0	2014-12-17	Issue for EDDS v1.5.0i1
7.0	2016-02-04	Issue for EDDS v1.6.0i1
8.0	2016-07-04	Issue for EDDS v2.0.0i1
10.0	2016-10-19	Issue for EDDS v2.1.0i1
12.0	2017-05-05	Issue for EDDS v2.2.0i1
14.0	2018-02-16	Issue for EDDS v2.3.0i1
15.0	2020-02-11	Issue for EDDS v2.4.0i1
16.0	2020-09-30	Issue for EDDS v3.0.0i1
17.0	2021-02-15	Issue for EDDS v3.1.0i1
18.0	2021-12-15	Issue for EDDS v3.2.0i1
19.0	2022-04-28	Issue for EDDS v3.2.1i1
20.0	2022-10-31	Issue for EDDS v3.3.0i1
21.0	2023-02-09	Issue for EDDS v3.3.1i1
22.0	2023-11-03	Issue for EDDS v3.4.0i1
23.0	2024-03-08	Issue for EDDS v3.5.0i1
24.0	2025-01-10	Issue for EDDS v3.6.0i1
25.0	2025-03-07	Issue for EDDS v3.6.1i1

Document Change Record

DCR No:	01		
Date:	2024-03-07		
Document Title:	Configuration and Installation Guide		
Document Reference:	EGOS-GEN-EDDS-CIG-1001		
Page	**Paragraph**		
14, 78	3.2.6, 5.2.	EDDS-1396: Upgrade Tomcat to 9.0.100	

TABLE OF CONTENTS

1. INTRODUCTION 7

1.1 PURPOSE 7
1.2 SCOPE 7
1.3 DOCUMENT OVERVIEW 7

2. GLOSSARY & REFERENCES 8

2.1.1 Applicable documents 8

3. INSTALLATION INSTRUCTIONS 9

3.1 PREREQUISITES 9
3.2 DEVELOPMENT ENVIRONMENT 11
3.2.1 User Account 11
3.2.2 Using the EDDS Source Code CD 11
3.2.3 Gerrit Repository 11
3.2.4 Open Source Software 12
3.2.4.1 ESA Software 12
3.2.5 Populating edds-build.properties 12
3.2.6 Populating edds-deploy.properties 14
3.2.6.1 Examples 33
3.2.6.1.1 edds.darc.database.dataspace 33
3.2.6.1.2 edds.darc.activemq.broker 33
3.2.6.1.3 edds.darc.stream.topic 33
3.2.6.1.4 edds.parc.corba.ns.port 33
3.2.6.1.5 edds.parc.default.tm.datastream 33
3.2.6.1.6 edds.parc.default.tc.datastream 33
3.2.6.1.7 edds.parc.default.ev.datastream 34
3.2.6.1.8 edds.parc.tc.spidlist 34
3.2.6.1.9 edds.parc.ev.log.spidlist 34
3.2.6.1.10 edds.parc.ev.ool.spidlist 34
3.2.6.1.11 edds.parc.corba.server.family.name 34
3.2.6.1.12 edds.parc.corba.app.name 34
3.2.6.1.13 edds.farc.archive.instanceid 34
3.2.6.1.14 edds.farc.archive.zoneid 34
3.2.6.1.15 edds.farc.activemq.broker 34
3.2.6.1.16 edds.farc.activemq.broker.username 34
3.2.6.1.17 edds.farc.activemq.broker.password 34
3.2.6.1.18 edds.farc.subscription.topic 34
3.2.7 Apache Ant Note 34
3.2.8 Development Note 35
3.2.9 Performing the Compilation 35
3.2.10 Creating the Runtime 35
3.2.11 Ant Project Help 36
3.2.12 Creating the JavaDoc 36
3.2.13 Publich documentation in Nexus 36
3.2.14 Building and running JUnit Tests 37
3.2.15 Measuring code test coverage 37
3.2.16 Updating the EDDS version 37
3.3 DEPLOYMENT 37
3.3.1 EDDS Environment Variables 37
3.3.2 EDDS Database Installation 38
3.3.3 Configure LDAP 38
3.3.4 Configure ActiveMQ 39
3.3.5 Deploy EDDS Archiver 39
3.3.5.1 External connections 39
3.3.6 Deploy EDDS Delivery Manager 39
3.3.6.1 External connections 39
3.3.7 Deploy EDDS Web Service 39

3.3.7.1 EDDS Web Pages ... 40
3.3.7.2 External connections ... 41
3.3.8 Deploy MMI Java Web Start Libraries ... 41
3.3.9 Deploy EDDS MMI Web Application ... 42
3.3.9.1 External connections ... 42
3.3.10 Deploy EDDS Server ... 42
3.3.10.1 External connections ... 43
3.3.11 Deploy Request Submitter (optional) ... 43
3.3.12 Deploy Stream Client (optional) ... 43
3.3.13 Deploy Performance Processor (optional) ... 43
3.3.14 Deploy EDDS Migrator (optional) ... 44
3.3.14.1 External connections ... 44
3.4 FIREWALL CONFIGURATION ... 45
3.5 REQUEST ATTRIBUTES LABEL - "ALIAS" CONFIGURATION ... 46
3.6 RUNNING THE EDDS COMPONENTS ... 47
3.6.1 Start EDDS Archiver ... 47
3.6.2 Start Delivery Manager ... 47
3.6.3 Start the EDDS Server ... 47
3.6.3.1 Configuring EDDS Server memory usage ... 48
3.6.3.2 Configuring the user limit of inotify watches ... 49
3.6.4 EDDS Web Service ... 49
3.6.5 Start the Request Submitter ... 49
3.6.5.1 Request Submitter LDAP response read timeout ... 50
3.6.6 Start the Stream Client (optional) ... 50
3.6.7 Start the Performance Processor (optional) ... 51
3.7 EDDS MMI ... 52
3.7.1 Standalone EDDS MMI ... 52
3.7.1.1 Install the MMI ... 52
3.7.2 EDDS MMI Web Application ... 52
3.7.3 Runtime properties ... 52
3.8 PURE-FTP CONFIGURATION ... 54
3.9 CONFIGURATION FOR SFTP USAGE ... 54
3.10 CONFIGURING EDDS WEB SERVICES FOR SSL (HTTPS) ... 54
3.10.1 Configure the EDDS Web Server ... 54
3.10.1.1 Generate a key pair and store it ... 55
3.10.1.2 Get your public key certified by a Certificate Authority (CA) ... 55
3.10.1.3 Add a new connector to Tomcat ... 55
3.10.1.4 Exporting the certificate ... 56
3.10.2 Configure the EDDS client ... 56
3.10.2.1 Importing a certificate to truststore ... 56
3.10.2.2 Configuring EDDS Standalone MMI ... 56
3.11 LOGGING ... 57
3.11.1 Introduction ... 57
3.11.2 EDDS Web Server ... 57
3.11.3 EDDS Server ... 57
3.11.4 EDDS Delivery Manager ... 58
3.11.5 EDDS Request Submitter ... 58
3.11.6 EDDS Archiver ... 58
3.11.7 Audit log ... 58
3.12 INSTALLING SECURE FILE TRANSFER (SFT) FOR USE IN EDDS ... 59
3.13 HANDLING MULTIPLE MISSIONS ... 59
3.13.1 Configuration Scenarios ... 59

4. MISSION SPECIFIC CONFIGURATION ... 60

4.1 EGSCC MISSIONS ... 60
4.2 EDDS SERVER CONFIGURATION PROPERTIES ... 60
4.2.1 Introduction ... 60
4.2.2 EDDS Properties ... 60
4.3 USING MISSION SPECIFIC CONFIGURATION (MSC) FILES: ... 71

4.4 ENABLING OR DISABLING EDDS SERVICES FOR A MISSION 72
4.5 CONFIGURING THE EDDS SERVER TO RETRY TO PROCESS FAILED REQUESTS 73
4.6 XSL TRANSFORMATIONS 73
4.6.1 Transformation files 73
4.6.2 Email transformation 74
4.7 SCOS 2000 MIB REQUIRED FILES 75
4.7.1 Process ID File 75
4.7.2 Telemetry Packets Definition File 75
4.7.3 Command Subsystem Mapping File 75
4.7.4 Command Host File 75
4.7.5 MISCbase.ids File 75
4.7.6 MISCcontext_SHARED.sta File 75
4.7.7 CMDsourceTypeMapping.properties 75
4.7.8 CMD_ttRqstObqStatusMapping.properties 76

5. SETTING UP THE DEVELOPMENT ENVIRONMENT 78

5.1 INSTALLING THE REQUIRED SOFTWARE 78
5.2 CONFIGURING ANT 78
5.3 CONFIGURING ECLIPSE 78
5.4 NOW READ JAUTODOC TEMPLATES 80
5.5 CHECKING OUT THE SOURCE CODE 80
5.6 FORMATTING 81
5.7 CODE TEMPLATES 81
5.8 RUNNING MMI FROM DEVELOPMENT ENVIRONMENT 82
5.8.1 Setting up Run configuration 82
5.8.1.1 Create a new run configuration manually 82
5.9 ISSUING AN EDDS RELEASE 82
5.9.1 Release preparation 82
5.9.2 Creating the CDs 82

APPENDIX A PACKET DATA 84

A.1 CREATING THE HEADER STRUCTURE INFORMATION FILES FOR TM 84
A.2 CREATING THE HEADER STRUCTURE INFORMATION FILES FOR TC 84
A.3 CREATING THE HEADER STRUCTURE INFORMATION FILES FOR DATAPROVISION TC 85
A.4 DEFINING THE TC PACKET STRUCTURE 85
A.5 DEFINING DATAPROVISION TC PACKET STRUCTURE 88
A.6 CREATING THE HEADER STRUCTURE INFORMATION FILES FOR EV 88

APPENDIX B CONFIGURING SMON SERVER FOR USE WITH EDDS 89

APPENDIX C PURE-FTP INSTALLATION 90

C.1 PREREQUISITES 90
C.2 INSTALLATION 90

APPENDIX D APACHE ACTIVEMQ CONFIGURATION 91

D.1 CONFIGURING ACTIVEMQ 91
D.2 ENABLING PASSWORD PROTECTION ON ACTIVEMQ 91
D.3 CONFIGURING EDDS TO USE ACTIVEMQ PASSWORD 93
D.4 DEBUGGING ACTIVEMQ DATABASE CONTENTS AND CONNECTIONS 93
D.5 IMPORTANT NOTE ON CLOCK SYNCHRONISATION 93

APPENDIX E FARC SUBSCRIPTIONS 95

APPENDIX F CREATING RSA PUBLIC/PRIVATE KEY 97

APPENDIX G COLD BACKUPS 99

APPENDIX H LDAP CONFIGURATION 100

H.1 CREATING A BACKUP OF LDAP DATA 100
H.2 IMPORTING A PREVIOUS BACKUP INTO LDAP 100
H.3 DEPLOYING MULTIPLE MASTER AND SLAVE LDAP SERVERS 101
H.3.1 LDAP Master Prime 101
H.3.2 LDAP Master Backup 101
H.3.3 LDAP Slave Prime 102
H.3.4 LDAP Slave Backup 102

APPENDIX I USING EDDS WITH AN EXISTING CENTRAL LDAP INSTALLATION 103

APPENDIX J EXAMPLE REQUESTS FOR THE REQUEST SUBMITTER 104

J.1 CANCEL REQUESTS 104
J.2 SUSPEND REQUESTS 104
J.3 RESUME REQUESTS 105

APPENDIX K DEVELOPMENT ENVIRONMENT RESOURCES 106

K.1 FORMATTING.XML 106
K.2 CODETEMPLATE.XML 112
K.3 JAUTODOC 115
K.3.1 File Header 115

APPENDIX L EDDS DEVOPS PROJECT CONFIGURATION 117

L.1 CONFIGURING MISSION FOR EDDS DEPLOYMENT 117
L.1.1 Configuring EDDS Server 117
L.1.2 Configuring Delivery Manager 117
L.1.3 Configuring Request Submitter 118
L.1.4 Configuring the deployment 118

FIGURE 1 REQUEST NAME MAPPING CONFIGURATION 46
FIGURE 2 CHECKING OUT THE SOURCE CODE 81

1. Introduction

1.1 Purpose

This document contains guidelines on the use of the Configuration and Installation of the EGOS Data Dissemination System and its pre-required subsystems and libraries.

1.2 Scope

This document is intended for the following readers:

ESOC System Administrators

EDDS Developer

1.3 Document Overview

Section 1: Introduction and document details

Section 2: Document References and Glossary

Section 3: Build and Deployment of the EDDS sub-systems.

Section 4: Comprehensive mission specific configuration of EDDS

Section 5: Setting up the Development Environment

Appendix A: Packet Data

Appendix B: Configuring SMON Server for use with EDDS

Appendix C: Pure-FTP Installation

Appendix D: Apache ActiveMQ Configuration

Appendix E: FARC Subscriptions

Appendix F: Creating RSA Public/Private Key

Appendix G: Cold Backups

Appendix H: LDAP Configuration

Appendix I: Using EDDS with an Existing Central LDAP Installation

Appendix J: Example Requests for the Request Submitter

Appendix K: Development Environment Resources

2. Glossary & References

See Glossary[AD-8]

2.1.1 Applicable documents

Ref.	Document Title	Issue and Revision, Date
[AD-1]	EGOS-GEN-EDDS-SRS-1001	Issue 23 Revision 0, 2021-12-03
[AD-2]	ESOC Generic Ground Systems: System Configuration Baselines: Part 3-Baseline Definitions [EGGS-ESOC-GS-SCD-1002]	Version 1.3, 2011-08-04
[AD-3]	Apache Tomcat 9.0 - SSL Configuration HOW-TO https://tomcat.apache.org/tomcat-9.0-doc/ssl-howto.html	2016-05-17
[AD-4]	JAutodoc - Eclipse Plugin http://jautodoc.sourceforge.net/	2011-01-21
[AD-5]	EGOS-GEN-FARC-ICD-1001_i4r5 - FARC Client Services API Issue 4 Revision 5	2012-10-31
[AD-6]	EGOS-GEN-EUD-CG-0001 - EUD2 Configuration and Installation Guide	Version 1.5, 2012-05-02
[AD-7]	EGOS-GEN-EDDS-EUG-1001	Version 16.0, 2021-12-03
[AD-8]	EGOS-GEN-EDDS-GLO-0001	Version 10.0, 2021-12-03

3. Installation Instructions

The following sections gives a description of the installation procedures and steps needed to create a development environment and/or run-time deployment of the EDDS components.

3.1 Prerequisites

In order to extract, compile and deploy EDDS the following products must be installed and configured. The installation and configuration of these products is not within the scope of the EDDS installation.

The following list contains the required pieces of software that should be installed prior to commencing the installation of EDDS.

- **SLES 15** Installed
- **Java** installed. See section 3.2.5 for more details.
  - For compiling the source code, OpenJDK 11 (11.0.20+9) is required.
  - For running the software, JRE 11 (11.0.20+9) is required.
- **Apache Tomcat** installed (See SLES 15 Baseline [AD-2] for correct version)
  - **Supported by:**
    - SuSE Linux Enterprise Server (SLES) x86 64-bit version 15 with Service Pack 1 plus additional patches (SLES15-064-ESOCL01-i3r1).
  - Management Console enabled.
  - The Tomcat application must be reachable via HTTP from the computer on which the build and deployment is being executed (in order to allow the ANT script to automatically deploy the WAR archive on the web server through the manager application).
  - Modify the property file *{TOMCAT_HOME}/conf/tomcat-users.xml*. Add to it a user that has permissions to deploy an application. A basic configuration would be:

    <tomcat-users>

    <role rolename="manager-gui"/>

    <user username="edds" password="manager" roles="manager-gui"/>

    </tomcat-users>
  - For setting up HTTPS for Tomcat see section 3.10.1.
  - In addition to the password restrictions, the Tomcat Manager web application should be restricted by the remote IP address or host by adding a RemoteAddrValve or RemoteHostValve. See Apache Tomcat Valve Component documentation for details (https://tomcat.apache.org/tomcat-9.0-doc/config/valve.html). Here is an example of restricting access just to the localhost by IP address by editing the file *{TOMCAT_HOME}/webapps/manager/META_INF/context.xml*:

```
<Context privileged="true">

      <Valve
className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.0\.0\.1"/>

</Context>
```

- **MariaDB** should be installed (See SLES 15 Baseline [AD-2] for correct version)
  - **Supported by:**

        - SuSE Linux Enterprise Server (SLES) x86 64-bit version 15 with Service Pack 1 plus additional patches (SLES15-064-ESOCL01-i1r0).
    - The maximum packet size should be increased in /etc/my.cnf by changing max_allowed_packet to 50M.

- **Apache Maven** should be installed and available on the PATH environment variable.
- **Apache Ant** should be installed and available on the PATH environment variable.
    - **Supported by:**
        - SuSE Linux Enterprise Server (SLES) x86 64-bit version 15 with no Service Pack (SLES12-064-ESOCL00-i1r0).
        - SuSE Linux Enterprise Server (SLES) x86 64-bit version 15 with Service Pack 1 plus additional patches.SLES15-064-ESOCL01-i1r0
    - The *ANT_HOME* environment variable should be set containing the path to the location of where Ant is installed:

      e.g. export ANT_HOME=/usr/apache-ant-1.9.10/
    - *The ANT_OPTS environment variable has be set allowing a 256mb Java Heap Space for ANT*

      e.g. export ANT_OPTS=-Xmx512m
- **Apache ActiveMQ** should be installed with the broker running (See SLES 15 Baseline [AD-2] for correct version). See Section Appendix D for configuration information.
- **OpenLDAP** should be installed (See SLES 15 Baseline [AD-2] for correct version)
    - Should be installed as part of the SLES 15 installation.
    - **Supported by:**
        - SuSE Linux Enterprise Server (SLES) x86 64-bit version 15 with no Service Pack (SLES15-064-ESOCL00-i1r0)
        - SuSE Linux Enterprise Server (SLES) x86 64-bit version 15 with Service Pack 1 plus additional patches (SLES15-064-ESOCL01-i3r1).
- **Pure-FTPd** is recommended to be installed (See SLES 15 Baseline [AD-2] for correct version)
    - Should be installed as part of the SLES 15 installation.
    - Should have been built with the `--with-virtualchroot` and `–with-ldap` options. See Appendix C for more information.
    - See Section 3.8 for additional configuration instructions
    - Other FTP server programs may be used, but the functionality may be limited (e.g. users can only use the “Private” privacy tag option). Users would not then be authenticated with the EDDS LDAP database, and so they would have to be created separately.
- **GFTS** is recommended to be installed if the EDDS Server and Delivery Manager are running on different machines. To use SFT, see section 3.12 of this document.

This guide takes the approach of having a single build/deployment repository which each of the EDDS components will be deployed. During deployment the deploy location must be available as a path to the EDDS scripts. If it is the case that the EDDS components are spread over a number of machines that it is recommended that cross mounted directories are created to point to each of the remote file systems.

The table below shows the software that is required to develop, compile and build and deploy EDDS. A plus (+) indicates that the software is required, and a minus (-) indicates that it is not required. A plus or minus in brackets indicates that the component is optional.

	Development	**Compile/Build**	**Deploy/Runtime**
SLES 15	+	+	+
Apache Tomcat	+	-	+
MariaDB	+	-	+
Maven	+	+	-
Apache Ant	+	+	+
Apache ActiveMQ	+	-	+
OpenLDAP	+	-	+
GFTS or SFT	(+)	(-)	(+)
FTP Server	+	_	+
Java OpenJDK 11	+	+	-
Java 11 JRE	-	-	+

Table 3-1: EDDS Required Software

3.2 Development Environment

This section provides the instructions to extract the source code and build the EDDS project. This section finishes with a basic configured deployment of the EDDS environment. More detailed configuration of the various EDDS properties can be found in Section 3.13.

3.2.1 User Account

Before commencing with the installation of EDDS a user account must be configured that has access to the machine on which EDDS will be installed.

For the remainder of this Configuration Guide, it is assumed that a user account has been created with the username “edds” and the user’s home directory is located at */lhome1/edds*

3.2.2 Using the EDDS Source Code CD

The EDDS Source Code CD can be used to build the EDDS software so that the resulting runtime folder can be used to deploy EDDS either on the same machine, or on a different machine that might not have all the tools available to build the EDDS source code.

The source code is supplied as a compressed TAR file. To extract the source code, navigate to the directory you would like to extract the source code to and run the following command from the SLES 15 command line:

```
tar zxvf EDDS_SOURCE_DIST.tar.gz
```

If the tar file is on a CD and has not been copied to the same directory where the command is executed, it will be necessary to add the full path to the tar file, e.g. */media/cdrom/EDDS_SOURCE_DIST.tar.gz*

3.2.3 Gerrit Repository

This step is only applicable if the source CD is not available or if building from the development environment. The EDDS source code is contained within a Git repository hosted by Gerrit which must be cloned to set up a development or build environment. To clone the Git repository, log into Gerrit using a web browser and the username and password that has been provided to you. Select “Projects” and

“List”. Select the EDDS project. Gerrit will provide you with the command to run to clone the repository. It is then possible to use the regular git commands to check-out the branch or tag to use.

3.2.4 Open Source Software

EDDS makes use of a number of Open Source Software (OSS) and Third Party Products that are obtained through Maven.

3.2.4.1 ESA Software

ESA Products that EDDS depends on have official Maven releases and they are available on an ESA Nexus repository where access is controlled. The dependencies are pulled in by Maven during the build process. There is no need to manually download any libraries for EDDS to build properly.

DARC Note: Always use the DARC-dataprovision-3.0.0.jar file with EDDS v3.6.0, even when connecting to DARC v2.2.2 or earlier. When connecting to DARC v2.2.2 or earlier, be sure to set the property “edds.darc.post.2.3.0” in edds-deploy.properties to false.

3.2.5 Populating edds-build.properties

Before performing compilation the *edds-build.properties* file should be modified in the root EDDS/edds directory. The following table summarises the properties that should be modified.

Property	**Description**
edds.version	Should be updated to set the version number of EDDS to be built (e.g. 2.0.0). This is appended to the end of the EDDS Jar files.
edds.java.version	The version of Java that the generated class files should be compatible with. This should be set to 11.
edds.runtime.dir	After compilation it is possible to generate a runtime distribution of the EDDS product that can then be used to deploy all components of the system. This property should be the path to the folder of where this should be created.
edds.javaws.signjar.keystore	Keystore location for signing Java WS libs. e.g. /lhome1/edds/.keystore For generating a keystore and key for signing Java *keytool* can be used. Also see 3.10.1.
edds.javaws.signjar.keystore.type	The type of the above keystore. e.g. jks
edds.javaws.signjar.storepass	Password for the keystore. e.g. changeit
edds.javaws.signjar.alias	Alias of the key for signing. e.g. eddsjws
edds.javaws.signjar.keypass	Password for the above key. e.g. password

edds.javaws.server	EDDS MMI Java Web Start server (accessed from the outside) with protocol and optionally port. e.g. http://10.48.18.159:8080 This will be used in the JNLP files to point where to download the updated libraries from.
edds.javaws.context	EDDS MMI Java Web Start server deploy path. The context path to deploy the edds-javaws.war to. e.g. /javaws
edds.javaws.proxyhost	The HTTP proxy host to use if the MMI needs to connect to the server through a proxy. If no proxy connection is used (direct connection), leave this property blank This can be changed later by downloading the main.jnlp file (clicking the Java Web Start link on the web page) and editing it, then running it.
edds.javaws.proxyport	The HTTP proxy port to use if the MMI needs to connect to the server through a proxy. If no proxy connection is used (direct connection), leave this property blank This can be changed later by downloading the main.jnlp file (clicking the Java Web Start link on the web page) and editing it, then running it.
edds.javaws.proxyuser	The HTTP proxy username to use if the MMI needs to connect to the server through a proxy. If no proxy connection is used (direct connection), leave this property blank This can be changed later by downloading the main.jnlp file (clicking the Java Web Start link on the web page) and editing it, then running it.
edds.javaws.proxypassword	The HTTP proxy password to use if the MMI needs to connect to the server through a proxy. If no proxy connection is used (direct connection), leave this property blank This can be changed later by downloading the main.jnlp file (clicking the Java Web Start link on the web page) and editing it, then running it.
edds.rest.api.jwt.encryption.key	This is the secret token used to sign the token generated for the JWT REST API authentication for clients. The REST API clients passes the generated token as part of the API request and the edds web server will identify it as a valid request because the request is signed with this unique secret token.
jdk.home.dir	This optional property is needed by the deploy-environment-settings Ant target (Section 3.3.1) and should point to the home directory of the Java 11 JDK installation. This property should only be specified when using the Ant target “deploy-environment-settings” for use in the build environment. For the target runtime environment, ensure this property is empty or commented out. *e.g.* */usr/java/jdk11*

Table 3-2: Populating edds-build.properties

Note that for Java Web Start, the JNLP files must be signed during the build process, and as a result, the properties cannot be changed in the runtime distribution if they are incorrect.

3.2.6 Populating edds-deploy.properties

Throughout the deployment process the build scripts use a number of properties from a property file. Before commencing, the *edds-deploy.properties* file must be updated with values required for installation. After extraction has completed in the previous step the property file can be found in the directory:

{EDDS Runtime Dist Dir}

The following table lists the properties contained within the file, and their description. Where provided the example value represents the recommended location or suggested value. It is recommended that when entering the name of a server, that the IP address is used rather than the domain name.

Note that this list contains only the properties you are most likely to need to change. Each application, once deployed, contains a properties file that contains more properties that can be changed as required if the defaults are not correct. For the complete list of properties, see Section 4.2.

Property	Description
edds.mission.name	The name of the mission in LDAP that this EDDS Server instance will process requests for. If the mission does not yet exist in LDAP, a warning will be shown in the logs, but the server will continue to wait for requests for the mission specified.
edds.completedfiles.owneronly.writable	Flag as to whether files created by EDDS Server should be writable only by the user running the server. It may be necessary to set this flag to false if you are using software to move files out of the EDDS Server completed directory to the Delivery Manager's inbox, and this process is running under a different user account. e.g. *false*
edds.db.url	The hostname/IP address of the computer on which the EDDS MariaDB instance is running. e.g. *localhost:5080*
edds.db.name	The name of EDDS database in MariaDB instance. e.g. *edds*
edds.db.admin.user	The username of a database user with administrative privileges on the MariaDB database that can be used to create the database schema. e.g. *root*
edds.db.admin.password	The password of the database user with administrative privileges on the MariaDB database that can be used to create the database schema *e.g. password*

Property	Description
edds.db.prod.user.user	The username of the database user that will be used by EDDS when the application is running e.g. *edds*
edds.db.prod.user.password	The password of the database user that will be used by EDDS when the application is running e.g. *edds*
edds.tomcat.home	The directory of the tomcat home directory *e.g. /lhome1/edds/apache-tomcat-9.0.100* This is used for the deployment of the EDDS Web Server and EDDS Java Web Start libs.
edds.ws.max_incorrect_logins	The maximum number of incorrect logins before the user account is suspended. e.g. 3 This applies to all missions. This option is only used by EDDS Web server.
edds.ws.user.password.expiry.period	The default period after which a user's password expires and must be changed. This setting is taken if a user is created or updated and the expiry period has not been set. The minimum period is 1 day (P1D). e.g. P6M (6 months)
edds.ws.number.of.passwords.to.check	Number of previously used passwords which are checked when user changes password. New password has to be different from the checked passwords. Any value below 1 will disable password uniqueness check. Maximum value is 5. Default value is 3.
esa.egos.edds.max.scheduled.requests	The maximum number of scheduled requests that can be generated in a repeating schedule. Default value is 100. Both EDDS Archiver and the EDDS WebServer use this value, make sure if this is changed on one component, it should be changed to the same value on the other component as well.
edds.delivery.manager.home	The location of where the EDDS Delivery Server should be deployed to. e.g. */lhome1/edds/EDDS_RUNTIME/DELIVERY_MGR*
edds.delivery.manager.inbox	The location where files the Delivery Manager should process are placed (usually the EDDS Server outbox when the two servers are co-deployed) e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_HOME/edds-server-completed

Property	**Description**
edds.delivery.manager.working	The working directory for the Delivery Manager where files are placed temporarily e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_HOME/delivery-manager-working
edds.delivery.manager.failed	The failed directory for the Delivery Manager where files are placed when the delivery fails. The files can be placed into the Delivery Manager's inbox to initiate a re-delivery attempt. e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_HOME/delivery-manager-failed
edds.server.home	The location of where the EDDS Server should be deployed to. e.g. */lhome1/edds/EDDS_RUNTIME/EDDS_SERVER*
edds.server.supported.batch.requests	The type of batch requests that this EDDS Server instance will process. This should be a comma separated list with each item in single quotes ('). The possible values can be obtained from DataAccessData element from the EDDS schema file "common.xsd" within edds-ws-common/src/main/wsdl e.g. 'Param','EgsccParam','ParamStatistics','EgsccParamDefinition','ParamDefinition','ParamPreview','EgsccPktTcReport' PktEv','EgsccPktTmReport','EgsccPktTm','EgsccPktTmRaw','PktTm','PktTc','PktEvRaw','PktTmRaw', 'PktTcRaw','PktTmGapReport','PktTmStatistics','PktTcStatistics','PktEvStatistics', 'PktTmReport','PktTcReport','EventRecordReport', 'ArchiveCatalogue','ArchiveFile','ArchiveSubscription', 'OolRecordReport','EddsUsageReport','FileSystemFileCatalogue','FileSystemFolderCatalogue','FileSystemFile','FileSystemSubscription', 'EgsccActivityReport'
edds.server.supported_data_sources	The data source requests that this EDDS Server will serve. e.g. 'DARC','PARC','HARC'

Property	**Description**
edds.server.supported.stream.requests	The type of stream requests that this EDDS Server instance will process. This should be a comma separated list with each item in single quotes (‘). The possible values can be obtained from the EDDS schema file “common.xsd” within edds-ws-common/src/main/wsdl e.g. 'ParamStream','PktTmStream’,'PktTcStream’,'PktEvStream’,'OolStream’
edds.server.enable.performance.test	If true debug messages for performance testing will be output e.g. false
edds.server.enable.egscc	If true enable egscc integration. False otherwise.
edds.server.workingdir	The directory where files being worked on are stored for the EDDS Server e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_HOME/edds-server-working
edds.server.outbox	The directory where completed files are placed for the EDDS Server e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_HOME/edds-server-completed
edds.server.streaming.threadpool.size	The number of threads on the Camel route for consuming messages on the topic for each stream request. Increase this value if EDDS is not consuming messages fast enough, but be aware memory usage will increase. e.g. 5
edds.server.streaming.max.threadpool.size	The maximum number of threads on the Camel route for consuming messages on the topic for each stream request. Increase this value if EDDS is not consuming messages fast enough, but be aware memory usage will increase. e.g. 10

Property	Description
edds.server.streaming.corba.maxproc.period	The maximum number for Corba processing period. It limits the periodic processing time during which EDDS receives and buffers messages received from S2K at the CORBA interface. The smaller the value, the smaller the buffered messages size at the CORBA interface. Required to be 5, up to S2K 7, baseline omniORB 4.2.0-1.1 e.g. 5
edds.server.streaming.corba.max_messages_amount	It limits the max amount of messages to be retrieved from the data provision, if available e.g. 1000
edds.server.submitted.recovery.frequency	Time in minutes. Edds-server will try to recover pending SUBMITTED requests every 'frequency' minutes. The request will be processed if the request's execution time is higher than bufferzone time. e.g 45 Means that every 45 minutes Edds will check for submitted requests, if it finds any request with execution time older than bufferzone minutes that has not been processed, it will be processed immediately.
edds.server.submitted.bufferzone.time	Bearing time in minutes for the recovery of SUBMITTED requests. The purpose of this time is to avoid processing of too recent SUBMITTED requests. e.g. 30 In this case EDDS will ignore SUBMITTED requests younger than 30 minutes from being recovered.
edds.egos.edds.jms.receive.timeout	The number of milliseconds for the Web Server to wait for the EDDS Server components to respond to a request for information before informing the user that the request could not be completed. e.g. 5000
edds.request.submitter.home	The location of where the Request Submitter will be deployed to. e.g. */lhome1/edds/EDDS_RUNTIME/REQUEST_SUBMITTER*

Property	Description
edds.request.submitter.username	The 'username' of the account created for this instance of the File Request Submitter. *This property is used also by Stream Client e.g RS_USER This property must match the username for the user that will be created for the request submitter to issue requests with.
edds.request.submitter.password	The 'password' of the account created for this instance of the File Request Submitter. * This property is used also by Stream Client e.g 1Password This property must match the password for the user that will be created for the request submitter to issue requests with.
edds.request.submitter.role	The name of the role assigned to the user account.
edds.request.submitter.domain	The domain assigned to the default mission.
edds.request.submitter.gdds.mission	The name of the mission that the GDDS requests will be submitted to.
edds.request.submitter.server.address	The endpoint address of the EDDS Web Server. * This property is used also by Stream Client e.g http://10.48.29.159:8080/edds/EddsService?wsdl
edds.request.submitter.edds.schema	The path to the eddsSchema file. e.g *edds.xsd*
edds.request.submitter.gdds.xslt	The path to the gddsXslt transformation file e.g gdds/GDDS2EGGS.xsl
edds.request.submitter.inbox	The directory to poll for new requests e.g *_/lhome1/edds/EDDS_RUNTIME/EDDS_HOME/submitter-inbox_*
edds.request.submitter.outbox	The directory to place completed requests e.g. *_/lhome1/edds/EDDS_RUNTIME/EDDS_HOME/submitter-completed_*

Property	Description
edds.request.submitter.failed	The directory to place failed requests e.g *lhome1/edds/EDDS_RUNTIME/EDDS_HOME/submitter-failed*
edds.stream.client.home	The location of where the Stream Client will be deployed to. e.g. */lhome1/edds/EDDS_RUNTIME/STREAM_CLIENT*
edds.stream.client.outbox	The directory to place completed files saved from the stream e.g. */lhome1/edds/EDDS_RUNTIME/EDDS_HOME/stream-outbox*
edds.archiver.home	The directory to deploy the EDDS Archiver to. e.g. */lhome1/edds/EDDS_RUNTIME/EDDS_ARCHIVER*
edds.performance.processor.home	The location of where the EDDS Performance Processor will be installed e.g. */lhome1/edds/EDDS_RUNTIME/ EDDS_PERFORMANCE_PROCESSOR*
edds.migrator.home	The location of where the EDDS Migrator application will be installed e.g. */lhome1/edds/EDDS_RUNTIME/EDDS_MIGRATOR*
edds.darc.database.host	The database hostname/IP address of the computer that contains the DARC database *e.g. localhost:3306*
edds.darc.database.schema	The database schema name of the DARC database. *e.g. DARC_DB*
edds.darc.database.dataspace	The DARC Dataspace to use. *e.g. DEFAULT*
edds.darc.database.user	The username that would be used by EDDS to access the DARC database *e.g. DARC_USER*
edds.darc.database.password	The password that EDDS will use to access to the DARC database *e.g. password*

Property	Description
edds.darc.post.2.3.0	Whether DARC 2.3.0 or newer is used as the back-end. How to interpret timestamps received from the DARC. For DARC v2.3.0 and later, this should be set to true, as the timestamps are stored to microsecond precision. For DARC v2.2.2, this should be set to false, as timestamps are stored to millisecond precision. *e.g. true*
edds.darc.activemq.broker	The DARC ActiveMQ Message Broker to use for receiving streamed live parameter data. If the DARC version used does not support streaming, leave the value at its default as it is not used. *e.g. failover:(tcp://10.48.18.86:61616)?timeout=3000&amp;trackMessages=true*
edds.darc.activemq.broker.username	The DARC ActiveMQ Message Broker username *e.g. system*
edds.darc.activemq.broker.password	The DARC ActiveMQ Message Broker password *e.g. password*
edds.darc.stream.topic	The name of the topic that the parameters are placed on from the DARC *e.g. esa.egos.darc.realtimeParams*
edds.darc.stream.simulator.paramnames	Used by the DARC Parameter stream simulator. Lists the possible parameter names that the simulator can randomly choose from. *e.g. CMDMOD,DPTV,JOGS0001,LOGS0001,OBSMDADD,S2KSPS10*
edds.provision.packet	The name of the Packet Data Provision Service in the CORBA name service. This is usually "PacketProvisionService_" followed by the hostname of the machine running SCOS. e.g. *PacketProvisionService_gimus106*
edds.provision.command	The name of the Command Data Provision Service in the CORBA name service. This is usually "CommandProvisionService_" followed by the hostname of the machine running SCOS. e.g. *CommandProvisionService_gimus106*

Property	Description
edds.provision.event	The name of the Event Data Provision Service in the CORBA name service. This is usually "EventProvisionService_" followed by the hostname of the machine running SCOS. e.g. *EventProvisionService_gimus106*
edds.provision.ool	The name of the OOL Data Provision Service in the CORBA name service. This is usually "OolProvisionService_" followed by the hostname of the machine running SCOS. e.g. *OolProvisionService_gimus106*
edds.parc.corba.ns.hostname	The name server hostname / IP Address for the S2K Name Server instance for PARC access only used for PacketTCStructureHelper e.g. *localhost*
edds.parc.corba.ns.hostnames	The list of name server hostnames / IP Addresses for the S2K Name Server instances for PARC access e.g. *ip:port, ip:port, ip:port*
edds.provision.streaming.packet	The name of the Packet Data Provision Service in the CORBA name service for streaming live data ONLY. This is usually "PacketProvisionService_" followed by the hostname of the machine running SCOS. e.g. *PacketProvisionService_gimus106*
edds.provision.streaming.command	The name of the Command Data Provision Service in the CORBA name service for streaming live data ONLY. This is usually "CommandProvisionService_" followed by the hostname of the machine running SCOS. e.g. *CommandProvisionService_gimus106*
edds.provision.streaming.event	The name of the Event Data Provision Service in the CORBA name service for streaming live data ONLY. This is usually "EventProvisionService_" followed by the hostname of the machine running SCOS. e.g. *EventProvisionService_gimus106*
edds.provision.streaming.ool	The name of the OOL Data Provision Service in the CORBA name service for streaming live data ONLY. This is usually "OolProvisionService_" followed by the hostname of the machine running SCOS. e.g. *OolProvisionService_gimus106*
edds.parc.corba.ns.port	The port of the s2k CORBA naming service for PARC access. Only used for PacketTCStructureHelper *e.g. 1050*

Property	**Description**
edds.parc.corba.server.family.name	The name of the CORBA Server family for the PARC. Can be found in MISCcontext under OPERATIONAL_SERVER_FAMILY e.g. *PRIME*
edds.parc.streaming.corba.ns.hostnames	The name server hostnames / IP Addresses for the S2K Name Server instances for PARC access for streaming live data ONLY. e.g. *ip:port, ip:port, ip:port*
edds.parc.streaming.corba.server.family.name	The name of the CORBA Server family for the PARC for streaming live data ONLY. Can be found in MISCcontext under OPERATIONAL_SERVER_FAMILY e.g. *PRIME*
edds.parc.corba.app.name	The name of the PARC Manager application as registered in the Name Server e.g. *PAMASviews*
edds.parc.corba.archive.management	The name of the PARC Archive Management service as registered in the Name Server e.g. *PAMASmanageArchive*
edds.parc.dataspace	The PARC dataspace to use if one isn't specified in the request (i.e. "EDDS Default" is selected in the EDDS MMI) and the edds.parc.dataspaces.enabled property is set to true This property can be commented out or left empty. In which case, the PARC will use the data space set as current. e.g. *(empty)*
edds.parc.dataspaces.enabled	Is the new PARC API used that has the data spaces functionality enabled e.g. *true*
edds.parc.bigendian.enabled	If set to “true”, then the output from the PARC will be big endian. Only supported in SCOS 5.5.4 or later. For all earlier versions, ensure this is set to “false” otherwise packet requests will fail. The human readable report formats are not affected by this setting. e.g. *false*

Property	**Description**
edds.parc.pkttc.structure.file	Defines the location of the xml file which describes the TC packet structure e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_SERVER/config/MIB/StandardSCOS55.xml
edds.parc.default.tm.datastream	The default data streams for TM packets (comma separated list) *e.g. 65535,65534,4,248,4000*
edds.parc.default.tc.datastream	The default data streams for TC packets (comma separated list) *e.g. 65535*
edds.parc.default.ev.datastream	The default data streams for EV packets (comma separated list) *e.g. 65535,4,248,4000*
edds.parc.tc.spidlist	Defines the list of TC SPIDs. The values are separated by a colon *e.g. 100*
edds.parc.ev.log.spidlist	Defines the EV Event Log packet SPID. *e.g. 1000*
edds.parc.ev.ool.spidlist	Defines the EV Out of Limits packet SPID. *e.g. 3000*
edds.parc.pus.data.field.header.size	Defines the PUS data field header size. This is after the PUS header size of 6 bytes. *e.g. 12*
edds.parc.spacecraftevents.type	The PUS service type for Spacecraft Events TM packets *e.g. 5*
edds.parc.spacecraftevents.subtypes	The PUS service subtype(s) for Spacecraft Events TM packets *e.g. 1,2,3,4*
edds.dataprovision.pkttc.structure.file	Defines location of the xml file to get custom fields through reflection for Data Provision TC Requests e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_SERVER/config/MIB/DataProvisionCustomFields.xml

Property	Description
edds.ldap.master.urls	A comma separated list of hostnames / IP addresses of the LDAP Master (Producer) instance(s) to be used by EDDS prefixed with “ldap://”. The port number is added to the end after a colon as shown below: *e.g. ldap://localhost:20001,ldap://localhost:20003*
edds.ldap.master.binddn	*The Master LDAP Administration DN* *e.g. cn=Manager,o=edds*
edds.ldap.master.username	The username of the Unix account that will start the LDAP Master server. If this doesn’t match the name of the Unix user that is used to start LDAP, LDAP will not start. *e.g. edds*
edds.ldap.master.admin.password	The admin password of the Master LDAP instance that will be used by EDDS *e.g. secretpassword*
edds.ldap.master.home	The location of where the LDAP master directory will be created. This directory contains scripts related to the population and starting and stopping of the EDDS LDAP servers e.g. */lhome1/edds/EDDS_RUNTIME/LDAP*
edds.ldap.slave.urls	A comma separated list of hostnames / IP addresses of the LDAP Slave (Consumer) instance to be used by EDDS prefixed with “ldap://”. The port number is added to the end after a colon as shown below: *e.g. ldap://localhost:20002,ldap://localhost:20004*
edds.pureftp.ldap.slave.hostname	The hostname / IP address of the LDAP Slave (Consumer) instance to be used by PureFTP configuration (See Section 3.8) *e.g. localhost*
edds.pureftp.ldap.slave.port	The port used to look-up entries in LDAP used by the PureFTP configuration (See Section 3.8) e.g. *1021*
edds.ldap.slave.binddn	*The Slave LDAP Administration DN* *e.g. cn=Manager,o=edds*
edds.ldap.slave.username	The username of the Unix account that will start the LDAP Slave server. If this doesn’t match the name of the Unix user that is used to start LDAP, LDAP will not start. *e.g. edds*

Property	**Description**
edds.ldap.slave.admin.password	The admin password of the Slave LDAP instance that will be used by EDDS *e.g. secretpassword*
edds.ldap.slave.home	The location of where the LDAP slave directory will be created. This directory contains scripts related to the population and starting and stopping of the EDDS LDAP servers. If the LDAP servers are co-deployed, the value for this property is ignored. e.g. */lhome1/edds/EDDS_RUNTIME/LDAP*
edds.ldap.user.basedn	*The base DN of where the users are stored in LDAP* *e.g. o=USERS,o=edds*
edds.ldap.defaultuid	*The default user ID for the FTP host user* *e.g. 1061*
edds.ldap.defaultgid	*The default group ID for the FTP host user* e.g. 100
edds.ldap.central.enabled	Whether to enable authentication with a central LDAP server. If true, the following two fields must be set e.g. *false*
edds.ldap.central.urls	The URL(s) of the central LDAP server. Format is ldap://<hostname>:<port> where <hostname> is the hostname or IP of the LDAP server, and <port> is the port number that the Central LDAP server is running on. Default is 389. Multiple entries can be made separated with commas e.g. *ldap://10.48.29.219:389/*
edds.ldap.central.basedn	The base distinct name where the users can be found. e.g. *o=People,o=esoc,dn=esa,dn=int*
edds.ldap.central.binddn	The Central LDAP user to connect with
edds.ldap.central.password	The password of the user to connect to the central LDAP
edds.ldap.slapd.location	Where slapd can be found. Could be /usr/sbin or /usr/lib/openldap (SLES 15 SP1) e.g. */usr/lib/openldap*

Property	Description
edds.ldap.openldap.schema.location	Where the OpenLDAP schema can be found. Could be /etc/openldap/schema or /etc/ldap/schema e.g. */etc/openldap/schema*
edds.farc.orb.host	The hostname or IP address of the ORB for the FARC instance that will be used by EDDS. e.g. *10.48.29.180*
edds.farc.orb.port	The port of the ORB for the FARC instance that will be used by EDDS. e.g. *28200*
edds.farc.orb.zone	(Only relevant if the Consolidated FARC version 2.0i5 is used) The zone of the FARC Server to use by EDDS. Refer to the FARC Client Services API for more details on the Zone ID [AD-5] e.g. *edds*
edds.farc.archive.instanceid	The Instance ID for FARC. e.g. *PRIME*
edds.farc.archive.zoneid	The Zone ID for the FARC. Optional, can be left empty. Important Note: The FARC Zone ID specified here should match one of the Zone IDs specified in the orb_configuration.xml file in the EDDS Server within config/FARC. However, it is recommended to leave the edds.farc.archive.zoneid property empty unless you are running FARC v2.2.0 or later due to SPR farc#212. The default zone ID will be used instead as long as only one zone has been specified in the orb_configuration file. e.g. <empty>
edds.farc.checkoutdir	The directory where files checked out from the FARC are placed e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_HOME/farc-checkout
edds.farc.activemq.enabled	Whether FARC Subscription support is enabled or not. Requires FARC 2.1.9 or later. e.g. *false*
edds.farc.activemq.broker	The FARC ActiveMQ Message Broker to use. Check the FARC configuration file “server.config” for the correct information. e.g. *failover:(tcp://localhost:61616)?timeout=3000&amp;trackMessages=true*

Property	Description
edds.farc.activemq.broker.username	The FARC ActiveMQ Message Broker username (or blank for no username). Check the FARC configuration file “server.config” for the correct information. e.g. *edds*
edds.farc.activemq.broker.password	The FARC ActiveMQ Message Broker password username (or blank for no password). Check the FARC configuration file “server.config” for the correct information. e.g.*password*
edds.farc.subscription.topic	The name of the topic that the notifications are sent from the FARC. Check the FARC configuration file “server.config” for the correct information. e.g. *esa.egos.farc.FarcSubscriptionInfo*
edds.smtp.enabled	If set to “true”, e-mail messages will be sent when needed. If no mail server exists that EDDS can use, set this to “false” to avoid seeing exception messages in the log files. e.g. *true*
edds.smtps.enabled	If set to “true”, e-mail messages will be sent when needed using SMTPs instead of SMTP. If no SMTPs mail server exists that EDDS can use, set this to “false” to avoid seeing exception messages in the log files. If set to “true” EDDS will ignore edds.smtp.enabled property. e.g. *true*
edds.smtps.auth	*If set to "true", use authentication information to authenticate with SMTPs service. If the SMTPs server requires user/password information set this value to true. If set to “true” EDDS will use edds.smtp.user and edds.smtps.password to authenticate.* *e.g. true*
edds.smtps.password	*SMTPs service password, used if SMTPs requires authentication.* *e.g. 1Logica!*
edds.smtp.host	Address of the SMTP/SMTPs Server that will be used for sending emails *e.g. 10.20.30.40*

Property	**Description**
edds.smtp.port	Port of the SMTP/SMTPs Server e.g. *25 (for SMTP)* *e.g 465 (for SMTPs)*
edds.smtp.user	User for the SMTP/SMTPs Server e.g. *no-reply@esa.int*
edds.smtp.transform	If set to "true", acknowledgement e-mails will be transformed using XSLT before they are sent. The XSLT file should be placed in the directory specified in the property *EDDS_MISSION_XSL_DIRECTORY* on EDDS Server and called “AcknowledgementPart.xsl”. e.g. *true*
edds.smtp.subscription.transform	If set to "true", subscription notification e-mails will be transformed using XSLT before they are sent e.g. *true*
edds.ftp.home	The location of where the EDDS_HOME (FTP Directory) will be created. e.g. */lhome1/edds/*
edds.ftp.account	*The Linux user account that hosts the virtual FTP user accounts. Must have ownership of the edds.ftp.home directory* *e.g. edds*
edds.ftp.accountid	*The Linux user account ID number of FTP_LINUX_ACCOUNT_NAME. This can be discovered by logging on to the account and entering "id"* *e.g. 1061*
edds.ftp.groupid	*The Linux group ID number of FTP_LINUX_ACCOUNT_NAME. This can be discovered by logging on to the account and entering "id"* *e.g. 100*
edds.ftp.useactivemode	*Enables the use of Active or Passive mode for FTP connections. To use active mode, set this property to “true”. To use passive mode, set this property to “false”.* *e.g. false*

Property	Description
edds.ftp.home	*The location of where the EDDS_HOME FTP directory will be located. This should always match where the FTP service is running. The EDDS Server uses this information to set the user's home directory in LDAP.* *e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_HOME*
edds.ftp.credentials.encrypt.keystore	Keystore location for encrypting FTP password for File Delivery. e.g. /lhome1/edds/.keystore For generating a keystore and key for signing Java *keytool* can be used. Use **RSA** algorithm to generate the keypair Also see 3.10.1.
edds.ftp.credentials.encrypt.keystore.type	The type of the keystore. e.g. jks
edds.ftp.credentials.encrypt.storepass	Password for the keystore. e.g. changeit
edds.ftp.credentials.encrypt.alias	Alias of the key for signing. e.g. eddsjws
edds.ftp.credentials.encrypt.keypass	Password for the key. e.g. password
edds.encryption.aes.enabled	*A flag to set whether the AES algorithm is enabled for this mission. Should be "true" if enabled.* *e.g. true*
edds.encryption.aes.keystrength	*Sets the generated AES encryption key strength. Default is 128 bit. This can be increased to 192 or 256 bits, by installing Oracle's Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy from http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html* *e,g, 128*
edds.encryption.rsa.publickeylocation	*RSA public key location. The public key file should be in the X.509 encoded key specification and should just contain the body (the DER part of a PEM file)* *e.g. /lhome1/edds/EDDS_RUNTIME/EDDS_SERVER/config/rsakey.txt*

Property	Description
edds.compression.level	*The default compression level to apply when Zipping files. Can be BEST_COMPRESSION for best compression, BEST_SPEED for fastest compression, DEFAULT_COMPRESSION for the default compression level or NO_COMPRESSION to just store files in the zip without compressing them* *e.g. DEFAULT_COMPRESSION*
edds.logging.level	*The following log levels can be used, in decreasing level of severity. The level sets the lowest level at which log messages will be displayed. For example, a level of "info" will show log messages with the level "fatal", "error", "warn" and "info" but will not show log messages with "debug" and "trace". In normal operation, "info" should be used.* *OFF - Logging is disabled* *FATAL- Severe errors that cause premature termination.* *ERROR - Other runtime errors or unexpected conditions.* *WARN - Other runtime situations that are undesirable or unexpected, but not necessarily "wrong".* *INFO - Interesting runtime events (startup/shutdown).* *DEBUG - Detailed information on the flow through the system.* *TRACE - More detailed information.* *ALL - Same as TRACE* *e.g. info*
edds.activemq.broker	The ActiveMQ message broker to use. This should specify all of the options for the broker, along with the IP address and port of the broker to connect to. *e.g. failover:(tcp://localhost:61616)?timeout=3000&trackMessages=true* The failover option specifies that a re-connection attempt should be made should connection to the broker become lost.
edds.activemq.broker.username	The username to connect to the broker. Can be empty if no username and password is needed. *e.g. edds*
edds.activemq.broker.password	The password to connect to the broker. Can be empty if no password is needed. *e.g. secret*

Property	Description
esa.egos.edds.db.cleanup.checkperiod	The interval to check for old log messages and requests to delete. (Cron Expression. Replace spaces with +) Format is: min, hour, day-of-month, month, day-of-week, year. A Cron Expression of “0 0 2 * * ?” would then become: *e.g. 0+0+2+*+*+?*
esa.egos.edds.db.cleanup.batch.size	The batch size for each request while deleting old logs in database.
esa.egos.edds.db.cleanup.log.oldest	The oldest log messages to keep (in days). A value of 0 disables the feature. *e.g. 30*
esa.egos.edds.db.cleanup.requests.oldest	The oldest completed (i.e. in an error state, delivered with responses deleted or cancelled) requests to keep (in days). A value of 0 disables the feature. *e.g.30*
esa.egos.edds.edds_filesystem.indexed_root	The directory that EDDS Server will index for File System requests. The directory must exist prior to EDDS Server startup. *e.g.* */lhome1/edds/EDDS_RUNTIME/EDDS_HOME/filesyste m-root*
jre.home.dir	The property is needed by the deploy-environment-settings Ant target (Section 3.3.1) and should point to the home directory of the Java 11 JRE installation. *e.g.* */usr/java/jre11*
ant.home.dir	The property is needed by the deploy-environment-settings Ant target (Section 3.3.1) and should point to the home directory of the Ant installation. *e.g. /usr/share/ant*
edds.account.home.dir	The property is needed by the deploy-environment-settings Ant target (Section 3.3.1) and should point to the home directory of the runtime account where EDDS will be used. *e.g. /lhome1/edds*

Property	Description
edds.account.login.script	The property is needed by the deploy-environment-settings Ant target (Section 3.3.1) and should point to the log-in script used when logging in to the runtime account where EDDS will be used. This can either be .bashrc or .cshrc. *e.g. /lhome1/edds/.bashrc*
edds.egscc.rest.dataaccess.url	The EGSSC REST end point base URL for retrieving the EGSCC data. *e.g. http://<IP-EGSCC-HOST>:8181/data-access*
edds.egscc.rest.retrieval.mode	The retrieval mode of delivering data to EDDS by EGSCC end point. Possible values is HTTP *e.g. HTTP*
edds.egscc.rest.dataaccess.kafka	This property is required if the retrieval mode is KAFKA. This describes the IP address and port where KAFKA is running. *e.g. <IP-EGSCC-HOST>:9092*
edds.server.rest.batch.size	The limit corresponds to the number of data received in one batch. If this property is not specified, the default value of 5000 will be used. *e.g. 5000*

Table 3-2: EDDS Deploy Properties

3.2.6.1 Examples

This section gives examples of where it is possible to obtain some of the information required for the properties.

3.2.6.1.1 edds.darc.database.dataspace

This can be found by starting the DARC MMI and viewing the available dataspaces. Please refer to the DARC documentation for how to do this.

3.2.6.1.2 edds.darc.activemq.broker

This is defined in the configuration for the PDG. Please refer to the DARC documentation for how to find this.

3.2.6.1.3 edds.darc.stream.topic

This is defined in the configuration for the PDG. Please refer to the DARC documentation for how to find this.

3.2.6.1.4 edds.parc.corba.ns.port

This is usually the same as the CORBA name service for all SCOS applications. This can be found by logging into the prime server and viewing the file ".s2k.env".

3.2.6.1.5 edds.parc.default.tm.datastream

This can be found by starting the PARC GUI (under Others) and selecting "View" → "Define view" then selecting TM from packet type and clicking the "Data Partitions" button.

3.2.6.1.6 edds.parc.default.tc.datastream

This can be found by starting the PARC GUI (under Others) and selecting "View" → "Define view" then selecting TC from packet type and clicking the "Data Partitions" button.

3.2.6.1.7 edds.parc.default.ev.datastream

This can be found by starting the PARC GUI (under Others) and selecting “View” → “Define view” then selecting EV from packet type and clicking the “Data Partitions” button.

3.2.6.1.8 edds.parc.tc.spidlist

This can be found by editing ~/config_global/MISCcontext.sta and searching for TMSM_SPID on the prime SCOS server.

3.2.6.1.9 edds.parc.ev.log.spidlist

This can be found by editing ~/config_global/MISCcontext.sta and searching for PART_EV_LOG_SPIDS on the prime SCOS server.

3.2.6.1.10 edds.parc.ev.ool.spidlist

This can be found by editing ~/config_global/MISCcontext.sta and searching for OOL_PACKET_APID on the prime SCOS server.

3.2.6.1.11 edds.parc.corba.server.family.name

This can be found by editing ~/config_global/MISCcontext.sta and searching for OPERATIONAL_SERVER_FAMILY on the prime SCOS server.

3.2.6.1.12 edds.parc.corba.app.name

This is the name of the PARC Manager application as registered in the CORBA name server. Up to SCOS 5.4 this value is hardcoded in the PARC Manager application as “PAMASviews” and as such, can’t be changed. The property is present in EDDS in case this should change in the future.

3.2.6.1.13 edds.farc.archive.instanceid

This can be found on the in the FARC GUI under “Archive Administrator”, “Archives” and then examine the “Instance ID” column. Note that it is case sensitive. This should not be confused with “Server Instance”.

3.2.6.1.14 edds.farc.archive.zoneid

This is optional and can be left empty. This is only necessary if you need to connect to different FARC Servers under different CORBA Name Servers. These are configured in the orb_configuration.xml file under the config/FARC directory in the deployed EDDS Server.

3.2.6.1.15 edds.farc.activemq.broker

This can be found on the machine where the FARC Server EDDS will be using runs within ~/FARC/xml/config/server.config under “activemq_broker”. It is recommended to add the “failover” entry to the URI, so that if the connection to the broker is lost, it will automatically be reconnected when it becomes available again. For example: failover:(tcp://mission.esa.int:61616)?timeout=3000&amp;trackMessages=true

3.2.6.1.16 edds.farc.activemq.broker.username

This can be found on the machine where the FARC Server EDDS will be using runs within ~/FARC/xml/config/server.config under “activemq_username”.

3.2.6.1.17 edds.farc.activemq.broker.password

This can be found on the machine where the FARC Server EDDS will be using runs within ~/FARC/xml/config/server.config under “activemq_password”.

3.2.6.1.18 edds.farc.subscription.topic

This can be found on the machine where the FARC Server EDDS will be using runs within ~/FARC/xml/config/server.config under “activemq_topic”.

3.2.7 Apache Ant Note

If you are having problems running Ant, then it might help to create an alias for the Ant command using the following command:

```
alias ant="ant --noconfig"
```

The EDDS Ant target "deploy-environment-settings" (see Section 3.3.1) will add this alias to your log-in script. If you still have problems running Ant (error message about not being able to find the Launcher Jar file) then, as root, execute the following commands:

```
mkdir /usr/share/jdk11
mkdir /usr/lib64/jdk11
```

3.2.8 Development Note

EDDS uses the MyBatis Generator to create the code needed to access the database. This generated code is included in the EDDS source distribution so that access to the database is not required to compile the source code and doesn't need to be re-generated unless the database structure has changed.

If it is required to re-generate the generated code, ensure MariaDB is running and the newest database structure exists. Run the following commands from within the *{User Home Dir}/EDDS/edds/edds-db* directory:

mvn initialize mybatis-generator:generate

3.2.9 Performing the Compilation

Executing the following Maven command will compile all of the EDDS source code in a number of deployable jar files. The ***edds-doc*** module must be compiled from inside the project. The generated jars will be created in the "edds-runtime". From within the *{User Home Dir}/EDDS/edds* directory execute the following commands:

mvn clean install

The compilation should complete without error and the result should be "*BUILD SUCCESSFUL*". If there are errors please check the output and address any points as necessary. Executing *mvn clean* will cause the generated directories to be deleted and then *mvn install* can be re-run.

This phase will only compile the java code. The compiled code can be distributed to be signed later and to generate the runtime. Once the project is built, all the compiled code required to create the runtime will be located under the edds-runtime folder. This folder can be compressed manually to be distributed, or run the following command in the folder *{User Home Dir}/EDDS/edds/edds-runtime-dist*:

mvn package

This will create a file named "edds-runtime-dist-{EDDS-VERSION}-pack-distribution.tar.gz" in the target folder that can be distributed to create the runtime described in the next step.

In order to build both java 8 and java 11 jars for EDDS client libraries. From within the *{User Home Dir}/EDDS/edds* directory execute the following commands

mvn clean install

mvn -pl edds-ws-client,edds-ws-common,edds-common -P java-8 -am clean install

The first command will genereate standard Java 11 libraries and the second command creates the Java 8 libraries of edds-ws-client,edds-ws-common and edds-common.

3.2.10 Creating the Runtime

Before creating the runtime, it is necessary to make sure you have received the necessary key from your security authority so that the Java Web Start libraries can be signed. This can also be manually performed with the following command:

$JAVA_HOME/bin/keytool -genkey -alias eddsjws -keyalg RSA

Where *eddsjws* should match the alias specified in property *edds.javaws.signjar.alias* in the edds-build.properties file described in Section 3.2.5. Refer to Section 3.10.1 which describes a similar process for creating a key for the web server should HTTPS be required.

To generate the runtime we will work in the folder that contains the compiled binaries, that corresponds to the edds-runtime folder or "edds-runtime-dist-{EDDS-VERSION}-pack-distribution.tar.gz" uncompressed, we'll call this folder as {Bin-Dist} folder.

The first step is to correctly configure files *edds-build.properties* and *edds-deploy.properties* in folder {Bin-Dist}.

From within the *{Bin-Dist}* directory, execute the commands:

mvn install

This will copy all of the required files to the location specified by the property *edds.runtime.dir* in *edds-build.propertie*s.

The runtime distribution of the EDDS MMI is stored in *edds.runtime.dir/mmi/edds.mmi.linux.zip*.

The COTS will be signed for the Java Web Start installation. Should the signing process fail with an error similar to:

```
[signjar] jarsigner: unable to sign jar: java.util.zip.ZipException: invalid entry compressed size (expected 8066 but got 8347 bytes)
```

Then you will need to unjar the Jar file that could not be signed into a temporary directory, and delete the manifest files. Next re-jar the file and run the script again.

3.2.11 Ant Project Help

It is possible to obtain a full list of Ant targets and their descriptions with the Ant command:

ant –projecthelp

or

ant –p

for short hand.

For example, several additional Ant targets have been added for convenience:

- *deploy-all-edds-server-components* – performs the targets “deploy-edds-server”, “deploy-archiver”, “deploy-performance-processor” in one step
- *deploy-all-web-server-components* – performs the targets “deploy-tomcat”, “deploy-javaws”, “deploy-delivery-manager”, “deploy-request-submitter” in one step

3.2.12 Creating the JavaDoc

The JavaDoc can be generated from the source code using the following command:

mvn clean install javadoc:aggregate-jar -Dadditionalparam=-Xdoclint:none *-DskipTest=true*

This will place the generated JavaDoc into the target/apidocs directory and create a target/edds-parent-3.6.0-javadoc.jar file.

3.2.13 Publich documentation in Nexus

The ***edds-doc*** module is responsible for creating the zip containing the documentation for edds. This module is a maven project and can therefore be published on the nexus. To generate the zip manually, simply run the **mvn install** command in the edds-doc folder. As a result the zip file with the following name will be generated: **edds-doc-<version>-dist.zip**

3.2.14 Building and running JUnit Tests

Unit tests are compiled and run as part of the normal build process. The tests are compiled and placed into a separate Jar file to that of the runtime Jars. To bypass running the tests set the *skipTests* property to true while building. E.g.

```
mvn install –DskipTests=true
```

3.2.15 Measuring code test coverage

To measure code test coverage open source tool Cobertura is used. To run unit test and measure code coverage run following command from the {User Home Dir}/EDDS/edds directory:

```
mvn install cobertura:cobertura
```

Generated reports can be found in each projects folder under *target/site/cobertura*. The report format can be selected using the cobertura.report.format property. Supported formats are xml and html, default is html.

3.2.16 Updating the EDDS version

To update the version of EDDS, it is not necessary to update the poms' file manually. Instead, the automatic tool EUD Delivery Helper must be used. This tool is present in the folder delivery-tool.

When EDDS is in a SNAPSHOT version and you want to upgrade the version to a Release version then edit the file *EDDS_Delivery_Recipe.tx*t and update VERSION, PRE_VERSION and RELEASEYEAR property. Where version is the next desired version, PRE_VERSION is the snapshot version number and RELEASEYESR is the year of the release of the new version. Then run the *delivery.sh* script. This script will update all pom files, html files and loading images, changing them from the snapshot version to the Release version indicated in the VERSION property.

To update the snapshot version is a similar procedure but in this case, the file to edit is *EDDS_Snapshot_Recipe.txt*. In this file, you will need to update the VERSION property to the SNAPSHOT version that you want. Then start the *snapshot.sh* script.

Note that both scripts work on a snapshot version, so you can not make the scripts work on a release version.

So if you want to create a release version and then a new snapshot version it is advisable to create two different branches, one to create the release version and in the other the next snapshot version and then import the snapshot version with the favourite merging mechanisms over the release version.

3.3 Deployment

This section outlines taking the compiled EDDS code and deploying it in to a runtime environment. This includes both development and production scenarios.

When upgrading from the last release of EDDS, in addition to the following instructions, some additional steps need to be taken as specified in [AD-7]. Since the EDDS deployment might include several machines, these steps might need to be carried on different machines, depending on the selected deployment architecture.

3.3.1 EDDS Environment Variables

Some environment variables, such as the Java home location, need to be set before any EDDS components can be launched. To make this step easier, there is an Ant target to create a file containing the needed environment variables. These are taken from the edds-deploy.properties file defined in Section 3.2.6. To run this target, enter:

```
cd {EDDS Runtime Dist Dir}
ant deploy-environment-settings
```

You will need to log out and log back in for the changes to take effect. If you do not keep the *{EDDS Runtime Dist Dir}* directory when running EDDS, you'll need to either edit the edds-deploy.properties file to point to the correct Java home location by editing the property *jdk.home.dir.*

The EDDS start-up scripts use the JRE_HOME environment variable for executing Java. If this environment variable isn't set, then the version of Java found on the path is used. Confirm the version of Java you are using in your runtime environment with the command `java -version` – it should be version 11.0.2+9 or later.

3.3.2 EDDS Database Installation

EDDS uses a MariaDB database instance and the schema must be deployed before the application can be used. The database must be created with a user that has administrative privileges. The database is used in the development environment for unit testing and is also required in production.

Before executing this step, ensure that the database related properties (*edds.db.admin.user* and *edds.db.admin.password*) in the deploy properties file have been updated as per Section 3.2.6.
Change current location to the base directory of the EDDS build area. The database script will create the EDDS database and tables and create the EDDS database user. This means that the admin user specified in the properties will need permission to create databases and users ("with grant option").

cd {EDDS Runtime Dist Dir}

Note: From this directory all remaining build and deployment commands can be executed.

Execute the following command that will execute the EDDS SQL scripts against the database

ant build-db

This activity should complete without failure or error.

3.3.3 Configure LDAP

EDDS requires two LDAP instances running in order to operate.

- LDAP Slave – Must be available for the EDDS Web Server and Delivery Manager
- LDAP Master – Must be available for the EDDS Server and EDDS Archiver

It is possible for the two EDDS LDAP instances to be co-deployed, that is, they can operate on the same computer.

A number of scripts have been provided to allow the stopping and starting of EDDS. After deployment, these scripts will be located in **{EDDS Deployed Runtime Dir}*LDAP/scripts***.

In order to deploy the scripts in an environment where both the LDAP Master and Slave are deployed on the same machine execute the following command:

ant deploy-ldap-co

In order to deploy the scripts in an environment where the LDAP Master and Slave are deployed on different machines, copy the *{EDDS Runtime Dist Dir}* to the LDAP Master server (creating the relevant Unix users if necessary) and execute the following command:

ant deploy-ldap-master

Next, copy the *{EDDS Runtime Dist Dir}* to the LDAP Slave server (creating the relevant Unix users if necessary) and execute the following command:

ant deploy-ldap-slave

Note that the relevant COTS (i.e. OpenLDAP) will need to be installed on each server that is to run LDAP. The initial database containing the edds root element and the MISSIONS and USERS tree will be created.

When deploying the LDAP Master, the LDAP server is started in order to populate it with the basic root entries required for EDDS.

It is now possible to start LDAP with the command:

./start
This step should complete without errors.

3.3.4 Configure ActiveMQ

This only needs to be performed once. Details can be found in Appendix D.

3.3.5 Deploy EDDS Archiver

The EDDS Archiver is responsible for archiving log messages to the database, and processing requests for information from the Delivery Manager and Web Server from the database as well as initially processing requests (creating a new request for each scheduled request and adding into the database). The following task will deploy the EDDS Archiver to the location specified by the *edds.archiver.home* property in the *edds-deploy.properties* file.

ant deploy-archiver

3.3.5.1 External connections

The EDDS Archiver needs access to

- The ActiveMQ Broker configured with *edds.activemq.broker*
- MariaDB server configured with *edds.db.url*
- Master LDAP server configured with *edds.ldap.server.hostname* and *edds.ldap.master.port*

3.3.6 Deploy EDDS Delivery Manager

The following task will deploy the EDDS Delivery Server to the location specified by the *edds.delivery.manager.home* property in the *edds-deploy.properties* file.

ant deploy-delivery-manager

Before attempting to start the Delivery Manager, ensure that the polling inbox directory exists and is readable by the Delivery Manager. Only one Delivery Manager instance should be configured for each web server regardless of the number of EDDS Servers installed. The outbox of each EDDS Server should be the same as the inbox for the Delivery Manager.

3.3.6.1 External connections

The EDDS Delivery Manager needs access to

- The ActiveMQ Broker configured with *edds.activemq.broker*
- SMTP server configured with *edds.smtp.host* and *edds.smtp.port* (only if *edds.smtp.enabled* is set to “true”)
- SMTPs server configured with *edds.smtp.host, edds.smtp.port, edds.smtps.auth and edds.smtps.password (only if edds.smtps.enabled is set to “true”)*

3.3.7 Deploy EDDS Web Service

Before attempting to deploy the EDDS Web Server (EDDS WS) please ensure that your Tomcat installation meets the requirements outlined in the Prerequisites section and that the *edds-deploy.properties* file has been configured as per Section 3.2.6. In addition, please ensure that the Master and Slave LDAP instances have been configured as per Section 3.3.3 before deploying the EDDS Web service.

Before running the Web Service deploy command:

- Ensure that the Tomcat instance is running (it can be started using the *startup.sh* and *shutdown.sh* scripts in the *bin* directory under the Tomcat installation directory).
- Under the *logs* directory check the *catalina.out* file to check that no errors occurred whilst starting the server, for example: a port conflict. If there is a port conflict, update the *server.xml* file under the *conf* directory of the Tomcat installation directory to a free port. The *edds-deploy.properties* file must then be updated to correctly reflect this.

- Check the URL of the Tomcat welcome page by visiting the IP address and port number that have been configured for the server (in *server.xml* under the Tomcat directory).

  For example: *http://10.48.29.159:8080*. If a "Congratulations" page is displayed, the server has successfully started.
- It is also recommended to click the Tomcat Manager link on the left hand side of this page and ensure that the credentials specified in *tomcat-users.xml* can successfully view the management page.

From within the {EDDS Runtime Dist Dir} directory, execute the command
ant deploy-tomcat

Note: if the EDDS Web Server had previously been deployed with the ant deploy-tomcat command, running it again will overwrite any existing documents. The file 'status.txt' contains all previous statuses reported and this will be overwritten during the deployment. Copy and backup the file if previous status information is required.

After deploying the EDDS Web Server, it may be necessary to update the file that specifies which client IP addresses are allowed to perform user management operations, even if the client logs in with a user authorised to perform user management requests.

To specify the IP addresses and subnets of the clients that can submit User Management requests, update the file *{Apache Tomcat Home}/webapps/edds/WEB-INF/allowedAddresses.txt* Any IPv4 and IPv6 addresses or subnets are accepted. The file should contain one address or subnet per line. e.g.

```
127.0.0.1/8
::1
10.48.0.0/16
```

The first example shows a 24-bit IPv4 block, e.g. 127.0.0.0 – 127.255.255.255, the second example shows an IPv6 localhost address and the third example shows a 16-bit IPv4 block e.g. 10.48.0.0 – 10.48.255.255. Further information about IP address ranges can be found here: http://en.wikipedia.org/wiki/IP_address

Once you've updated the file, restart Apache Tomcat as described above.

After the deployment of the EDDS WS has completed, please check the Tomcat log which can be found in the log directory of the Tomcat installation.

3.3.7.1 EDDS Web Pages

The EDDS webpages will be deployed to the '/edds' directory (within the Tomcat hot deploy directory of "webapps"), accessible at URL/edds.

For example: *http://10.48.29.159:8080/edds*
The index page will display the help, user manuals, installation and configuration guides. The documents are stored within a subfolder at 'edds/docs'.

The 'edds_status.jsp' page, will display the current status of the EDDS server. The status can be updated by editing the 'status.txt' file.

To replace any documentation with the latest version, upload and replace the file within the 'docs' folder. Ensure that the name of the latest version is exactly the same as the previous file so the links will not need to be updated.

File name	File description
EGOS-GEN-EDDS-SUM.pdf	***Software User Manual*** This document provides the EDDS Software User Manual.
EGOS-GEN-EDDS-GLO.pdf	***EDDS Glossary*** This document is the Glossary of the EGOS Data Dissemination System (EDDS), containing the definition of the applicable and reference

	documents, as well as the acronyms and terms applicable to the system.
EGOS-GEN-ICD.pdf	***External User Interface Control Document (EUICD)*** This document defines the service interfaces available to clients of the EGOS Data Dissemination System (EDDS).
EGOS-GEN-EDDS-CIG.pdf	***Configuration and Installation Guide*** This document contains guidelines on the use of the Configuration and Installation of the EGOS Data Dissemination System and its pre-required subsystems and libraries.
EGOS-GEN-EDDS-SRN.pdf	***Software Release Notes*** This document contains the release notes for the installed EDDS release.
EGOS-GEN-EDDS-SDD.pdf	***Software Design Document*** This document provides a top-level architectural design of the EDDS. This document does not include a Detailed Design that is covered by the JavaDoc. The readership is expected to be software developers and ESA technical authorities with interest in the EDDS component
EGOS-GEN-EDDS-EUG.pdf	***EDDS Upgrade Guide*** This document provides information on how to upgrade from an existing installation running the last issued release of EDDS.

Table 3-3 EDDS Web Pages

3.3.7.2 External connections

The tomcat instance running EDDS Web Service needs access to

- The ActiveMQ Broker configured with *edds.activemq.broker*
- LDAP server configured with *edds.ldap.slave.hostname* and *edds.ldap.slave.port*

3.3.8 Deploy MMI Java Web Start Libraries

This will deploy the EDDS MMI Java Web Start libraries to Tomcat. The EUD and EDDS MMI jar files will be signed with the key from the keystore provided by edds.javaws.signjar.* properties and the Tomcat web server should be available to the clients at the address specified at edds.javaws.server. The Java Web Start libraries will be available at URL {edds.javaws.server}{ edds.javaws.context}.
Java *keytool* utility that comes with Java installation can be used for generating the keystore and the key for signing the libraries. See also 3.10.1 for information on generating keys.

ant deploy-javaws

This command will overwrite an existing deployment of Java Web Start client package if it already exists.

EDDS Clients that are using Java Web Start to run start their MMI aren't affected by the redeployment. Since the libraries are deployed as a separate web application, redeploying these does not affect the current sessions with EDDS Web Server.

The clients' EDDS MMI will be updated the next time they start the MMI through Java Web Start.

Note that as the JNLP files must be included in an EDDS Jar file that has been signed to comply with the security restrictions in Java 1.7.0_51 and later, it is not possible to change the web server hostname/IP address from the runtime directory. Instead, the edds-build.properties file must be updated and the runtime generated again.

If using a self-signed certificate to sign the Jar files, it will be necessary for the end user to add the EDDS Web Server to the list of exceptions in the Java Control Panel, otherwise they will not be able to launch the application. It is **strongly** recommended to use a certificate from a recognised certificate authority and to use HTTPS for the web server for security reasons.

3.3.9 Deploy EDDS MMI Web Application

This will deploy the EDDS MMI Web Application libraries to Tomcat.

From within the {EDDS Runtime Dist Dir} directory, execute the command

ant deploy-webapp

The EDDS web application will be deployed as the 'eddsweb' context (name within the Tomcat hot deploy directory of "webapps"), The EDDS MMI will be accessible at URL /eddsweb/eddsMmi.

For example: *http://10.48.29.159:8080/eddsweb/eddsMmi*.

After deploying, it is necessary to configure some context parameters for the Tomcat server. In the Tomcat installation directory, a context file is automatically created (see manual for more information): `$CATALINA_BASE/conf/[enginename]/[hostname]/eddsweb.xml`

e.g. `$CATALINA_BASE/conf/Catalina/localhost/eddsweb.xml`

When the context file is changed, the application is automatically reloaded and the new context parameters applied. This file contains two parameter entries for `EDDS_HOME` and `edds.server.endpoint.url`. The values of these parameters need to be changed.

One of the main differences between the standalone and web application is handling the `EDDS_HOME` directory in the client application. On the MMI Web Application this property is mandatory and it should be pointing to a directory on the filesystem of the Tomcat machine. All the EDDS users have unrestricted access to this shared folder, subfolders and files. As a result, it is strongly recommended to point this property to a new, empty directory specifically for this purpose, and not the EDDS home folder.

The default EDDS Web Server endpoint address is configured with `edds.server.endpoint.url`. This will be used as the default endpoint address for new client sessions. Users can override this value from their application preferences.

For the proper working of the EUD RAP client it is necessary to ensure that the file {$CATALINA_HOME}/bin/setenv.sh contains:

#!/bin/sh

export JAVA_OPTS= "-Dosgi.nl.user=en_GB -Dorg.apache.activemq.SERIALIZABLE_PACKAGES="*

3.3.9.1 External connections

The Tomcat instance running EDDS MMI Web Application needs access to

The EDDS Web Server. E.g. when EDDS Web Server and the MMI Web Application are deployed to the same Tomcat instance, only local access is needed otherwise network access to the machine and port of the machine that runs the EDDS Web Server is needed.

3.3.10 Deploy EDDS Server

The following task will deploy the EDDS Server to the location specified by the *edds.server.home* property in the *edds-deploy.properties* file. When deploying several EDDS Servers on the same machine in the same account, the location must be different.

From within the *{EDDS Runtime Dist Dir}* directory, execute the following command:

ant deploy-edds-server

Note that if you deploy and run EDDS Server on the same machine but in different user accounts, it will be necessary to modify the file config/FARC/model.config in the EDDS Server directory and change the temporary locations for the properties "working_dir" and "log_dir" to something unique. When the FARC Client API creates these directories, it is only writeable by the current user account.

Before starting the EDDS Server, it is necessary to configure it for the SCOS installation. Information on this can be found in Appendix A and Appendix B.

3.3.10.1 External connections

The EDDS Server needs access to:

- The ActiveMQ Broker configured with *edds.activemq.broker*
- Master LDAP server configured with *edds.ldap.server.hostname* and *edds.ldap.master.port*
- MariaDB server configured with *edds.db.url*
- FARC server configured with *edds.farc.orb.host* and *edds.farc.orb.port*, if it is configured to handle any FARC requests
- PARC server configured with *edds.s2k.corba.ns.hostname* and *edds.s2k.corba.ns.port*, if it is configured to handle any PARC requests.
- DARC database configured with *edds.darc.database.host*, if it is configured to handle any DARC requests.
- Data Provision Services configured with *edds.provision.packet, edds.provision.command, edds.provision.event, edds.provision.ool*, if it is configured to handle any Data Provision requests

3.3.11 Deploy Request Submitter (optional)

The following task will deploy the Request Submitter to the location specified by the *edds.request.submitter.home* property in the *edds-deploy.properties* file. When deploying several instances of Request Submitter on the same machine in the same account, the location must be different.

From within the *{EDDS Runtime Dist Dir}* directory, execute the following command:

ant deploy-request-submitter

3.3.12 Deploy Stream Client (optional)

The following task will deploy the Stream Client to the location specified by the *edds.stream.client.home* property in the *edds-deploy.properties* file. When deploying several instances of the Stream Client on the same machine in the same account, the location must be different. Note that Stream Client uses some of Request Submitter's properties (*edds.request.submitter.username, edds.request.submitter.password, edds.request.submitter.server.address*). So if you are going to use Stream Client, these three properties must be filled.

From within the *{EDDS Runtime Dist Dir}* directory, execute the following command:

ant deploy-stream-client

3.3.13 Deploy Performance Processor (optional)

The following task will deploy the Performance Procesor to the location specified by the *edds.performance.processor.home* property in the *edds-deploy.properties* file. This step is optional, and is only needed if information about the time taken to complete a batch request is needed. The Performance Processor must be running while the EDDS Server is processing the batch requests to be reported, as it listens to the log messages that the EDDS Server creates as they are created. The log messages are only created by the EDDS Server if the property "enable.performance.test" is set to "true" in the EDDS Server properties file.

From within the *{EDDS Runtime Dist Dir}* directory, execute the following command:

```
ant deploy-performance-processor
```

3.3.14 Deploy EDDS Migrator (optional)

The EDDS Migrator is responsible for migrating data from the last version of EDDS to the current version. The following task will deploy the EDDS Migrator to the location specified by the *edds.migrator.home* property in the *edds-deploy.properties* file.

```
ant deploy-migrator
```

When the migration is complete, the EDDS Migrator runtime can be deleted. Refer to the EDDS Upgrade Guide ([AD-7]) for more information.

3.3.14.1 External connections

The EDDS Migrator needs access to

- LDAP master server configured with *LDAP_CONNECTION_URLS*
- Admin access to MariaDB server EDDS schema configured with *edds.db.url*

3.4 Firewall Configuration

Before any of the EDDS Components can be started, it is necessary to configure the firewall on each machine to allow the following ports to be open:

Component	Port	Direction	Description
All	61616	Outbound	ActiveMQ Broker. Port is configurable in the ActiveMQ configuration file.
LDAP Master	20002	Outbound	For replication to the LDAP Slave. Port is configurable when deploying LDAP and can be specified in the EDDS properties.
SFT Sender	3887	Outbound	For connecting to the SFT Receiver. Port is configurable in the SFT properties.
EDDS Archiver, EDDS Server	3306	Outbound	For connecting to MariaDB. Port is configurable in the MariaDB configuration and can be specified in the EDDS properties.
EDDS Archiver, EDDS Server	20001	Outbound	For connecting to the LDAP Master server. Port is configurable when deploying LDAP and can be specified in the EDDS properties.
FTP Server	22	Inbound	To allow FTP connections
Web Server	8080	Inbound	For HTTP connections to the web server. Port is configurable in the Tomcat properties and can be specified in the EDDS properties
Web Server	8443	Inbound	For HTTPS connections for the web server. Port is configurable in the Tomcat properties and can be specified in the EDDS properties
Web Server	8005	For information	This is the port that by default Tomcat listens on for shutdown requests. No changes on the firewall are needed, this may need to be changed in Tomcat configuration if multiple copies of EDDS are running on the same hardware to avoid port conflicts.
Web Server	20002	Outbound	For connecting to the LDAP Slave server. Port is configurable when deploying LDAP and can be specified in the EDDS properties.
Web Server	389	Outbound	Only needed for when a central LDAP server is in use and it is not located in the same network as the EDDS Web Server.
Delivery Manager	20002	Outbound	For connecting to the LDAP Slave server. Port is configurable when deploying LDAP and can be specified in the EDDS properties.
EDDS Web Client	8080 / 8443	Inbound	Depends on Tomcat configuration. Needed for users to access the EDDS Web Client application in their browser.

Table 3-3 Firewall Configuration

In a typical set-up, where the EDDS Archiver, EDDS Server(s), LDAP Master, SFT Sender and MariaDB database are deployed on the OPS LAN and the Web Server, Delivery Manager, LDAP Slave, FTP Server and SFT Receiver are deployed on the RELAY LAN and all the ports are left at the default values, the following set-up can be made:

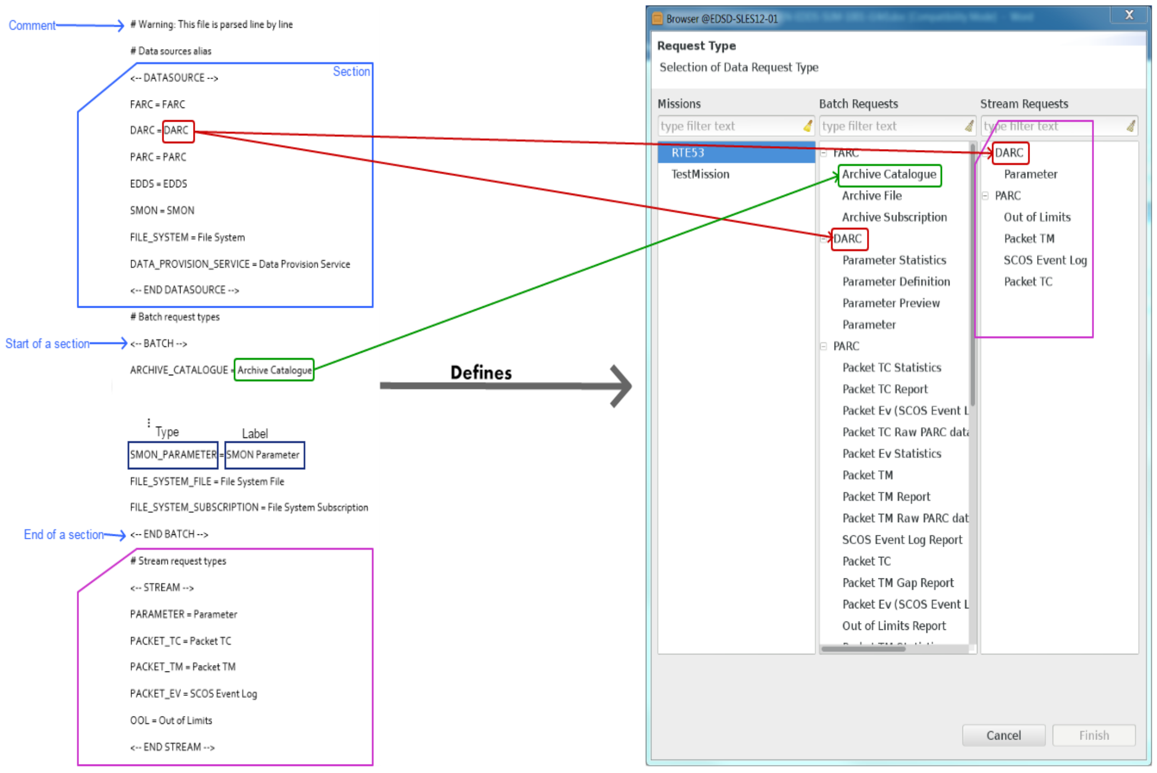
OPS-LAN: allow outbound ports: 61616 (ActiveMQ), 20002 (for LDAP replication), 3887 (SFT)
RELAY-LAN: it is assumed all incoming ports are open, so no configuration is needed, otherwise refer to the above table.

It may also be necessary to allow the EDDS Server machine to connect to the LDAP Master machine by editing the file /etc/hosts.allow as root. The following line should be added:

slapd : <IP of the machine connecting to ldap> : allow

3.5 Request Attributes label - “Alias” configuration

Edds gives you the ability to define the label of requests and Datasources to make it more user friendly and better adjust to the missions terminology. To configure such aliases, you need to edit the file “Aliases.conf” in folder {EDDS runtime dist dir}/EDDS_SERVER/config.



```
# Warning: This file is parsed line by line
# Data sources alias
<-- DATASOURCE -->
FARC = FARC
DARC = DARC
PARC = PARC
EDDS = EDDS
SMON = SMON
FILE_SYSTEM = File System
DATA_PROVISION_SERVICE = Data Provision Service
<-- END DATASOURCE -->
# Batch request types
<-- BATCH -->
ARCHIVE_CATALOGUE = Archive Catalogue

:
SMON_PARAMETER = SMON Parameter
FILE_SYSTEM_FILE = File System File
FILE_SYSTEM_SUBSCRIPTION = File System Subscription
<-- END BATCH -->
# Stream request types
<-- STREAM -->
PARAMETER = Parameter
PACKET_TC = Packet TC
PACKET_TM = Packet TM
PACKET_EV = SCOS Event Log
OOL = Out of Limits
<-- END STREAM -->
```

Browser @EDSD-SLES12-01

Request Type

Selection of Data Request Type

Missions	Batch Requests	Stream Requests
RTE53	FARC	DARC
TestMission	Archive Catalogue	Parameter
	Archive File	PARC
	Archive Subscription	Out of Limits
	DARC	Packet TM
	Parameter Statistics	SCOS Event Log
	Parameter Definition	Packet TC
	Parameter Preview	
	Parameter	
	PARC	
	Packet TC Statistics	
	Packet TC Report	
	Packet Ev (SCOS Event L	
	Packet TC Raw PARC data	
	Packet Ev Statistics	
	Packet TM	
	Packet TM Report	
	Packet TM Raw PARC dat	
	SCOS Event Log Report	
	Packet TC	
	Packet TM Gap Report	
	Packet Ev (SCOS Event L	
	Out of Limits Report	

Cancel Finish

Figure 1 Request name mapping configuration

In the figure on the left, we can see an example of the Aliases.conf file. The file is subdivided into three sections: Datasource, Batch, and Stream. Tags that open and close sections should not be deleted or edited as they are essential for proper parsing of the file. You can add comments on a blank line and comments should start with ‘#’ char. Each definition has the following form <internal type>=<label>, just change the label to customize the view. When a user logs in within EDDS MMI, the alias definition table for each mission are asked to EDDS Server. This way, for each mission, the administrator can overwrite what the user will see in the EDDS MMI. The administrator have to define “aliases” before the server starts up, as this file is loaded only when booting. If you change that file while the server is running, you must restart the EDDS Server to make the changes effective. Note that this file is only for defining the labels, to act on what requests the user may see you have to modify the permissions of the user.

3.6 Running the EDDS Components

3.6.1 Start EDDS Archiver

Browse to the location of the EDDS Archiver installation as specified by the *edds.archiver.home* property and execute the command

```
        ./EddsArchiverService.sh start
Started.
```

When the EDDS Archiver starts for the first time, it checks LDAP to see if there are any users present. If not, it creates a new EDDS Admin user with the username of “admin” and a default password of “password”. The password must be changed after logging in for the first time.

The EDDS Archiver can be stopped by running:

```
        ./EddsArchiverService.sh stop
Kill signal sent.
```

If the EDDS Archiver is configured to log messages, this data will be put into the file logs/edds-archiver.log (configurable within the file log4j.xml within the EDDS Archiver runtime directory). It is important that the EDDS Archiver is started first so that all log messages from the other components are saved into the database for later retrieval.

The EDDS Archiver also has the important role of receiving batch requests from the web server, saving them into the database and then sending a message with the details of the scheduled request(s) for the relevant EDDS Server to process (the original request contains the schedule information, and a new request needs to be created for each scheduled request). The Archiver also responds to other requests that are not mission specific, for example the status of requests, the acknowledgement information of a request, requests for historical log messages, the number of ongoing requests (for quota management) and for file and request management from the Delivery Manager.

3.6.2 Start Delivery Manager

Browse to the location of the Delivery Manager installation as specified by the *edds.delivery.manager.home* property and execute the command

```
        ./DeliveryManagerService.sh start
Started.
```

The delivery manager can be stopped by running:

```
        ./DeliveryManagerService.sh stop
Kill signal sent.
```

If the Delivery Manager is configured to log messages, this data will be put into the file logs/delivery-manager.log (configurable within the file log4j.xml within the Delivery Manager runtime directory).

3.6.3 Start the EDDS Server

Before starting the EDDS Server, ensure you have edited the edds.properties file and added the following line:

```
MISSION_NAME=mission
```

Replace “mission” with the name of the mission that you want this EDDS Server installation to process requests for. EDDS Server will verify that this mission exists in LDAP when starting up, and log a warning if the mission doesn't exist. The EDDS Server will continue to start up however, waiting for requests for the mission and request sub types specified.

Browse to the location of the EDDS Server installation as specified by the *edds.server.home* property and execute the command

```
        ./EddsServerService.sh start
Started.
```

The EDDS Server can be stopped by running:

```
        ./EddsServerService.sh stop
Kill signal sent.
```

If the EDDS server is configured to log messages, this data will be put into the file logs/edds-server.log (configurable within the file log4j.xml within the EDDS Server runtime directory).

For testing purposes, a DARC parameter stream simulator service is available in the EDDS Server deployment. This service generates a random parameter with a name selected from one of the parameter names specified in the property *edds.darc.stream.simulator.paramnames* in the EDDS Server properties file. By default, a parameter is put onto the DARC real-time parameter topic every 100 milliseconds. This can be changed by editing the *paramStreamSimContext.xml* file and altering the value for "period" in the Camel route for the timer uri.

The simulator can be started with the command:

```
        ./ParameterSimulatorService.sh start
```

and stopped the command:

```
        ./ParameterSimulatorService.sh stop
```

To include the type of the parameter value in the message header, edit the file "paramStreamSimContext.xml" which can be found in the EDDS Server directory (the same place where the ParameterSimulatorService.sh file is) and find the following entry:

```
<bean id="paramGenerator" class="esa.egos.edds.server.stream.simulator.ParamGenerator">
   <property name="includeValueType" value="false" />
</bean>
```

Set the value "false" to "true". The simulator will then set an extra property on the message header with a key name of "ValueType" and a value of either BOOLEAN, INTEGER, FLOAT, DOUBLE or STRING as appropriate. This can then be filtered on when performing a Parameter Stream request, in that a filter value of "double value greater than 0.2" will only match double values greater than 0.2 and not float and integer values too. Ensure you set the property "esa.egos.edds.darc.stream.enablefilterbytype" to "true" within "edds.properties" in the EDDS Server directory for the mission supporting DARC streaming.

3.6.3.1 Configuring EDDS Server memory usage

Since EDDS Server is the main processing component in EDDS, it typically uses more memory and computing resources than other components. It is possible to configure the limits for the memory usage and the strategies for runtime garbage collection using the standard Java Virtual Machine (JVM) configuration options.

The JVM is started from EDDS Server launcher script in EDDS Server installation directory. You can add the arguments right after the program execution as java arguments as with any other java application.

For a list of possible arguments for the JVM look at the corresponding documentation by the provider: https://docs.oracle.com/javase/8/docs/technotes/tools/unix/java.html andhttps://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/.

For example, to limit the memory usage of EDDS Server to 512MB you would add -Xmx512m and to make the garbage collector wake up earlier than it would by default (the default value is 68%) add -XX:+UseCMSCompactAtFullCollection=50

Excerpt from eddsServerLauncher.sh:

```
# Execute EDDS Server
exec java -Xmx512m -XX:+UseCMSCompactAtFullCollection=50 -cp ${CLASSPH1} -
Dlog4j.configuration=file:///${DIR}/log4j.xml
esa.egos.edds.server.EddsServerStarter file:///${DIR}/eddsServerContext.xml
```

3.6.3.2 Configuring the user limit of inotify watches

EDDS Server uses the OS's inotify watches to monitor any changes in the folder configured as the File System Root. The watches are not recursive, so a watcher needs to be registered for every subfolder in the root. The number of watchers that can be registered is limited by the OS (default setting on the baseline is 65536) and if EDDS Server reaches the limit the following exception will be thrown:

```
java.io.IOException: User limit of inotify watches reached
```

If the number of folders cannot be reduced it is possible to raise the limit. Generally however, this is discouraged, because it comes with a performance penalty. That being said, in some cases raising the limit still makes sense. Here's how this can be achieved:

To read the current value:

```
$ sudo sysctl fs.inotify.max_user_watches
```

To temporarily (does not persist through machine restart) change the value:

```
$ sudo sysctl fs.inotify.max_user_watches=<new_value>
```

To permanently change the value `/etc/sysctl.conf` should be edited to include the line:

```
fs.inotify.max_user_watches=<new_value>
```

The value in `/etc/sysctl.conf` will take effect after the next reboot. However, there is a way to load the new config without restarting the machine the following command can be run:

```
$ sudo sysctl -p
```

3.6.4 EDDS Web Service

The EDDS Web Service and Web Client should be started and stopped using the Tomcat scripts that are provided as part of the standard Tomcat installation.

The two scripts, *startup.sh* and *shutdown.sh* scripts reside in the 'bin' directory of the tomcat installation.

3.6.5 Start the Request Submitter

The RS requires a valid user account to submit requests to the EDDS Server. The user account can be created from within the MMI in the User Management section. It is important that the correct Data Access Set, Role and Mission(s) are assigned to this user.

Request submissions through the RS adhere to the same restrictions as the MMI. The account must have appropriate permissions to process the given requests. Requests that are beyond the permissions of the Account will not be completed.

For security reasons it is recommended that the RS user account is not an Admin and can access only the required information and request types.

For submitting GDDS requests, the Request Submitter user needs access to submit TM packet, TC packet, TC Packet Report and Archive File requests.

Browse to the location of the Request Submitter installation as specified by the `edds.request.submitter.home` property

Check the config.properties file, ensuring that the username and password for the RS user matches the defined values.

execute the command

```
        ./RequestSubmitterService.sh start
Started.
```

The Request Submitter can be stopped by running:

Browse to the location of the Request Submitter installation as specified by the `edds.request.submitter.home` property and execute the command

```
        ./RequestSubmitterService.sh stop
Kill signal sent
```

Batch requests can be created from the EDDS MMI and saved to create XML templates that can be submitted to the Request Submitter. Although the batch requests contain the username to submit the request as, the Request Submitter always overrides this to use its defined username and role as specified in the configuration file.

Cancel, suspend and resume requests can also be submitted through the request submitter. See Appendix J for examples of the XML requests. When a request is successfully submitted, the request file is placed in the Request Submitter completed directory (see the configuration file) along with a log file with the same name as the request but with ".log" appended. The log file consists of a comma separated list of the request IDs provided by EDDS enclosed in square brackets.

3.6.5.1 Request Submitter LDAP response read timeout

If the requests from the Request Submitter are failing because of LDAP response read timeout, then the following 3 possible solutions are recommended. If the first solution doesn't work, then the next one should be tried and so forth.

- Add *com.sun.jndi.ldap.read.timeout=<READ_TIMEOUT>* to **/tomcat/conf/catalina.properties**, where *<READ_TIMEOUT>* is the timeout value in milliseconds. This value can be set to 3600000 (1 hour).
- Disable Follow Referral option in LDAP, add *java.naming.referral="ignore"* to **/tomcat/conf/catalina.properties**
- Update JRE to Java 11

3.6.6 Start the Stream Client (optional)

The SC is a standalone client that can be started as a service and its job is to save streaming data to files periodically. It currently supports only the Packet TM Stream type and the save can be done in xml, GDDS binary or proto (the message is PacketTMBinary) format. The SC requires a valid user account to receive streaming data from EDDS Server. SC needs an existing stream request ID because SC don't make any new request, it just listen to an existing request.

So before we can start the service, we have to edit the file *{edds.stream.client.home}/config.properties.* Some properties are already set as "username", "password", "eddsServerAddress" and "outbox", but can be modified as needed.

Let's see the new properties to configure:

- **output.format**: The value must be one between "xml", "proto", "gdds", without quotes. Indicates the format of the final files that will be saved in the folder *{outbox}.*
- **save.interval.sec**: indicates the duration (in seconds) for which the client will collect data until it must create a new file. Each file in the name includes the creation date/time so that it can be easily sorted by any filesystem. It is advisable to keep a low range to frequently save data (for example, 60-120 seconds).
- **requestId**: ID of an active Packet TM Stream request. You can right click on an active request on EDDS MMI and click on "Get Full Status Report as text" and then copy-paste RequestId field.
- **param.darcParamBinary**: This configuration is only applicable for Darc Parameter Stream request. This should be configured as 'true' if the output of Darc Parameter Stream request needs to be saved as DarcParameterBinary. By default, this value is 'false' and the output will be saved as DarcParamRec. The DarcParameterBinary is supported for output.format "xml", "proto" and "rawbody".

Note: To be able to save data in rawbody format, the original request must be made with the "Include raw data" checkbox selected.

Once configured, the service can be started with the command

```
./StreamClientService.sh start
```

Note that if the client does not receive any packets over a period of time, no new file will be generated for that range.

Being a service, it will not stop automatically. To stop it, launch the command:

```
./StreamClientService.sh stop
```

The service will save the last file it was working on (if there are data available) and will be closed.

3.6.7 Start the Performance Processor (optional)

This step is only needed if it is required to analyse the time taken to process a batch request. When the process is stopped, the results are output to the log file.

Browse to the location of the Performance Processor installation as specified by the *edds.performance.processor.home* property

Check the config.properties file of the EDDS Server, ensuring that the property "enable.performance.test" is set to "true"

execute the command

```
./PerformanceProcessorService.sh start
Started.
```

The Performance Processor can be stopped by running:

Browse to the location of the Performance Processor installation as specified by the *edds.performance.processor* property and execute the command

```
./PerformanceProcessorService.sh stop
Kill signal sent
```

3.7 EDDS MMI

EDDS provides two ways to interact with the backend via an MMI. First a standalone Java application based on Eclipse RCP platform and second, a web application based on Eclipse RAP platform. Both are compiled and built as part of the main EDDS compilation and deployment process.

3.7.1 Standalone EDDS MMI

Once the Runtime distribution has been generated, at least three zip files will be created in the generated runtime directory under the “mmi” folder. One contains the Linux 64 bit version of the MMI, and the two other zip files contain the Windows 32 and 64 bits distributions.

Also Linux and Windows EDDS MMI Java Web Start Clients are deployed to the web server during the deployment of EDDS components.

3.7.1.1 Install the MMI

In order to install the MMI, extract the MMI zip file.

Open the Edds.ini file and configure the properties as per Section 3.7.3.

3.7.2 EDDS MMI Web Application

Once the web application has been deployed, the application can be accessed by pointing the web browser to the application URL e.g.

`http://10.48.18.86:8080/eddsweb/eddsMmi`

The URL contains the hostname and port of the Tomcat installation, `eddsweb` is the name of the deployed WAR file without the .war extension in `$CATALINA_HOME/webapps/` directory. `/eddsMmi` is mandataory part of the URL, it’s the name of the servlet serving the web application.

3.7.3 Runtime properties

The client application needs some runtime properties to be set in order to work. These properties are listed in the EDDS configuration Edds.ini under the EDDS MMI directory

EDDS_HOME	The absolute path to the directory where the client will store/read locally the request files. This directory will be used by the Request View Display (as explained in the SUM document) and it is mandatory. If this property is mis-configured an error will occur when starting the application.
edds.server.endpoint.url	The HTTP/HTTPS URL to the EDDS Web Service (e.g. http://esa.egos.int/edds/EddsService?wsdl https://esa.egos.int:8443/edds/EddsService?wsdl) This will be used as a default EDDS endpoint address when the MMI is opened, the clients can override this setting in the running application in MMI Preferences.
javax.net.ssl.trustStore	The location of truststore that contains the certificate for the EDDS web server. Used for SSL connection. See Section 3.10.2 for more info.
javax.net.ssl.trustStorePassword	Password for the given truststore. Used for SSL connection.

status_view_start_time_offset	This property can be optionally set to change the initial start time used in the Request Summary View. By default, the past 7 days will be used. The value must be entered in the “duration” format – i.e. “-PnYnMnDTnHnMnS”. A value of “-P7D” would be the past 7 days.
status_view_end_time_offset	This property can be optionally set to change the initial end time used in the Request Summary View. By default, the next 7 days will be used. The value must be entered in the “duration” format – i.e. “PnYnMnDTnHnMnS”. A value of “P7D” would be the next 7 days.
http.proxyHost	Optional. If connecting via a proxy server, the hostname or IP address of the proxy server should be entered here.
http.proxyPort	Optional. If connecting via a proxy server, the port of the proxy server should be entered here. Must be an Integer value and must be specified if the http.proxyHost entry has been specified.
http.proxyUser	Optional. The username to connect to the proxy server if the proxy server requires authentication.
http.proxyPassword	Optional. The password to connect to the proxy server if the proxy server requires authentication. Must be specified if the http.proxyUser has been specified.
http.maxBufferSize	Optional. The string representiation of a strictly positive integer corresponding to the maximum number of bytes of an HTTP response. Reponses exceeding this limit will be discarded.The default value is 33554432.
Limit	Optional. The number of rows to limit the Request Summary View to. Default is 5000. On the standalone client, can also be set to -1 to disable the feature.

Table 3-7 Runtime properties

3.8 Pure-FTP Configuration

Before FTP can be used with EDDS, the following modifications are required to the configuration. Pure-FTPd must be installed on the same machine as the Delivery Manager:

1. Open the Pure-FTPd configuration file, usually found in `/etc/pure-ftpd/pure-ftpd.conf`, as root
2. Ensure “ChrootEveryone” is set to “yes”
3. Ensure “NoRename” is set to “no”
4. Ensure “PAMAuthentication” is commented out.
5. Uncomment the line “LDAPConfigFile” and set it to “`/home/edds/EDDS_RUNTIME/DELIVERY_MGR/pureftpd-ldap.conf`”. This file is created during the deployment of the delivery manager. Change the location to match the one for your installation.
6. Restart the FTP daemon with `/etc/init.d/pure-ftpd restart`

This will enable users to log-on to the FTP service using their EDDS account in LDAP. The required directories are created automatically by the Web Server when the entries are created in LDAP.

3.9 Configuration for SFTP usage

The remote hosts that are going to be used have to be in the list of known hosts. This can be performed as a one-off by logging onto the server running the delivery manager, then entering:

```
ssh -o HostKeyAlgorithms=ssh-rsa user@remotehost
```

At the command line. When prompted, enter “yes” to add the remote host to the list of known hosts. If this is not done, the connection to the remote host will be rejected when the Delivery Manager attempts to make the connection.

3.10 Configuring EDDS web services for SSL (HTTPS)

Configuring EDDS to use secure connection between the EDDS MMI and the EDDS web server is a 2-sided-process. You'll have to set up tomcat to accept SSL connections and the client (EDDS MMI) to trust the web server's certificate.

The examples here are based on the tool that comes with standard Java distribution. Other setups are also possible.

3.10.1 Configure the EDDS Web Server

Please refer to [AD-3] for more detailed instructions.

The web server is going to provide its clients a certificate to prove its authenticity and a public key that the client can encrypt its initial connection with. Only the holder of the corresponding private key (the server) can decrypt the data.

3.10.1.1 Generate a key pair and store it

Bare minimum:

$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA

Password for the keystore: *changeit*

When it asks for firstname and lastname (CN field on the certificate), you'll have to enter the name of the server you will use the certificate in. The clients will refer to this server in their EDDS web service client configuration.

Password for the key (same as keystore): *changeit*

This will generate a keystore at {$USER_HOME}/.keystore and store the keypair there. The key is initially self-signed.

Usually it is not necessary to generate these keys yourself – they should be provided to you by your security authority.

3.10.1.2 Get your public key certified by a Certificate Authority (CA)

This step is not required if self-signed certificate is used, but generally certification is strongly recommended otherwise the client has no way of verifying whether the server who it is trying to connect to is actually the one that it claims to be.

The certification steps generally involve

- generating a Certificate Signing Request (CSR) for your (newly generated) public key:

  Bare minimum:

  $JAVA_HOME/bin/keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <your_keystore_filename>
- receiving a certificate from CA for your key
- importing the Chain Certificate of the CA to your keystore
- importing your received certificate to your keystore (for publishing it to the clients)

Refer official documentation from Tomcat for more details https://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html

3.10.1.3 Add a new connector to Tomcat

Add a new connector to listen for the secure connections using the keys and certificates from the previous step.

Bare minimum:

Uncomment the default SSL connector in {$CATALINA_HOME}/conf/server.xml and add keystore related configuration and use TLSv1.2 or higher. Do not use TLS version below 1.2:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           connectionTimeout="20000" maxThreads="150" SSLEnabled="true" scheme="https" secure="true">
      <SSLHostConfig>
```

```
    <Certificate certificateKeystoreFile="/home/edds/.keystore"
certificateKeystorePassword="changeit"
         type="RSA" certificateVerification="none" protocols="TLSv1.2,TLSv1.3" />
  </SSLHostConfig>
</Connector>
```

Then create the file {$CATALINA_HOME}/bin/setenv.sh:

```
#!/bin/sh
export JAVA_OPTS="-Djavax.net.ssl.trustStore=<keystore complete path> -
Djavax.net.ssl.trustStorePassword=changeit -Dosgi.nl.user=en_GB"
```

Save the file and make it executable with *chmod +x*.

Restart the web server for the changes to take effect.

3.10.1.4 Exporting the certificate

The clients will need to add the server certificate (or one of its authorities' certificates) to their truststore. To export certificate from the keystore:

```
$JAVA_HOME/bin/keytool -export -keystore ~/.keystore -alias tomcat -file tomcert.cer
```

This exports the certificate with alias tomcat to a file *tomcert.cer*.

3.10.2 Configure the EDDS client

For the client to be convinced that it is talking to a legitimate server (to trust the server), the server must provide the client with a certificate that the client trusts or a certificate, that is signed by a certificate that the client trusts. The certificates that the client trusts are stored in its truststore. The truststore can be somewhat compared to SSH's "known hosts".

By default, a Java program does not trust any certificate. There's no truststore set up by default.

The client needs to:

- Import at least one of the certificates in the server Certificate Chain to the client's truststore. You can use the server's keystore as the client's truststore, but generally, the client does not need the private key to be present in the truststore just the public key (with its certificate).
- Use the EDDS web server's secure protocol address instead of plain HTTP for communication.
- Configure in the EDDS web server's preferences the SSL server endpoint: Window->Preferences->EDDS preferences. Insert in the Server Endpoint field the value "https://<hostname>:8443/edds/EddsService?wsdl

3.10.2.1 Importing a certificate to truststore

To import the server certificate to the client's truststore:

```
$JAVA_HOME/bin/keytool -import -file tomcert.cer -alias tomcat -keystore truststore.ts -storepass
tspassw
```

If the truststore doesn't already exist it will be created with the given password and stored in file *truststore.ts*.

3.10.2.2 Configuring EDDS Standalone MMI

Set system properties to use appropriate truststore and point to the secure address of the EDDS web service.

Bare minimum:

Open Edds.ini in EDDS_MMI runtime directory and change the properties:

- Set the service endpoint to https and the right port. The server name must be the same as common name (CN) specified on the server certificate.

  *-Dedds.server.endpoint.url=**https**://<Host Path>:**8443**/edds/EddsService?wsdl*

Note that “Host Path” must be the same of CN field used to generate the certificate.

- Refer to your truststore using absolute path to the filename and provide the truststore password.

  -Djavax.net.ssl.trustStore=/path/to/truststore.ts

  -Djavax.net.ssl.trustStorePassword=tspassw

- Alternatively, refer to the Java Runtime Package CA certificates.

  -Djavax.net.ssl.trustStore=/path/to/JAVA_HOME/lib/security/cacerts

  -Djavax.net.ssl.trustStorePassword=changeit

3.11 Logging

3.11.1 Introduction

Logging for all of the server components of EDDS is performed using Log4j. Log4j is a well-proven logging system that allows the operator to change the logging level at runtime without changing the application code. It also enables the log messages to be stored in different locations, which are also customisable at runtime. By default, EDDS is deployed with logging at info level and log messages are stored in a rolling log file and to JMS topics. The EDDS Archiver listens to the JMS logging topics and saves the log messages into the database. The database log messages are used in conjunction with the Historical Log Messages display within the MMI to display these log messages to the user without having to gain access to the server itself. A custom EddsJmsAppender sends all the log messages to ActiveMQ topics for the EUD Live Message Display. Other clients can also request and download the log messages with the Web WSDL interface. And subscribe to live messages via Client API.

3.11.2 EDDS Web Server

The logging for the EDDS Web Server can be customised by editing the log4j configuration file. This can be found in the Apache Tomcat installation directory under webapps/edds/WEB-INF/log4j.xml. The logging level can be changed by changing the*root* entry. The possible options are:

- trace (very low level messages are displayed)
- debug (messages useful for debugging are displayed)
- info (only informative messages are displayed)
- warn (only warning messages are displayed)
- error (only error messages are displayed)
- fatal (only serious errors are displayed)

The other options are configured automatically during deployment, and shouldn’t be changed.

3.11.3 EDDS Server

The logging for the EDDS Server can be customised by editing the log4j.xml configuration file. This can be found in the directory the EDDS Server was deployed to in a file called log4j.xml. The logging level can be changed by editing the “value” attribute for the XML element “level” within “root”. The possible options are as above.

3.11.4 EDDS Delivery Manager

The logging for the EDDS Delivery Manager can be customised by editing the log4j.xml configuration file. This can be found in the directory the EDDS Delivery Manager was deployed to in a file called log4j.xml. The logging level can be changed by editing the “value” attribute for the XML element “level” within “root”. The possible options are as above.

3.11.5 EDDS Request Submitter

The logging for the EDDS Request Submitter can be customised by editing the log4j.xml configuration file. This can be found in the directory the EDDS Request Submitter was deployed to in a file called log4j.xml. The logging level can be changed by editing the “value” attribute for the XML element “level” within “root”. The possible options are as above.

3.11.6 EDDS Archiver

For the log messages to be archived into the database, the EDDS Archiver must be running.

The logging for the EDDS Archiver can be customised by editing the log4j.xml configuration file. This can be found in the directory the EDDS Archiver was deployed to in a file called log4j.xml. The logging level can be changed by editing the “value” attribute for the XML element “level” within “root”. The possible options are as above.

3.11.7 Audit log

For the security reasons EDDS components have special logger configured for the audit log. Audit log entries are marked with logger name “esa.egos.edds.audit” in log output. Audit log entries contain information about important events from security perspective - login attempts, account creation and changes, request submissions, component startups and shutdowns etc. Audit log entries are written by default to the log files and not sent to the JMS topic to avoid making them visible in MMI. This behaviour can be changed as with any other log messages using log4j configuration files mentioned above.

Important: audit log contains sensitive information like usernames, request and session identifiers. Administrators have to be extra careful to make sure this information is not exposed to unauthorized users when changing default logging configuration.

3.12 Installing Secure File Transfer (SFT) for use in EDDS

Full installation details are outside the scope of this document, follow the instructions described in SFT installation guide.

SFT is used as a mechanism for the transfer of response files in a scenario where the EDDS Server and Delivery Manager are running on separate machines.

The machine running the EDDS Server will transfer files using the 'Sender' component of SFT. The 'Sender' will poll the ''edds-server-completed" folder for new response files.

A 'Receiver' listening on the same port as defined for the 'Sender' will be running on the Delivery manager's machine.

SFT will transfer all files to the 'edds-server-completed' folder on the Delivery manager's machine. The following property values need to be set for each sender and receiver to ensure files are transferred to and from the correct locations.

Sender.properties

Property	Description
sender.in_tray	The location of 'edds-server-completed' on the machine running the EDDS Server *Example value:* *{User Home Dir}*//edds/EDDS_RUNTIME/ EDDS_HOME/edds-server-completed

Receiver.properties

Property	Description
receiver.target_destination	The location of 'edds-server-completed' on the machine running the Delivery Manager/ Web Server *Example value:* *{User Home Dir}*//edds/EDDS_RUNTIME/ EDDS_HOME/edds-server-completed

3.13 Handling Multiple Missions

The EDDS delivery manager can serve multiple missions. The delivery manager will attempt to deliver all files that are transferred to its 'edds-server-completed' directory.

Unless all missions are running on the same machine as the Delivery manager, one or more instances of SFT's Sender and Receiver will be required.

3.13.1 Configuration Scenarios

Multiple missions on the same remote machine – in this scenario it is possible to use only one instance of a Sender and Receiver. Both missions must deliver response files to a shared folder (default configurations will achieve this). An SFT Sender will poll this directory and transfer all files to the Receiver running on the Deliver Manager's machine.

Multiple missions on separate machines - in this scenario each machine will need to run its own instance of the SFT Sender. The Delivery Manager's machine will need to run a Receiver for each Sender. It is not possible to use the same Receiver for multiple Senders, as each set of Sender and Receiver needs to connect using their own port. Every Receiver instance must set the 'target_destination' to the 'edds-server-completed' directory.

4. Mission Specific Configuration

4.1 EGSCC Missions

EGSCC missions do not necessarily need any mission specific configuration at the moment. The only thing that should be done is to create a domain with the same name as the dataspace of the mission defined in the Hbase cluster. The EGSCC specific configuration files are available under 'config/HARC' folder into the EDDS_SERVER.

4.2 EDDS Server Configuration Properties

4.2.1 Introduction

The EDDS Server must be configured for the SCOS 2000 mission it is to retrieve data from. This configuration information is stored in a configuration file. EDDS knows all of the properties it requires, and if any are missing from the configuration file that it loads, a warning is displayed in the log file and a suitable default is taken instead. This approach helps to ensure that all configurations can be kept in one place, making configuration and maintenance easier.

The following section lists each of the properties that the EDDS Server requires, along with the default that is taken should the configuration item not be present in the configuration file and a brief description. The sections that follow the table describe the more complex configuration items in more detail.

4.2.2 EDDS Properties

Property	Description	Default Value
esa.egos.edds.acknowledgement.mail.enabled	If set to "true", e-mails will be sent, otherwise e-mails won't be sent	true
esa.egos.edds.acknowledgement.mail.debug	If true, enables the debug information in java.mail	false
esa.egos.edds.acknowledgement.mail.smtps.enabled	If set to "true", e-mail messages will be sent using SMTPs instead of SMTP.	false
esa.egos.edds.acknowledgement.mail.smtps.auth	Set to "true" if SMTPs server requires authentication	true
esa.egos.edds.acknowledgement.mail.smtps.password	The password to be used to authenticate to the SMTPs service	1Logica!
esa.egos.edds.acknowledgement.mail.smtp.user	Define the SMTP user that will be used to send e-mails	no-reply@esa.esoc.edds
esa.egos.edds.acknowledgement.mail.transform	If set to "true", e-mails will be transformed using XSLT before they are sent	true
esa.egos.edds.jms.receive.timeout	The amount of time (in milliseconds) to wait for a reply message to be received (e.g. from the Web Server to EDDS Server)	5000
enable.performance.test	If set to "true", timestamps at various stages of the EDDS Server will be logged	false
report.file	The location of the results file for the performance processor	results.csv
EDDS_LOG4J_CONFIGURATION_FILE	The location of the EDDS Server log4j configuration file	log4j.xml
esa.egos.edds.xsd_file	The location to the EDDS schema file for batch requests and user requests. The default is to load the schema within the EDDS Server Jar file.	edds.xsd

esa.egos.edds.pkt_structure_xsd_file	The location to the EDDS Server schema file for the structure of TC packets and the packet headers. The default is to load the schema within the EDDS Server Jar file.	esa/egos/edds/edds-server.xsd
MISSION_NAME	The name of the mission that the EDDS Server instance will process requests for. This must match the name of the mission specified in LDAP. This property is optional. If omitted, the EDDS Server will process requests for the first mission it finds in LDAP when starting up.	
esa.egos.edds.edds_server.working_dir	The full path to the location where EDDS Server can store files it is working on. The name should not end in a directory separator character and should be a directory and not a file. If there are any files in the directory they will be removed during the EDDS server start up.	
esa.egos.edds.edds_server.outbox	The full path to the location where EDDS Server will place files that can be transferred for delivery by the Secure File Transfer program. The name should not end in a directory separator character and should be a directory and not a file.	
SUPPORTED_BATCH_REQUESTS	The list of batch request types supported by the EDDS Server instance.	No default. Must be specified in properties file.
SUPPORTED_DATA_SOURCES	The list of data source requests that the EDDS Server will serve.	No default. Must be specified in properties file.
SUPPORTED_STREAM_REQUESTS	The list of stream request types supported by the EDDS Server instance.	No default. Must be specified in properties file.
PKT_DEFAULT_TM_DATA_STREAM	The TM data partitions supported by the PARC for this mission. Requests that specify the data stream will be checked against this list.	1,2
PKT_DEFAULT_TC_DATA_STREAM	The TC data partitions supported by the PARC for this mission. Requests that specify the data stream will be checked against this list.	1
PKT_DEFAULT_EV_DATA_STREAM	The EV data partitions supported by the PARC for this mission. Requests that specify the data stream will be checked against this list.	1,2
PARC_CORBA_SERVER_NAME	IP address or host name of the SCOS 2000 CORBA server for connecting to the PARC.	
PARC_CORBA_PORT	Port number of the SCOS 2000 CORBA server for connecting to the PARC	28200
PARC_CORBA_SOURCE_SERVER_FAMILY_NAME	The name of the CORBA Server family for the PARC. Can be found in MISCcontext under OPERATIONAL_SERVER_FAMILY	PRIME

PARC_CORBA_APP_NAME	The name of the PARC Manager application as registered in the Name Server	PAMASviews
PARC_CORBA_ARCHIVE_MANAGEMENT	The name of the PARC Archive Management service as registered in the Name Server	PAMASmanageArchive
PARC_EV_STREAMING_DATA_PARTITIONS	Name(s) of the partion(s) used in the subcription to S2K Out of Limits & Events Messages Provision Services. See §5.2 & §5.3 of **Error! Reference source not found.**	Real-Time,Playback
PARC_TM_STREAMING_DATA_PARTITIONS	Name(s) of the partion(s) used in the subcription to S2K Packet Provision Service. See §5.4 of **Error! Reference source not found.**	Real-Time,Playback
PARC_STREAMING_CORBA_SERVER_NAME	IP address or host name of the SCOS 2000 CORBA server for connecting to the PARC for streaming live data ONLY.	
PARC_STREAMING_CORBA_PORT	Port number of the SCOS 2000 CORBA server for connecting to the PARC for streaming live data ONLY.	28200
PARC_STREAMING_CORBA_SOURCE_SERVER_FAMILY_NAME	The name of the CORBA Server family for the PARC for streaming live data ONLY. Can be found in MISCcontext under OPERATIONAL_SERVER_FAMILY	PRIME
edds.parc.dataspace	The PARC dataspace to use if one isn't specified in the request (i.e. "EDDS Default" is selected in the EDDS MMI) and the edds.parc.dataspaces.enabled property is set to true This property can be commented out or left empty. In which case, the PARC will use the dataspace set as current.	(empty)
edds.parc.dataspaces.enabled	Is the new PARC API used that has the dataspaces functionality enabled (SCOS 5.4.18 or higher)	true
edds.parc.bigendian.enabled	Should the PARC output be big endian? (SCOS 5.5.4 or later)	false
TC_PKT_CMD_LEN_PAR_DESCR	The value of the constant CMD_LEN_PAR_DESCR in SCOS (model-commanding-rep CMDrqstPktDataDetails.H or CMDdataDetails.idl)	24
TC_PKT_CMD_LEN_PAR_VAL_UNIT	The value of the constant CMD_LEN_PAR_VAL_UNIT in SCOS (model-commanding-rep CMDrqstPktDataDetails.H or CMDdataDetails.idl)	4
TC_PKT_CMD_LEN_PAR_SET_NAME	The value of the constant CMD_LEN_PAR_SET_NAME in SCOS (model-commanding-rep CMDrqstPktDataDetails.H or CMDdataDetails.idl)	8

TC_PKT_CMD_LEN_PAR_INFO	The value of INFO field length from SCOS (model-commanding-rep CMDrqstPktDataDetails.H or CMDdataDetails.idl)	2 for SOCS < 6.2.0 3 for SCOS >= 6.2.0
TM_HDR_CUSTOM_SIZE	Value, in short, of extra data added in the TM header compared to s2k vanilla. Increasing the value by 1, will skip over 0000.	For vanilla SCOS = 0. Eg. Use the value 2 if you want to skip 00000000 from extra mission specific elements.
COMPRESSION_LEVEL	The default compression level for Zip files. Options are: BEST_COMPRESSION (smallest file size but takes longer to complete) BEST_SPEED (fastest compression speed, but results in a larger file size) DEFAULT_COMPRESSION (the default level that is optimised for compression and speed) NO_COMPRESSION (the files are simply stored in the Zip file)	DEFAULT_COMPRESSION
MAX_NUMBER_OF_PARALLEL_REQUEST	The maximum number of batch requests that can be run in parallel	10
esa.egos.edds.provider.farc.max_number_of_requests	The maximum number of requests to FARC that can be run in parallel	
esa.egos.edds.provider.darc.max_number_of_requests	The maximum number of requests to DARC that can be run in parallel	
esa.egos.edds.provider.parc.max_number_of_requests	The maximum number of requests to PARC that can be run in parallel	
esa.egos.edds.provider.data_provision.max_number_of_requests	The maximum number of requests to the Data Provision service that can be run in parallel	
esa.egos.edds.provider.fs.max_number_of_requests	The maximum number of requests to the File System service that can be run in parallel	
MAX_NUMBER_OF_REPEATING_SCHEDULED_REQUESTS	The maximum number of scheduled requests that can be generated in a repeating schedule. Should be placed in the EDDS Archiver properties file.	100
MAX_NUMBER_OF_STREAMING_REQUEST	The maximum number of stream requests that can be run in parallel	10
STREAMING_THREAD_POOL_SIZE	The number of threads on the Camel route for consuming messages on the topic for each stream request. Increase this value if EDDS is not consuming messages fast enough, but be aware memory usage will increase.	5
STREAMING_MAX_THREAD_POOL_SIZE	The maximum number of threads on the Camel route for consuming messages on the topic for each stream request. Increase this value if EDDS is not consuming messages fast enough, but be aware memory usage will increase.	10
COMPLETED_FILES_OWNER_ONLY_WRITABLE	Flag as to whether files created by EDDS Server should be writable only by the user running the server	false

esa.egos.edds_server.enable.diskspace	The flag to enable/disable diskspace check. 'false' to disable and 'true' to enable diskspace check. The default value is 'false'.	false
esa.egos.edds_server.diskspace_check_period	Configures the period for EDDS server disk space check when a request is Active. This property will be in seconds. The default value is 60 seconds.	60
jms.url	The ActiveMQ message broker connection string	failover:(tcp://localhost:61616)?timeout=3000&trackMessages=true
jms.username	The (optional) username to connect to the ActiveMQ message broker with.	
jms.password	The (optional) password to connect to the ActiveMQ message broker with.	
darc.database.driver	The JDBC driver to use to connect to the DARC database.	com.mysql.jdbc.Driver
darc.database.url	The database connection URL to connect to the DARC database.	Populated at deploy time, e.g. jdbc:mysql://localhost/DARC_DB
darc.database.username	The username to connect to the DARC database	Populated at deploy time
darc.database.password	The password to connect to the DARC database	Populated at deploy time
darc.dataspace	The DARC dataspace to use	(empty) – use default dataspace
edds.darc.post.2.3.0	How to interpret timestamps received from the DARC. For DARC v2.3.0 and later, this should be set to true, as the timestamps are stored to microsecond precision. For DARC v2.2.1, this should be set to false, as timestamps are stored to millisecond precision.	Populated at deploy time
darc.retrievallimit	The number of samples to retrieve per query, if the retrievalAll or retrieveAllByType queries are used. If there are more samples in the given time range than the limit, multiple queries are run, until all of the samples are retrieved.	10000
darc.jms.url	The message broker to connect to for the DARC stream data	failover:(tcp://localhost:61616)?timeout=3000&trackMessages=true
darc.jms.username	The username for the message broker (blank if none needed)	
darc.jms.password	The password for the message broker (blank if none needed)	
esa.egos.edds.darc.stream.topic	The name of the topic that the parameters are placed on from the DARC	esa.egos.darc.realtimeParams
esa.egos.edds.darc.stream.simulator.paramnames	Used by the DARC Parameter stream simulator. Lists the possible parameter names that the simulator can randomly choose from.	

esa.egos.edds.darc.stream.enablefilterbytype	If set to true, filtering on the parameter value will only match when the types match. For example, when set to true, a filter value for a Float value of >0.2 will only match parameters whose value type is also Float and >0.2. If set to false, a filter value for a Float value of >0.2 would match Integer, Float and Double values that are >0.2. This should only be set to true if this is the desired mode of operation *and* the DARC sends this information in the header of the parameter message.	false
edds.smon.corba.servers	List of CORBA servers with port for SMON Eg: ip:port, ip:port, ip:port	
edds.smon.mnemonic	The name of the SMON service for EDDS configured in the TKMA file in SCOS. See Appendix B.	SMON_EDDS_1
edds.smon.family	The family name that the SMON process	PRIME
edds.smon.instance.number		0
edds.smon.arguments	The arguments to pass to the SMON process when starting. See Appendix B.	EDDS 1
edds.smon.env	Environment variables for the SMON process. Usually leave empty.	
edds.smon.screen.number	The screen number to start the SMON process on. Can be left as 0 as it has no GUI.	0
edds.smon.unique.id	The SMON unique ID	
edds.smon.startup.timeout	The time in seconds to wait to be notified that the SMON process has started	30
edds.smon.tkma.mnemonic	The name of the CORBA service holding the TKMA process list. Can usually be left as the default.	TKMAprocessList_1
edds.smon.tkma.server.family	The TKMA server family.	PRIME
edds.smon.tkma.server.family.name	The TKMA server family name.	PRIME
edds.smon.tkma.domain	The domain that the SMON process is configured for	0
edds.smon.startup.wait	The time in seconds to wait for the SMON process to get ready. The SMON process is not usually ready to receive connections as soon as EDDS has been notified by the TKMA service that the SMON service is ready, and so a delay is required before EDDS attempts a connection.	1
esa.egos.edds.delivery_manager.log4j_config_file	The location of the Delivery Manager log4j configuration file	log4j.xml
esa.egos.edds.delivery_manager.thread_pool_size	The size of the pool containing threads to perform parallel operations. When this size is exceeded, requests are queued.	10

esa.egos.edds.delivery_manager. inbox	The directory where the Delivery Manager expects to find the finished files that have been transferred via the Secure File Transfer system.	
esa.egos.edds.delivery_manager. working_dir	The working directory of the delivery manager	
esa.egos.edds.delivery_manager. failed_dir	A directory for files that failed the delivery	
esa.egos.edds.delivery_manager. file_server.missions_dir	The file server directory where files tagged with a PRIVACY_TAG of MISSION are stored	
esa.egos.edds.delivery_manager. file_server.roles_dir	The file server directory where files tagged with a PRIVACY_TAG of ROLE are stored	
esa.egos.edds.delivery_manager. file_server.users_dir	The file server directory where files tagged with a PRIVACY_TAG of PRIVATE are stored	
esa.egos.edds.delivery_manager. file_server.public_dir	The file server directory where files tagged with a PRIVACY_TAG of PUBLIC are stored	
esa.egos.edds.delivery_manager. intervals_between_attempts	The time (in seconds) to wait between each FTP delivery attempt should the remote server be unavailable. *This property is set within the mission specific configuration file.*	60
esa.egos.edds.delivery_manager. max_attempts	The maximum number of attempts the delivery manager will try to deliver the response file when errors in delivery occur. *This property is set within the mission specific configuration file.*	1

esa.egos.edds.delivery_manager. multiple_targets_total_time	The total time (in seconds) to wait when sending a file to multiple targets.
	This value should be at least 1 seconds greater then (esa.egos.edds.delivery_manager.max_attempts + 1) * ((SFTP|FTP)_CONNECTION_TIME_OUT + esa.egos.edds.delivery_manager.intervals_between_attempts)
	Example:
	esa.egos.edds.delivery_manager.max_attempts = 2
	FTP_CONNECTION_TIME_OUT = 120
	esa.egos.edds.delivery_manager.intervals_between_attempts = 1
	esa.egos.edds.delivery_manager.multiple_targets_total_time = (3) * (121) + (1)= 364
	This property is set within the mission specific configuration file.
		300

esa.egos.edds.delivery_manager_ mission_specific_config_directory	The directory of all mission specific configuration files	mission_specific_config
esa.egos.edds.delivery_manager. polling_interval	How often to wait, in seconds, before checking the polling directory for new files	1
esa.egos.edds.delivery_manager. sftp_usage	Whether to use SFTP instead of FTP for file delivery *This property is set within the mission specific configuration file.*	false

esa.egos.edds.delivery_manager.sftp_port	The port number witch to use for SFTP	22
esa.egos.edds_server.batch.split	Response file size in megabytes where supported file formats should be split at or -1 for no splits.	-1
esa.egos.edds_server.batch.response.count	The number of files in each zip/response of 'FileSystem File' request. The default value is '1'. This means there will be one file in each zip/response of 'FileSystem File' request when the response contains several files. Set to -1 for disabling the feature.	-1
FTP_LINUX_ACCOUNT_NAME	The Linux user account that hosts the virtual FTP user accounts. Must have ownership of the FTP_BASE_HOME_DIRECTORY	edds
FTP_LINUX_ACCOUNT_ID	The Linux user account ID number of FTP_LINUX_ACCOUNT_NAME. This can be discovered by logging on to the account and entering "id"	1061
FTP_LINUX_GROUP_ID	The Linux group ID number of FTP_LINUX_ACCOUNT_NAME. This can be discovered by logging on to the account and entering "id"	100
FTP_USE_ACTIVE_MODE	The FTP mode to use. This can either be active or passive. To use active mode, set FTP_USE_ACTIVE_MODE to true, to use passive set FTP_USE_ACTIVE_MODE to false	true
FARC_CONFIGURATION_FILE	The location of the FARC configuration file. This file (model.config) contains the location of the orb_configuration.xml file (that details the CORBA name server for the FARC), the farc_events.xml file and the location to place checked out files from the FARC.	config/FARC/model.config
FARC_USERNAME	The username to connect to the FARC with. This user doesn't actually have to exist, it is simply used by the FARC to build a unique entry in the CORBA broker.	s
FARC_SERVER_DOMAIN	The domain the FARC Server is running on in SCOS.	0
FARC_APPLICATION_NAME	The identifier of EDDS to the FARC	FarcEddsDriver
ARCHIVE_INSTANCE_ID	The name of the instance ID to look for files in the FARC. Typical values are OPERATIONAL, LOCAL, HOLDING, EDITING, PRIME	PRIME

ARCHIVE_ZONE_ID	The name of the archive zone ID for multi-zone FARC installations. Optional - can be omitted or left empty. Important Note: The FARC Zone ID specified here should match one of the Zone IDs specified in the orb_configuration.xml file in the EDDS Server within config/FARC. However, it is recommended to leave the ARCHIVE_ZONE_ID property empty unless you are running FARC v2.2.0 or later due to SPR farc#212. The default zone ID will be used instead as long as only one zone has been specified in the orb_configuration file.	
CHECKOUT_DIRECTORY	The temporary directory that the FARC checks files out to.	~/FARC/checkout
FARC_TIMEOUT	The number of seconds to wait for the FARC to checkout files from the request	30
FARC_FOLDER_RECURSION	If "true", files within subfolders will also be checked out from the FARC	true
farc.jms.enabled	If "true", the FARC Subscription notifications will be enabled	true
PARC_CORBA_PACKET_PROVISION	The name of the Packet Data Provision Service in the CORBA name service. This is usually “PacketProvisionService_” followed by the hostname of the machine running SCOS.	PacketProvisionService_gimus106
PARC_CORBA_COMMAND_PROVISION	The name of the Command Data Provision Service in the CORBA name service. This is usually “CommandProvisionService_” followed by the hostname of the machine running SCOS.	CommandProvisionService_gimus106
PARC_CORBA_EVENT_PROVISION	The name of the Event Data Provision Service in the CORBA name service. This is usually “EventProvisionService_” followed by the hostname of the machine running SCOS.	EventProvisionService_gimus106
PARC_CORBA_OOL_PROVISION	The name of the OOL Data Provision Service in the CORBA name service. This is usually “OolProvisionService_” followed by the hostname of the machine running SCOS.	OolProvisionService_gimus106
PARC_STREAMING_CORBA_PACKET_PROVISION	The name of the Packet Data Provision Service in the CORBA name service for streaming live data ONLY. This is usually “PacketProvisionService_” followed by the hostname of the machine running SCOS.	PacketProvisionService_gimus106
PARC_STREAMING_CORBA_COMMAND_PROVISION	The name of the Command Data Provision Service in the CORBA name service for streaming live data ONLY. This is usually “CommandProvisionService_” followed by the hostname of the machine running SCOS.	CommandProvisionService_gimus106

PARC_STREAMING_CORBA_EVENT_PROVISION	The name of the Event Data Provision Service in the CORBA name service for streaming live data ONLY. This is usually "EventProvisionService_" followed by the hostname of the machine running SCOS.	EventProvisionService_gimus106
PARC_STREAMING_CORBA_OOL_PROVISION	The name of the OOL Data Provision Service in the CORBA name service for streaming live data ONLY. This is usually "OolProvisionService_" followed by the hostname of the machine running SCOS.	OolProvisionService_gimus106
MIB_FILES_LOCATION	The location of the MIB files for each domain. This is just the base location, and will be appended with the domain ID and filename. For example, if you set this location to be "/home/edds/EDDS_HOME/MIB" then the EDDS Server will look for the following files: <domain>/pid.dat <domain>/tpcf.dat <domain>/CMD_SUBSYSTEM_FILE <domain>/CMD_HOST_FILE <domain>/MISCbase.ids <domain>/MISCcontext_SHARED.sta <domain>/CMDsourceTypeMapping.properties Where "<domain>" is replaced with the domain ID for each domain configured.	config/MIB
RAW_TM_PACKET_HEADER	The location of the XML file containing the default header of EDDS telemetry packets in raw form. Refer to the EDDS EUICD for details on how this file is created.	config/MIB/defaultTmPacketHeader.xml
RAW_TC_PACKET_HEADER	The location of the XML file containing the default header of EDDS telecommand packets in raw form. Refer to the EDDS EUICD for details on how this file is created.	config/MIB/defaultTcPacketHeader.xml
RAW_EV_PACKET_HEADER	The location of the XML file containing the default header of EDDS event message packets in raw form. Refer to the EDDS EUICD for details on how this file is created.	config/MIB/defaultEvPacketHeader.xml
TC_PACKET_SPID_LIST	The list of SPIDs that contain telecommands that EDDS can decode. Entries are separated with a colon.	100
EV_LOG_PACKET_SPID	The SPID of the event packet containing event log messages	1000
EV_OOL_PACKET_SPID	The SPID of the event packet containing event out-of-limit messages	3000
PARC_SPACECRAFT_EVENTS_TYPE	The PUS service type for Spacecraft Events TM packets	5
PARC_SPACECRAFT_EVENTS_SUBTYPES	The PUS service sub-type(s) for Spacecraft Events TM packets	1,2,3,4

LDAP_CONNECTION_URLS	The LDAP connection URL(s) of the Master LDAP server(s). Format is ldap://<hostname>:<port> where <hostname> is the hostname or IP of the LDAP server, and <port> is the port number that the Master LDAP server is running on. Default is 389. Multiple entries can be made separated with commas.	ldap://localhost:389
LDAP_ADMIN_DN	The LDAP Administration user DN. If not specified, anonymous access will be used.	
LDAP_ADMIN_PASS	The LDAP Administration password. If not specified, anonymous access will be used.	
edds.ws.user.password.expiry.period	Defines the period after last password change when the current password expires. After that users will have to change their password. The time interval is specified in the following XML duration form "PnYnMnDTnHnMnS" where: P indicates the period (required) nY indicates the number of years nM indicates the number of months nD indicates the number of days T indicates the start of a time section (required if you are going to specify hours, minutes, or seconds) nH indicates the number of hours nM indicates the number of minutes nS indicates the number of seconds	P6M
esa.egos.edds.edds_filesystem.index_dir	The directory to store the index files	<EDDS_SERVER_HOME>/filesystem_index
esa.egos.edds.edds_filesystem.indexed_root	The directory to index	<EDDS_FILESYSTEM_INDEXED_ROOT>
esa.egos.edds.provider.fs.subscription.delay	Minimum delay between subscription notifications in milliseconds. The delay between change happening in the file system and the File System Subscription processing to start. If this time is too short, the File System Subscriptions will trigger multiple File System File requests before the file is finished copying to the File System root directory. That might happen because the OS will send an update notification every second or two when you are copying a big file or when temporary files are created. Can also be viewed as: if a file hasn't been updated for x amount of time, then expect that it is complete.	10000

ENCRYPTION_AES_ENABLED	A flag to set whether the AES algorithm is enabled for this mission. Should be "true" if enabled.	true
ENCRYPTION_AES_KEY_STRENGTH	Sets the generated AES encryption key strength. Default is 128 bit. This can be increased to 192 or 256 bits, by installing Oracle's Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy from http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html	128
ENCRYPTION_RSA_PUBLIC_KEY_LOCATION	RSA public key location. The public key file should be in the X.509 encoded key specification and should just contain the body (the DER part of a PEM file)	
esa.egos.edds.edds-server.retries.maximum	Maximum number of retries when a request has failed.	0
esa.egos.edds.edds-server.retries.interval	Interval in milliseconds between request retries.	
EDDS_MISSION_XSL_DIRECTORY	The directory where the eXtensible Stylesheet Language files for this mission are stored. See Section 4.6 for more details.	

Table 4-1: EDDS Properties

4.3 Using mission specific configuration (MSC) files:

To define the directory location where all MSC files are be stored. Set the following property in the configuration file –'edds.properties'

```
# DELIVERY_MANAGER_MISSION_SPECIFIC_CONFIG_DIRECTORY: The directory of all mission specific
configuration files
esa.egos.edds.delivery_manager_mission_specific_config_directory=directory_location
```

The following properties can be set in the mission configuration file by the mission admin.

```
<!-- DELIVERY_MANAGER_MAX_ATTEMPTS - The number of max attempts the delivery manager will try to do in
case of errors in the delivery -->
  <entry key="esa.egos.edds.delivery_manager.max_attempts">1</entry>

   <!-- DELIVERY_MANAGER_INTERVAL_BETWEEN_ATTEMPTS - The time (in seconds) between
each attempt. -->
   <entry key="esa.egos.edds.delivery_manager.intervals_between_attempts">60</entry>

<!--The total time (in seconds) to wait when sending a file to multiple targets
This value should be 1 seconds greater than
(esa.egos.edds.delivery_manager.max_attempts + 1) * ((SFTP|FTP)_CONNECTION_TIME_OUT +
esa.egos.edds.delivery_manager.intervals_between_attempts)
Example: esa.egos.edds.delivery_manager.max_attempts = 2 FTP_CONNECTION_TIME_OUT = 120
esa.egos.edds.delivery_manager.intervals_between_attempts = 1
esa.egos.edds.delivery_manager.multiple_targets_total_time = (3) * (121) + (1)= 364 -->
   <entry key="esa.egos.edds.delivery_manager.multiple_targets_total_time">364</entry>

<!-- DELIVERY_MANAGER_SFTP_USAGE - Whether sftp should be used instead of ftp. -->
   <entry key="esa.egos.edds.delivery_manager.sftp_usage">false</entry>
```

```
<!-- DELIVERY_MANAGER_SFTP_PORT - The port to connect to for SFTP -->
  <entry key="esa.egos.edds.delivery_manager.sftp_port">22</entry>

<!--DELIVERY_MANAGER_MAX_FILE_RETENTION_TIME The amount of time (in days) a file is
retained on the EDDS server after local delivery. Changing this value only affects the future requests, the
already existing files are not rescheduled.  -->
  <entry key="esa.egos.edds.delivery_manager.max_file_retention_time">4</entry>
```

If any of the properties are missing in the MSC, the EDDS hard-coded defaults (shown above) are used.

4.4 Enabling or Disabling EDDS Services for a Mission

It is possible to define which services for a particular mission are available to the EDDS Server. The list of services is defined in a property called SUPPORTED_BATCH_REQUESTS and SUPPORTED_STREAM_REQUESTS within the EDDS Server properties file. This property value is replaced with the value provided in the edds-deploy.properties at deploy time.

The entry must contain all of the request types (the names of which can be found in the file common.xsd located in edds-ws-common under the type “DataAccessDataElement”) separated by commas and placed in single quotes. Leaving this entry blank will result in an EDDS Server not processing any requests.

Typical setup could be to have different EDDS Servers for different archives.

Should either of these properties be left empty, it is important to disable the batch and/or stream request route processing as appropriate. To do this, after deploying EDDS Server edit the file “eddsServerContext.xml” within the root of the deployed EDDS Server directory. If you have set SUPPORTED_BATCH_REQUESTS to be empty (nothing after the equals sign) then find the route with the ID of “batchRequestsRoute” and comment it out by placing “<!--” at the start of the route and “-->” at the end. For example:

```
        <!-- Route to respond to batch requests with RequestSubType selector -->
        <!--
        <camel:camelContext>
                <camel:route id="batchRequestsRoute">
                        <camel:from

        uri="activemq:esa.egos.edds.request.batch.tobeprocessed?selector=MISSION_NAME='{{MISSION
_NAME}}'+AND+RequestSubType+IN+({{SUPPORTED_BATCH_REQUESTS}})" />
                        <camel:log message="Batch request received" loggingLevel="DEBUG" />
                        <camel:to uri="bean:batchRequestHandler?method=handleRequest" />
                </camel:route>
        </camel:camelContext>
        -->
```

If you have set SUPPORTED_STREAM_REQUESTS to be empty (nothing after the equals sign) then find the route with the ID of “streamRequestsRoute” and comment it out by placing “<!--” at the start of the route and “-->” at the end. For example:

```
        <!-- Route to respond to stream requests with RequestSubType selector -->
        <!--
        <camel:camelContext>
                <camel:route id="streamRequestsRoute">
                        <camel:from

        uri="activemq:esa.egos.edds.request.stream.tobeprocessed?selector=MISSION_NAME='{{MISSIO
N_NAME}}'+AND+RequestSubType+IN+({{SUPPORTED_STREAM_REQUESTS}})" />
                        <camel:log message="Stream request received" loggingLevel="DEBUG" />

                        <camel:to uri="bean:streamRequestHandler?method=handleRequest" />
                </camel:route>
        </camel:camelContext>
        -->
```

Should you enter one or more batch or stream request types in this property again later, make sure you uncomment the appropriate route.

4.5 Configuring the EDDS server to retry to process failed requests

EDDS server can be configured to retry to process the requests when they have previously failed (e.g. the archive server has been down). By default EDDS server will make no retries. To configure it to do so, one must configure following properties: esa.egos.edds.edds-server.retries.maximum and esa.egos.edds.edds-server.retries.interval. While EDDS server keeps trying to process the request, the status of the request stays as ACTIVE.

4.6 XSL Transformations

EDDS can transform the XML response file into an arbitrary format using the XML transformations. Each of the request types that also provide an XML format can have one or more transformation stylesheets in the folder defined by the property *EDDS_MISSION_XSL_DIRECTORY*. The stylesheets are applied if "XML Transform" is selected as the response format in the batch request with the name of the transformation file as the parameter.

The list of available transformations can be retrieved by getTransformations() operation of the EDDS Web Service.

4.6.1 Transformation files

The *{EDDS_MISSION_XSL_DIRECTORY}* in the filesystem of EDDS Server contains all the transformations available for the EDDS Server.

The following transformation files are picked up by EDDS server as available transformations:

TypeOfRequest.xsl

TypeOfRequest.Anyname.xsl

TypeOfRequest/any_name.xsl

TypeOfRequest/anyFile

TypeOfRequest/any.extension.txt

Here *TypeOfRequest* is one of the following types:

ArchiveCatalogue

EventRecordReport

OolRecordReport

Param

ParamDefinition

ParamPreview

ParamStatistics

SmonParam

PktEvRaw

PktEvStatistics

PktTcRaw

PktTcReport

PktTcStatistics

PktTmRaw

PktTmReport

PktTmGapReport

PktTmStatistics

FileSystemFileCatalogue

FileSystemFolderCatalogue

FileSystemSubscriptionNotification

EddsUsageReport

Transforming the Parameter Request into TDRS format is provided with default EDDS installation (filename Param.xsl).

4.6.2 Email transformation

If the property *esa.egos.edds.acknowledgement.mail.transform* on EDDS Server is set to true, there must exist an AcknowledgementPart.xsl file in the {*EDDS_MISSION_XSL_DIRECTORY*} directory, that transforms the AcknowledgementPart XML into desired format. The default implementation transforms the acknowledgement into an HTML table.

4.7 SCOS 2000 MIB Required Files

EDDS requires some files from the mission it communicates with in order to perform certain functions. These files must be kept up-to-date every time the MIB the mission is using changes. Failure to update the files will result in the inability to perform filters such as an APID search for TM packets. EDDS requires files from the mission described in the next subsections. To enable EDDS Server to find these files, create a directory in a location that the EDDS Server can read (e.g. /home/edds/EDDS_HOME/MIB) and specify this location in the EDDS Server properties file (MIB_FILES_LOCATION). Within this directory, create a new directory for each domain, using the name you have specified when configuring the mission within the EDDS MMI (e.g. “0”, “1”, “2” and so on). Within these domain directories, place the files from the mission described in the next sections.

4.7.1 Process ID File

The process ID file (pid.dat) can typically be found in the data/ASCII directories for each domain. Place the pid.dat file from each domain of the mission into the correct domain directory in the location you specified in Section 4.7. This file is needed to convert SPIDs into APIDs and so on.

4.7.2 Telemetry Packets Definition File

The telemetry packets definition file (tpcf.dat) can typically be found in the data/ASCII directories for each domain. Place the tpcf.dat file from each domain of the mission into the correct domain directory in the location you specified in Section 4.7. This file is needed to get the description of TM packets.

4.7.3 Command Subsystem Mapping File

The Command Subsystem Mapping file (CMD_SUBSYSTEM_FILE) can typically be found in the CMD directories for each domain. Place the CMD_SUBSYSTEM_FILE file from each domain of the mission into the correct domain directory in the location you specified in Section 4.7. This file is needed to map the subsystem ID stored in the packet with the subsystem name for inclusion in the TC report.

4.7.4 Command Host File

The Command Host file (CMD_HOST_FILE) can typically be found in the CMD directories for each domain. Place the CMD_HOST_FILE file from each domain of the mission into the correct domain directory in the location you specified in Section 4.7. This file is needed to map the source ID stored in the packet with the hostname for inclusion in the TC report.

4.7.5 MISCbase.ids File

The MISCbase variables file (MISCbase.ids) can typically be found in the config directories for each domain. Place the MISCbase.ids file from each domain of the mission into the correct domain directory in the location you specified in Section 4.7. This file is needed to map the domain ID stored in the packet with the domain name for inclusion in the TC report.

4.7.6 MISCcontext_SHARED.sta File

The MISCcontext_SHARED.sta file can typically be found in the config directories for each domain. Place the MISCcontext_SHARED.sta file from each domain of the mission into the correct domain directory in the location you specified in Section 4.7. This file is needed to map the domain number with the domain name and data partition name with the data stream ID. Only needed should the data provision services be used.

4.7.7 CMDsourceTypeMapping.properties

The CMDsourceTypeMapping.properties file must be manually created. An example is provided in the EDDS installation. This file is the mapping between the command source type number found in the TC packet and its description for the Packet TC Report.

The contents of the file is based on the enumeration value CMDsourceType that can be found in CMDmiscUtil.idl in model-commanding-rep/src/CMD/

Here is an example of the enumeration:

```
enum CMDsourceType {MANUAL_STACK,
AUTO_STACK,
EXT_SOURCE,
TC_SPACON,
OBQM_DISP,
SMF_SOURCE};
```

The format of the file CMDsourceTypeMapping.properties is:

```
0=MANUAL_STACK,MS
1=AUTO_STACK,AS
2=EXT_SOURCE,ES
3=TC_SPACON,TS
4=OBQM_DISP,OD
5=SMF_SOURCE,SMF
```

The number at the start is the source type number from the packet, starting from zero. The first value after the equals is the long description included in the XML report. The value after the comma is the short value as shown in the ASCII report. It is essential that the file exists and contains a mapping for all command source types expected, otherwise warnings will be generated and the output will contain the source ID as a number as present in the packet instead of the text description.

4.7.8 CMD_ttRqstObqStatusMapping.properties

The file CMD_ttRqstObqStatusMapping.properties must be manually edited. A basic version for vanilla scos is provided in the EDDS installation. This file is the mapping between the command request status type number found in the TC packet and its description for the Packet TC on EDDS.

The content of the file is based on the enumeration value CMDttRqstObqStatus provided by the data provision service through the backend service api.

Here is an example of the enumeration:

```
enum CMDttRqstObqStatus {TT_EXECUTED,
TT_LOADED,
TT_DISABLED,
TT_ENABLED,
TT_DELETED,
TT_REMOVED,
TT_PENDING_LOAD,
TT_PENDING_DISABLE,
TT_PENDING_ENABLE,
TT_PENDING_DELETE,
TT_ASSUMED_LOADED,
TT_ASSUMED_DISABLED,
TT_ASSUMED_ENABLED,
TT_ASSUMED_DELETED,
TT_FAILED_LOAD,
TT_UNKNOWN};
```

The format of the file CMD_ttRqstObqStatusMapping.properties is:

```
0=TT_EXECUTED,EXECUTED
1=TT_LOADED,LOADED
2=TT_DISABLED,DISABLED
3=TT_ENABLED,ENABLED
4=TT_DELETED,DELETED
5=TT_REMOVED,REMOVED
6=TT_PENDING_LOAD,PEND LOAD
7=TT_PENDING_DISABLE,PEND DISABLE
8=TT_PENDING_ENABLE,PEND ENABLE
9=TT_PENDING_DELETE,PEND DELETE
10=TT_ASSUMED_LOADED,ASSUMED LOADED
11=TT_ASSUMED_DISABLED,ASSUMED DISABLED
12=TT_ASSUMED_ENABLED,ASSUMED ENABLED
13=TT_ASSUMED_DELETED,ASSUMED DELETED
14=TT_FAILED_LOAD,NO LOAD
15=TT_UNKNOWN,UNKNOWN
```

The number at the start is the request status type number from the packet, starting from zero. The first value after the equals is the long description included in the XML report. The value after the comma is the text value. It is essential that the file exists under the folder <edds server>/config/MIB/0 and contains a mapping for all request status types expected, otherwise errors will be generated and EDDS may fail to decode TC packets.

5. Setting up the development environment

The development environment assumes that all the steps in section **3.1** and 3.2 have already been taken and the EDDS source code is available as described.

In order to install the development environment and check out the source code, the following additional steps are needed.

Note: It is highly recommended that you use a new and clean Eclipse workspace, especially if you already have an EDDS development environment that doesn't use Maven. Whenever you experience build problems you can try a clean build and/or right click on a project > Maven > Update Project... and select all projects and OK to update all Maven projects. Furthermore, if Eclipse refuses to download required libraries from the specified Maven repository, try to run Maven from command line as described in Section 3.2.9.

5.1 Installing the required software

Ensure you have jdk11.0.2+9 installed.
Download the RCP distribution of Eclipse Luna v4.4 (eclipse-rcp-luna-linux-gtk.tar.gz).
Install Ant 1.9.10.
Install Maven 3.5.4

5.2 Configuring Ant

1. Copy the catalina-ant.jar file from your Apache Tomcat lib directory to your Ant lib directory (e.g. ~/opt/apache/apache-tomcat/apache-tomcat-9.0.100/lib/catalina-ant.jar to ~/opt/apache/apache-ant/apache-ant-1.9.10/lib)
2. Within Eclipse, click Window and Preferences
3. Expand Ant then click Runtime
4. Select the Classpath tab and click Ant Home
5. Navigate to the location where you have Ant installed (e.g. ~/opt/apache/apache-ant/apache-ant-1.9.10)
6. Click OK to all the open windows.

Note: If you are having problems running Ant, then it might help to create an alias for the Ant command using the following command:

```
alias ant="ant --noconfig"
```

The EDDS Ant target “deploy-environment-settings” (see Section 3.3.1) will add this alias to your log-in script. If you still have problems running Ant (error message about not being able to find the Launcher Jar file) then, as root, execute the following commands:

```
mkdir /usr/share/jdk11
mkdir /usr/lib64/jdk11
```

5.3 Configuring Eclipse

Additional libraries for Eclipse need to be downloaded and the easiest way is to do it through its update discovery site. If you are behind a firewall you may need to set up your proxy connections first:

- Click Window and Preferences
- Enter "Network Connections" in the box
- Set Active Provider to Manual

- Change the Proxy entries for HTTP and HTTPS

Maven also requires access to the Internet or Intranet to download required libraries from the repository. Therefore, you might need to set up your proxy connection for Maven as well:

- Click Window and Preferences
- Go to Maven > User Settings
- Open the specified user settings file (should be `/home/user/.m2/settings.xml`):

```
<settings>
  <proxies>
    <proxy>
      <active>true</active>
      <protocol>http</protocol>
      <host>YOUR_PROXY_IP</host>
      <port>YOUR_PROXY_PORT</port>
      <nonProxyHosts>localhost|127.0.0.1 </nonProxyHosts>
    </proxy>
  </proxies>
</settings>
```

- Modify these settings with your proxy IP and port and save the file
- Click Update Settings in the User Settings Preferences to apply the changes

The version of Eclipse is Luna v4.4. We downloaded the RCP distribution (eclipse-rcp-luna-linux-gtk.tar.gz) which is missing the WTP part for the web development. Once you have correctly installed Eclipse check the following:

1. under Preferences > Java > Installed JREs add a new Standard VM pointing to your jdk1.8.0_25 and check this installation as the default one;
2. under Help > Install Updates
   1) select: Galileo - http://download.eclipse.org/releases/galileo
   2) expand Web, XML and Java EE Development and select the following:
      i. Eclipse Faceted Framework
      ii. Eclipse Faceted Framework JDT Enablement
      iii. Eclipse Java EE Developer Tools
      iv. Eclipse Web Developer Tools
      v. Eclipse XML Editors and Tools
      vi. Eclipse XSL Developer Tools
      vii. Eclipse Web Page Editor (Optional)
      viii. Eclipse WST server Adapters
   3) Install additional Maven plugins
      i. Maven Integration for Eclipse (Location: http://download.eclipse.org/technology/m2e/releases)
      ii. m2e connector for build-helper-maven-plugin (Location: http://repo1.maven.org/maven2/.m2e/connectors/m2eclipse-buildhelper/0.15.0/N/0.15.0.201207090124/)
      iii. m2e connector for jaxws-maven-plugin (Location: http://coderplus.com/m2e-update-sites/jaxws-maven-plugin/)
      iv. Tycho Project Configuration (Location: http://repo1.maven.org/maven2/.m2e/connectors/m2eclipse-tycho/0.8.0/N/0.8.0.201409231215/)
   4) For drag-and-drop support for SWT you can install WindowBuilder Pro
      i. Press Add...' button
         1. Name: WindowBuilder Pro
         2. Location: http://dl.google.com/eclipse/inst/d2wbpro/latest/3.6http://dl.google.com/eclipse/inst/d2wbpro/latest/3.6
      ii. Select SWT Designer
      iii. Press Next > button
      iv. Press Next > button
      v. Accept the term of the licence agreement and press Finish

3. go ahead and restart Eclipse

5.4 Now read JAutodoc templates

Optionally you can use JAutodoc for automatically adding JavaDoc and file headers to your source code. Read more from [AD-4].

Suggested templates are In Appendix: **JAutodoc**.

5.5 Checking out the source code

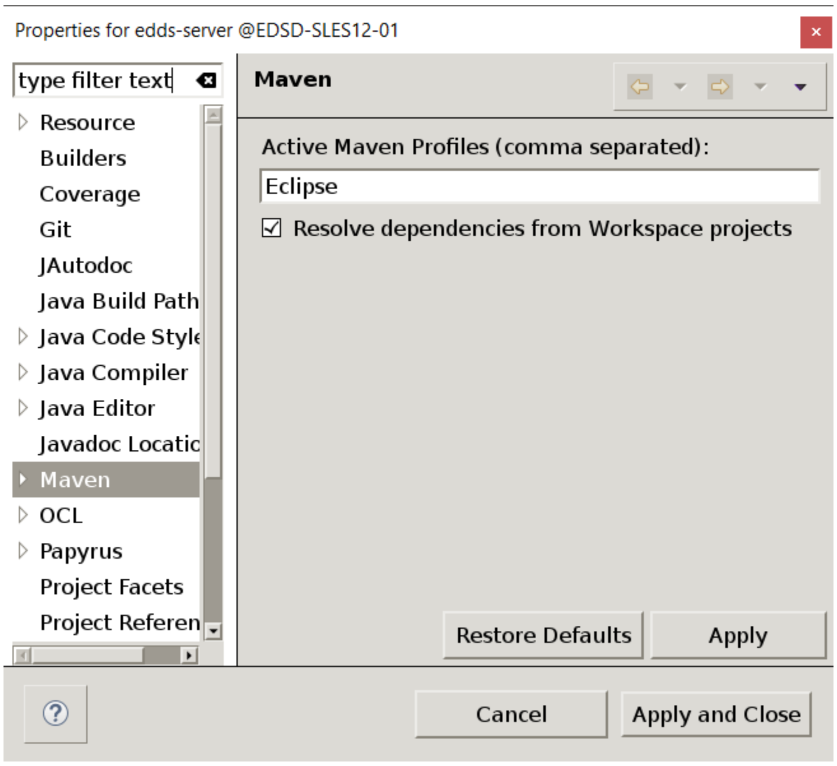
Follow the steps for checking out the source code from the Git repository in Section 3.2.2.
Next, you'll need to import each component of EDDS as a project to the workspace:

- Click File and Import...
- Expand Maven and select Existing Maven Projects
- Click Browse next to "Select root directory" and navigate to the root of your checked out source folder and select it
- Select all projects in the list and click Finish

Once the projects are imported, the maven profile called **Eclipse** must be enabled for the following projects in the maven plugin:

- edds-archiver
- edds-assembly-descriptor
- edds-common
- edds-configuration
- edds-dar
- edds-db
- edds-delivery
- edds-farc-2.0-interface
- edds-farc-interface
- edds-filesystem
- edds-formatters
- edds-javaws
- edds-javaws
- edds-migrator
- edds-parc
- edds-performance-processor
- edds-provider-base
- edds-request-submitter
- edds-server
- edds-smon
- edds-stream-client
- edds-ws-client
- edds-ws-common
- edds-ws-server
- esa.egos.edds.mmi-build
- esa.egos.edds.mmi/rap-build/rap-war

by clicking on **Project->Properties->Maven** then inserting **Eclipse** on Maven Active Profiles field.



Maven

Active Maven Profiles (comma separated):

Eclipse

☑ Resolve dependencies from Workspace projects

Figure 2 Checking out the source code

Once all projects are imported and built, the Eclipse Target Platform needs to be updated:

- Navigate to the project *esa.egos.edds.target* in the Eclipse Package Explorer
- Open the *esa.egos.edds.target.target* file
- In the top right, click on “Set as Target Platform”
- Note: It might be necessary to provide credentials to access the repository

5.6 Formatting

- Copy the XML from **K.1** and save it to a file on disk (the same computer as where Eclipse is installed)
- Click Window and Preferences
- Expand, Java, Editor, Save Actions
- Tick "Perform the selected actions on save" and "Format source code" and "Format all lines" and "Organise imports"
- Click the link "Formatter"
- Click Import and select the file you saved in the first step
- Click OK

5.7 Code Templates

- Copy the XML from **Codetemplate.xml** and save it to a file on disk (the same computer as where Eclipse is installed)
- Click Window and Preferences
- Expand Java, Code Style, Code Templates
- Click Import and select the file you saved in the first step
- Click OK

5.8 *Running MMI from development environment*

5.8.1 Setting up Run configuration

Right-click on eud.product in esa.egos.edds.mmi-build and "Run as->Eclipse Application"
Or follow the next instructions to create the configuration manually

5.8.1.1 Create a new run configuration manually

1. Open Run/Run Configuration...
2. Double-click on Eclipse Application
    a. Change Name to EDDS
    b. Program to Run
        i. Change Run a product to esa.egos.edds.product.product
    c. Change to tab Arguments
        i. Enter into VM arguments (You may need to modify the values)

```
-DEDDS_HOME=/lhome1/edds/EDDS_HOME
-Dedds.server.endpoint.url=http://10.48.29.158:8080/edds/EddsService?wsdl
-Dmmi.path.batchRequests=BATCH_REQUESTS
-Dstatus_view_start_time_offset=-P7D
```

    d. Change to tab Plug-ins
        i. Change Launch with: to plug-ins selected below only
        ii. Press Deselect All button
        iii. Select the 4 esa.egos.* plug-ins
        iv. Press Add Required Plug-ins
        v. Press Apply button
3. Press Close button

5.9 *Issuing an EDDS release*

This expects that you have set up your development environment as described in section 3.2

5.9.1 Release preparation

- Commit changed documents to the repository
- Tag the repository

5.9.2 Creating the CDs

1. Check out the version of source code you want to do a release for (a tagged revision) git checkout TAG
2. In {EDDS_Source_Dir}/edds

    mvn clean install javadoc:aggregate-jar -Dadditionalparam=-Xdoclint:none *-DskipTest=true*

    cd edds-release

    ant release -Drelease.dir=path/to/release/directory

    e.g.

    ant release -Drelease.dir=/lhome1/edds/EDDS_RELEASE

This will:

1) Create the documentation CD with generated JavaDoc

2) Create the Source Code CD with the current source code (from repository)

Appendix A Packet Data

This section provides instructions for the creation of the XML files describing the structure of a packet file. This file will be used by EDDS during the process of decoding of the file.

Detailed information about the fields can be found in the EUICD document, under the section “Format/Raw/Packet Data”.

A.1 Creating the header structure information files for TM

To create the XML file that defines the structure of the TM raw header, create a new XML file based on the RawPktHeaders.xsd file. This can be found within edds-server/src/main/resources in the source code repository. Create a root node of “RawTmPktHeader” followed by one or more Field elements. The attribute “name” of the Field element should define the name of the header field to include in the raw file, and the Length element within the element Field should define the size of the field. See the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<p:RawTmPktHeader
      xmlns:p="http://edds.egos.esa/provider/packet/data/pktheaders"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://edds.egos.esa/provider/packet/data/pktheade
rs /lhome1/edds/workspaces/edds/edds/edds-
server/src/main/resources/RawPktHeaders.xsd">
      <Field name="Apid">
            <Length>2</Length>
      </Field>
      <Field name="Type">
            <Length>2</Length>
      </Field>
</p:RawTmPktHeader>
```

A.2 Creating the header structure information files for TC

To create the XML file that defines the structure of the TC raw header, create a new XML file based on the RawPktHeaders.xsd file. This can be found within edds-server/src/main/resources in the source code. Create a root node of “RawTcPktHeader” followed by one or more Field elements. The attribute “name” of the Field element should define the name of the header field to include in the raw file, and the Length element within the element Field should define the size of the field. See the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<p:RawTcPktHeader
      xmlns:p="http://edds.egos.esa/provider/packet/data/pktheaders"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://edds.egos.esa/provider/packet/data/pktheade
rs /lhome1/edds/workspaces/edds/edds/edds-
server/src/main/resources/RawPktHeaders.xsd">
      <Field name="ServiceType">
            <Length>2</Length>
      </Field>
      <Field name="ServiceSubType">
            <Length>2</Length>
      </Field>
</p:RawTcPktHeader>
```

A.3 Creating the header structure information files for DataProvision TC

To create the XML file that defines the structure of the Data Provision custom fields, create a new XML file based on the DataProvisionCustomField.xsd file. This can be found within edds-parc/src/main/resources/esa/egos/edds. Create a root node of “CustomFields” followed by one or more CustomField elements. See the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<p: CustomFields
      xmlns:p="http://edds.egos.esa/provider/packet/dataprovision/customfiel
ds"
      xmlns:p1= "http://edds.egos.esa/provider/packet/data/pkthelper"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://edds.egos.esa/provider/packet/dataprovision
/customvields /lhome1/edds/workspaces/edds/edds/edds-
parc/src/main/resources/esa/egos/edds/DataProvisionCustomField.xsd">
      <CustomField FieldPath="command.record.releaseInf.subsystemName"
FieldName="Subsystem Name">
            <Type>String</String>
      </CustomField>
      <CustomField FieldPath="command.record.releaseInf.domain"
FieldName="Domain">
            <Type>String</String>
      </CustomField>

</p: CustomFields>
```

A.4 Defining the TC packet structure

Due to a limitation with the PARC, it is not possible to retrieve the required information decoded for each mission. Instead, this information is received by EDDS over CORBA from the PARC in a Hex encoded format and must be decoded by EDDS. The TC packet body contains a lot of information such as the command name, the sequence name, and all of the parameter data. The structure of the TC Packet data can vary depending on the mission, and as such the EDDS needs to know the structure of the packet so that it can decode the Standard SCOS data. The packet structure is defined by an XML file, and the schema for this XML file can be found in edds-parc/src/main/resources/TCPacketStructure.xsd. The schema will be described below:

The root element is “PacketStructure”. This element consists of one or more “Field” elements, one or more “VerStagePositions” elements and zero or more “CustomField” elements. The “Field” element contains the following elements:

- Offset – the offset from the start of the body data from the PARC to the name of the command. The offset should be the position in the returned Hex string where the command name starts.
- Length – the length of the field in the hex string. Note that two Hexadecimal numbers are needed to represent one byte.
- Type – the type of the data, for example String, UnsignedLong, Long, UnsignedInteger, Integer, UnsignedShort, Short or Byte as defined in ValidTypes.xsd
- Occurrence – how many times this field occurs in the packet. For all fields except “verStates”, this should be set to 1. For the “verStates” field, this should have the value of the constant “CMD_NUM_VER_STAGES” within CMDdataDetails.idl in model-commanding-rep of the SCOS mission source code.

The “VerStagePositions” field maps each verification stage to a known stage. The element “pos” indicates the number within the verStates array that this stage maps to. The attribute “stage” maps the stage to one of a number of stages. For example, the first stage is usually the “cmdRelPos” (command release stage) and this would be shown in the XML file as follows:

```
<VerStagePositions stage=”cmdRelPos”>
```

```
	<pos>0<pos>
</VerStagePositions>
```

The optional "CustomField" element allows for mission specific changes to the TC packet to be decoded and included in the TC Report XML and binary output. These are changes that have been made to the IDL "`model-commanding-rep/src/CMD/CMDdataDetails.idl`" on S2K 5.x and "`api-commanding/CMDI/src/main/idl/CMD/CMDdataDetails.idl`" for S2k 6.x, specific for a mission. EDDS can decode basic types and in addition a simple enumeration mapping can be provided or a mask can be provided to allow for more human readable output. Note that the FieldName value must be unique!

In addition, configure the TC_PKT_CMD_LEN_PAR_INFO parameter within the edds.properties file that indicates the length in bytes of the INFO field. This value is 3 for S2K 6.2.0+ and 2 for all previous versions. Adjust this parameter based on the version of S2K you are using to avoid errors during decoding.

For the simple case, here is some sample XML that defines (in this case a standard SCOS provided field) a field called "Command Description" which has an offset of 18 characters within the hexadecimal packet string provided by the PARC and a length of 50 characters. Note that two hex characters represents one byte.

```
<CustomField FieldName="Command Description">
	<Offset>18</Offset>
	<Length>50</Length>
	<Type>String</Type>
</CustomField>
```

The following example defines a Short type called "Source ID" which has an offset of 172 characters within the hexadecimal packet string provided by the PARC and a length of 4 characters.

```
<CustomField FieldName="Source ID">
	<Offset>172</Offset>
	<Length>4</Length>
	<Type>UnsignedShort</Type>
</CustomField>
```

The following example defines a field that applies a mask to the acknowledgement flags. The Mask represents a single byte represented as a hexadecimal character pair. Should the mask match, even partially, the value in the packet, the value in "Value" is provided in the output. If more than one of the specified masks matches the value in the packet, the values are included in the output as a comma separated string. For Mask types, the type specified in the output is always "String" as the String value is used in the output.

In the example below, if the binary value in the packet is `00000001` then the first mask would match – "`01`" as hex is "`00000001`" in binary and when the value compared to the mask using the bitwise AND operator, the result is true, so "accept" is included in the output.

If the binary value in the packet is `00000101` then the first mask would match – "`01`" as hex is "`00000001`" in binary and when the value compared to the mask using the bitwise AND operator, the result is true. However, the second mask would also match – "`04`" as hex is "`00000100`" in binary and when the value compared to the mask using the bitwise AND operator, the result is true, so "accept, progress" is included in the output.

```
<CustomField FieldName="Acknowledgement flags">
        <Offset>250</Offset>
        <Length>2</Length>
        <Mask Mask="01" Value="accept" />
        <Mask Mask="04" Value="progress" />
        <Mask Mask="08" Value="exec complete" />
</CustomField>
```

The following example defines an enumeration type. This allows for a mapping between the byte value in the packet to a String representation. If the value in the packet doesn't match any of the provided enumeration mappings, then the byte value is given in the output.

In the example below, if the value in the packet is "0", then "AD" is provided as the value in the output.

```
<CustomField FieldName="Uplink Mode">
        <Offset>238</Offset>
        <Length>2</Length>
        <Enum Name="AD" Value="0" />
        <Enum Name="BD" Value="1" />
</CustomField>
```

A sample file for mission specific configuration is provided in the EDDS installation within the EDDS Server installation folder under "`config/MIB/SampleMissionSpecificConfig.xml`".

To help with finding the offset of the Command Name from a packet within the PARC, a helper program is available. This program will take a packet from the PARC and look for the command name within the encoded packet and return the offset. It will also attempt to decode the packet using the XML packet structure file provided. To run this program, go to where the EDDS Server has been deployed and run the following command and follow the on-screen instructions:

```
ant –f PktTcStructureHelper.xml
```

You can pass the program a file containing all the required information, to save having to type it in each time at each prompt. Create a new text file with the following content (change the values as appropriate). Dates can be entered in the DOY format (2010-130T08:00:00.000) or DMY format (2010-05-11T08:00:00.000). The value for "firstparamname" can be omitted if there are no parameters for the command.

```
domain=0
starttime=2011-304T10:36:07.054
endtime=2011-304T10:36:08.000
commandname=S2KTC010
firstparamname=S2KCP013
dataspace=PRIME
packetstring=00000100c6015e5537380d00c6015e5537380d00dc6d00008f020500b702000
02530010...
packetxmlfile=/tmp/packetxmlfile.xml
```

The values `packetstring` and `packetxmlfile` are optional. If they are specified, then it is not necessary to specify `starttime` or `endtime`.

The `dataspace` is also optional. If not specified, the current dataspace will be used.

The text file is the same format as any Java properties file. Next start the program with the following command:

```
ant -f PktTcStructureHelper.xml -Drequest=<full path to file>
```

A.5 Defining DataProvision TC packet structure

The attribute "FieldPath" of the CustomField element should define the path to reach the desired field. The path is composed by the variable names that define the objects to cross in the path to reach the desired field, comma separated. To formulate the path you can use as reference the "CHIScommandRecord.idl" file. A second attribute is necessary named "FieldName" that defines the name of the field.

The "Type" field must be a valid type according to the "ValidTypes.xsd" file that can be found within edds-parc/src/main/resources/esa/egos/edds.

A.6 Creating the header structure information files for EV

To create the XML file that defines the structure of the EV raw header, create a new XML file based on the RawPktHeaders.xsd file. This can be found within edds-server/src/main/resources in the source code repository. Create a root node of "RawEvPktHeader" followed by one or more Field elements. The attribute "name" of the Field element should define the name of the header field to include in the raw file, and the Length element within the element Field should define the size of the field. See the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<p:RawEvPktHeader
      xmlns:p="http://edds.egos.esa/provider/packet/data/pktheaders"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://edds.egos.esa/provider/packet/data/pktheade
rs /lhome1/edds/workspaces/edds/edds/edds-
server/src/main/resources/RawPktHeaders.xsd">
      <Field name="Category">
            <Length>20</Length>
      </Field>
</p:RawEvPktHeader>
```

Appendix B Configuring SMON Server for use with EDDS

EDDS can retrieve parameter samples via the SMON Server process within SCOS instead of from the DARC. The SMON Server in SCOS can only handle one connection at a time. EDDS is able to handle multiple requests simultaneously. As a result, it is necessary to configure a separate SMON Server for each parallel request EDDS supports. To do this, follow these steps:

1) Open a command shell to the location where the SCOS prime server is installed
2) Navigate to the admin directory or 0/admin directory for multi-mission installations.
3) Open TKMA_master.txt. This points to the actual TKMA file in use for each domain. Open this file for editing or for multi-mission installations, each file in turn.
4) Search for the text “SMON instances”. Copy one of the lines to create a template that can be changed. Change the first part to read “SMON_EDDS_1”, after the first tab, ensure there is a “Y” in the column, at the end of the line, change the part after “--instanceName" to read “SMON EDDS 1”. The line will look like (→ represents a tab character):

   ```
   SMON_EDDS_1→Y→1→$scosii_basedir/$admin_global_dir/RunTask→SMON_S
   ERVER→0→$SCOSOPS→$SCOSGRP ->40→0→SMON_SERVER --instanceName SMON
   EDDS 1
   ```

5) Save and close the file. Repeat the previous step for multi-mission domains for the remaining domains.
6) Next open MMI_master.txt file. This points to the actual MMI file in use for each domain. Open this file for editing or for multi-mission installations, each file in turn.
7) Go to the end of the file. Create a new screen for EDDS. To do this, enter the following line, ensuring you separate each entry with a TAB character:

```
NewScreen→EDDS→1→PRIME→1→PRIME
```

8) Enter “Newline” on its own, then add the following line:

```
Command→EDDS SMON
Server→Y→100→50→DUMMY/resources/pbm/SMON.pbm→→→→SMON_EDDS_1→Y→Y
→N→1→→EDDS 1
```

9) Save and close the file. Repeat the previous step for multi-mission domains for the remaining domains.
10) Restart SCOS

To specify the datastreams to use, change the line in step 4 and add the “--datastreams” parameter to the end. This accepts an underscore separated list of datastreams e.g. for datastreams 1 & 2 (with a space after the --datastreams parameter);

```
SMON_EDDS_1→Y→1→$scosii_basedir/$admin_global_dir/RunTask→SMON_SERVER→0→
$SCOSOPS→$SCOSGRP 40→0→SMON_SERVER --datastreams 1_2 --instanceName SMON
EDDS 1
```

Optional: Add to the /etc/hosts the IP address and the correct name of the machine where EDDS server is located. This seems to solve some problems related to SMON connection.

Appendix C Pure-FTP Installation

The Pure-FTP binary that is supplied with SLES 11 is not compiled with the `--with-virtualchroot` option. This is needed by EDDS to enable FTP users to use the symbolic links in their home directory to the public, mission and role shared directories. To build a Pure-FTP binary with this option enabled, follow these instructions:

C.1 Prerequisites

You will need to have the `openldap2-devel` package installed. To install this package, log-in to the server as root and enter “yast”. Select “Software” then select “Software Management”. In the search field, enter “openldap” then select the package”openldap2-devel”. Select “Accept” and then quit YaST2 once the installation is complete. You may need your SLES 11 installation disks. Other development packages may also be needed, depending on how your server is installed. The configure script below will inform you of what libraries are missing.

If you cannot see the `openldap2-devel` package, it could be that your configured software repositories is missing the SDK source. Once the installation below has been performed, the built binaries can be used on other servers (as long as the architectures are the same). This will avoid having to have development packages on the server where Pure-FTP is to be used.

C.2 Installation

1) Stop Pure-FTP with “`/etc/init.d/pure-ftpd stop`” as root.
2) Download Pure-FTP from http://download.pureftpd.org/pub/pure-ftpd/releases/ Select the latest “tar.gz” version of the source
3) Untar the downloaded file with the command: “`tar zxvf pure-ftpd-1.0.34.tar.gz`” (Change the filename to match the downloaded file.
4) Enter the directory created above with the command “`cd pure-ftpd-1.0.34`”
5) Enter the command “`./configure --with-ldap --prefix=/usr --with-virtualchroot --with-tls --without-humor`” (Add any other configuration options that may be needed by other users. Enter “`./configure --help`" for more information).
6) Enter “`make`”
7) Log in as root with “`su`” then enter “`make install`”
8) Start Pure-FTPd with “`/etc/init.d/pure-ftpd start`”.

Refer to Section 3.8 for configuration instructions.

Appendix D Apache ActiveMQ Configuration

D.1 Configuring ActiveMQ

After downloading ActiveMQ extract the tar file onto the server. The following steps have to be performed manually, otherwise no scheduled requests will work and the priority of requests will not be taken into account.

1) Go into the base directory of the ActiveMQ installation
2) Edit conf/activemq.xml
3) Find the following line:

```
<broker xmlns="http://activemq.apache.org/schema/core"
brokerName="localhost" dataDirectory="${activemq.data}">
```

4) Add *schedulerSupport="true"* to the end so that the line looks like this:

```
<broker xmlns="http://activemq.apache.org/schema/core"
brokerName="localhost" dataDirectory="${activemq.data}"
schedulerSupport="true">
```

5) Save the file and change to the "bin" directory
6) Start ActiveMQ with the command: `./activemq start`

D.2 Enabling password protection on ActiveMQ

ActiveMQ can be configured to require a password on access. The password can be stored in the ActiveMQ properties file in an encrypted way.

1. Go to the base directory of the ActiveMQ installation

2. Encrypt the password:
$ bin/activemq encrypt --password activemq --input mypassword
...
Encrypted text: eeWjNyX6FY8Fjp3E+F6qTytV11bZItDp

The "password" here is used to encrypt the actual password. It is not recommended to use "activemq" as it is too easy to guess. The actual password in its encrypted form needs to be saved in a property file. The password to perform the decryption needs to be passed to ActiveMQ in some way so that it can decrypt the actual password. Here, we pass the decryption password as an environment variable which we later unset.

3. Add the encrypted password to the appropriate configuration file, e.g. $ACTIVEMQ_HOME/conf/credentials-enc.properties ensuring it is added in-between the brackets after "ENC":

   activemq.username=system
   activemq.password=ENC(mYRkg+4Q4hua1kvpCCI2hg==)

4. Edit the default security properties file $ACTIVEMQ_HOME/conf/activemq-security.xml to match the configurations (i.e. ensure only the system user is present, remove unnecessary privileges).

5. In the base directory set the decryption password environment variable (not the actual password which in this example would be "mypassword" above):
$ export ACTIVEMQ_ENCRYPTION_PASSWORD=activemq

6. Start the broker :
   ```
   $ bin/activemq start xbean:conf/activemq-security.xml
   ```

7. Once started, unset the environment variable:
   ```
   $ unset ACTIVEMQ_ENCRYPTION_PASSWORD
   ```

Finally, apply the changes made to the configuration in Appendix D.1 to the activemq-security.xml configuration. You can then remove the activemq.xml configuration file and rename activemq-security.xml to activemq.xml. This saves having to specify the configuration file to user when starting ActiveMQ. The destinationPolicy may be missing altogether in the activemq-security.xml file. In this case, you'll need to copy the whole section over from the activemq.xml file modified in Appendix D.1 to the correct place.

It is recommended to bind to a specific IP address by changing the URI from tcp://0.0.0.0:61616 to, for example, tcp://10.48.29.219:61616. When doing this, ensure that the EDDS properties files are configured to use this specific IP address/hostname instead of "localhost". It is also possible to have a network of brokers for redundancy and scalability, but this is out of the scope of this document. More details can be found here: http://activemq.apache.org/configuring-transports.html

For the web console to continue to work, it is necessary to change the Spring Context file for the web console by performing the following steps:

1. Edit the file $ACTIVEMQ_HOME/webapps/admin/WEB-INF/webconsole-embedded.xml

2. Remove these lines:

```
  <!-- Allows us to use system properties as variables in this
configuration file -->
  <bean
class="org.springframework.beans.factory.config.PropertyPlaceholder
Configurer">
      <property name="locations">

<value>file:${activemq.conf}/credentials.properties</value>
      </property>
  </bean>
```

3. Replace with these lines:

```
<!-- Allows us to use encrypted system properties as variables in
this configuration file -->
  <bean id="environmentVariablesConfiguration"
class="org.jasypt.encryption.pbe.config.EnvironmentStringPBEConfig"
>
    <property name="algorithm" value="PBEWithMD5AndDES" />
    <property name="passwordEnvName"
value="ACTIVEMQ_ENCRYPTION_PASSWORD" />
  </bean>

  <bean id="configurationEncryptor"
class="org.jasypt.encryption.pbe.StandardPBEStringEncryptor">
    <property name="config" ref="environmentVariablesConfiguration"
/>
  </bean>
```

```
  <bean id="propertyConfigurer"
class="org.jasypt.spring.properties.EncryptablePropertyPlaceholderC
onfigurer">
      <constructor-arg ref="configurationEncryptor" />
      <property name="location"
value="file:${activemq.conf}/credentials-enc.properties"/>
  </bean>
```

D.3 Configuring EDDS to use ActiveMQ password

In edds-deployment.properties set the *edds.activemq.broker.username* property to, in this case, “system” and *edds.activemq.broker.password* to the actual password (in this example “mypassword”).

D.4 Debugging ActiveMQ database contents and connections

In case the web console is not available to monitor the message counts for queues and topics and to see current active connections for debugging purposes, it is still possible to enable command-line access to the same information.

To enable the JMX access to the broker, add the attribute useJmx=”true” to the broker configuration:

```
...
<broker xmlns="http://activemq.apache.org/schema/core"
useJmx="true" brokerName="gimus212"
dataDirectory="${activemq.data}" schedulerSupport="true">
...
```

After the restart of ActiveMQ, it will be able to access the data through a JMX console or command line interface.

```
apache-activemq/bin> ./activemq -help
apache-activemq/bin> ./activemq query
```

For more information on JMX configuration see ActiveMQ documentation http://activemq.apache.org/jmx.html .

D.5 Important note on clock synchronisation

When the ActiveMQ message broker and one or more of the EDDS Server components (i.e. Web Server, EDDS Server, EDDS Archiver, Delivery Manager) are running on different machines, it is essential to ensure that the clocks of all the servers are kept in sync. If this does not happen, messages may not be delivered affecting EDDS functionality.

Example: The Delivery Manager is running on Server A and the message broker on Server B. The time on Server A is 5 minutes slower than the time on Server B. The Delivery Manager requests the status of a request using a request/reply message pattern. These messages are set to expire after the timeout has elapsed. This is configurable (see Section 3.2.6, property *edds.egos.edds.jms.receive.timeout*), the default is 5 seconds. The message is received by the message broker, which notices that the message has already expired, and immediately discards it. The message is never received by the EDDS Archiver, so it cannot respond to the request. The Delivery Manager finally times out.

More information on the TimeStampPlugin, which can be used to configure the message broker in cases where clocks cannot be synchronised, can be found here: http://activemq.apache.org/timestampplugin.html

Appendix E FARC Subscriptions

The FARC Subscription requests work using notifications sent over an ActiveMQ message bus. This feature is available in FARC 2.1.9 or later. The FARC must be configured to send messages to an ActiveMQ broker when a file is committed or deleted. This can be configured in the file "~/FARC/xml/config/server.config". The FARC will then send messages to the specified topic, and any 3rd party application can then listen to messages on that topic to be notified when a change is made. Each listening application will receive its own copy of the message.

After the FARC Server has been configured, configure EDDS to listen on the correct ActiveMQ broker and topic as specified in Section 3.2.6. EDDS uses a durable connection to the FARC's topic. This means that it will receive any messages it missed while it was shutdown since it was last listening for messages on the topic. Note that for this to work, the EDDS Server must listen to the topic at least once, otherwise the ActiveMQ broker won't know how many missed messages it needs to deliver.

If more than one EDDS Server is running for the same mission, then the "farc.jms.enabled" property within the EDDS Server properties file must be set to "false" so that only one instance will listen on this topic. If this is not done, each EDDS Server instance will receive the notification and submit request(s) for the file or send an e-mail as required for each active FARC subscription request, resulting in duplicate requests or multiple e-mails. If it is required to have load balancing of FARC subscription notifications, and you are using the same ActiveMQ broker cluster for the FARC and EDDS, then it is possible to configure EDDS and ActiveMQ to use a *Virtual Destination.* This can be easily configured at any time after EDDS has been deployed, as it involves simply changing the Camel route in the context file of each EDDS Server and the ActiveMQ configuration file. This is outside of the scope of this document. More details can be found here: http://activemq.apache.org/virtual-destinations.html

EDDS only uses a durable topic for handling FARC Subscription requests. It does not use a durable topic for sending notifications of check-ins to clients via the web server, as this would result in a client potentially receiving too many messages in one go that are no longer relevant, as clients would not want to be notified of historical events and may not be able to process so many messages at once.

Appendix F Creating RSA Public/Private Key

To be able to use the AES encryption feature of EDDS, it is necessary to create an RSA public/private key. This can be done using the `openssl` command found on SLES 15 and other Linux distributions. Open a Terminal window and enter the following commands:

Generate a private key of length 1024 bits:

```
openssl genrsa -out private.pem 1024
```

Create a private key in the required format:

```
openssl pkcs8 -topk8 -in private.pem -outform DER -out private.der –nocrypt
```

Create the public key from the private key:

```
openssl rsa -in private.pem -pubout -outform DER -out public.der
```

Keep the file “private.der” safe – this private key is needed to decrypt the AES key file that is used to decrypt your response file. The file “public.der” can be copied to the EDDS Server and the location of it must be specified in the edds.properties file in the property “ENCRYPTION_RSA_PUBLIC_KEY_LOCATION”.

Note that EDDS doesn’t provide a way to decrypt the resulting response file(s), however 3rd party applications can be used to do this. Also note that an EDDS Server instance only supports one public key at a time and doesn’t support any key stores, so the key file must be stored and secured with the correct permissions to prevent tampering of the file by third parties.

Note that a key length of 1024 is considered very insecure. It is recommended to create an RSA private key file with at least 2048 bit strength. However, this requires Oracle's Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy to be installed. This can be downloaded from http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

Appendix G Cold Backups

It is possible to have a cold back-up of the ActiveMQ broker, web server and delivery manager, which typically run on a machine in the Relay LAN. This means these components can be completely swapped out with a different installation previously installed on a different machine, without having to transfer the ActiveMQ database. ActiveMQ must still be configured as per Appendix D and continue to use the same username and password as the original installation.

The EDDS Server components store information about scheduled requests in the EDDS database via the Quartz Scheduler. The Quartz Scheduler takes care of ensuring scheduled jobs are persisted in the Quartz database tables within the EDDS database and triggers the appropriate job when it needs to be run. The EDDS Server(s) and EDDS Archiver run in a Quartz cluster, as they share the same database. This means that any one of these components that are running might receive the scheduled job notification. EDDS ensures that the correct EDDS Server processes the scheduled job. It is therefore important to maintain a regular backup of the EDDS database; so that when scheduled jobs stored in the database will be executed at the right time.

The process of processing a scheduled request is as follows:

1. The EDDS Archiver receives notification of a job that needs to be processed from the web server (via a Camel route)
2. The EDDS Archiver schedules the job with the Quartz scheduler so that it is fired at the correct time
3. The Quartz Scheduler stores the job (the details of the request) and trigger (when the job needs to be executed) information in its database tables, and fires the job when the trigger conditions are met (i.e. the request needs to be executed).
4. One of the EDDS backend components receives the job, which extracts the request information and sends a message via ActiveMQ that the request needs to be executed. The correct EDDS Server waiting for requests for the same mission and request type then receives the message (via a Camel route) and processes it. It therefore did not know that the request was actually scheduled, as it received the request in the same way it would have received a request to be executed immediately
5. Note that scheduled jobs remain in the SUBMITTED state until their execution time is reached.

Note that if one of the EDDS backend components is shutdown, you will see a message in the logs from the Quartz Cluster Manager that it detected one or more failed or restarted instances. This is normal, is only for information, and the remaining applications in the cluster will continue to receive the scheduled jobs. If the EDDS Server that needs to process the job is not running at the time, the message sent will remain on the ActiveMQ message queue until it is restarted, at which point the message will be delivered to the EDDS Server.

Appendix H LDAP Configuration

The EDDS software uses the Spring Framework to obtain connections to LDAP. This means that EDDS does not itself care about where the connection came from or if it works or not – this is all managed by Spring through configuration. When EDDS requests a connection, Spring will obtain a previously used connection from its pool and test that it works. If it doesn't, it will transparently obtain another from one of the LDAP connection URLs specified in the configuration and then pass this working connection to EDDS. This means that it is possible to restart LDAP or shutdown one of the servers from the list without having to restart any EDDS components.

As the LDAP connection is all managed through configuration, EDDS supports any set-up of LDAP servers. Typically the LDAP set-up is to have the Master LDAP server that provides read/write access behind a firewall where the EDDS Archiver and Server are running and a Slave LDAP server providing read-only access where the updates are pushed to it from the Master. The Slave LDAP server is used by the Delivery Manager and web server on a less secure LAN.

Often it is necessary to have a redundant backup chain of EDDS in case the prime servers fail, and so being able to have two or more LDAP masters and slaves is required. OpenLDAP supports Mirror Mode replication to keep the servers in sync. More information on OpenLDAP replication can be found here: http://www.openldap.org/doc/admin24/replication.html

H.1 Creating a backup of LDAP data

EDDS provides a useful script for creating a dump of the LDAP server data in LDIF format. To run this script, navigate to where you deployed the master LDAP server (e.g. EDDS_RUNTIME/LDAP) and go into the "scripts" directory. Enter the command:

```
./dumpLdapDb -h
```

The script will display a guidance on how to use it.

In particular, we have the following options:

-h / --help	Optional	Display the help message
-u / --users	Optional	Commma serapated list of user names to use into the filter
-m / --missions	Optional	Comma separated list of mission names to use into the filter
-o / --output	Mandatory	Output file

Both 'users' and 'missions' parameters will allow the insertion of comma separated names. If one(both) parameter(s) is(are) specified, the associated list will be used to filter the output exporting only the requested information.
The output parameter is mandatory and specifies the output file name.

e.g. ./dumpLdapDb —users=Alessandro,Javier —missions=RTE53 -o ldapdump.ldif

H.2 Importing a previous backup into LDAP

You can import a previous backup of LDAP into the master LDAP server by entering the following command from the "scripts" directory of where you deployed the master LDAP server:

```
./importLdapDb ldapdump.ldif
```

Changing "ldapdump.ldif" to the name of the file you want to import from.

Important: do not try to import a backup into the slave LDAP. The master LDAP will push all the changes to the slave.

H.3 Deploying multiple master and slave LDAP servers

The EDDS deployment script supports deploying one LDAP Master server and one LDAP slave on different machines, or a master and a slave co-deployed on the same machine. The deployment scripts do not support multiple masters and slaves with Mirror Mode replication, as this would be too complex. Instead, this must be configured manually by following these instructions.

In this example, we have two LDAP master servers running on different machines and two LDAP slave servers also running on different machines. One master/slave pair represents the main or "prime" servers, and the other master/slave pair represents the backup servers.

H.3.1 LDAP Master Prime

The first step is to configure the edds-deploy.properties file for the LDAP master on the prime server and use the normal deployment script "ant deploy-ldap-master" to deploy the LDAP master on the prime server. Next navigate to the directory to where LDAP has been deployed and use the "stop" script to stop the server (e.g. in EDDS_RUNTIME/LDAP/scripts).

Edit the file config/master.conf in the LDAP directory where the LDAP master server was deployed. Before the section that starts "Consumer Proxy that pulls in data via Syncrepl and pushes out via slapd-ldap" copy and paste the following configuration:

```
#######################################################################
##
# Mirror Mode configuration for replication to other masters
#######################################################################
##

serverID        1

syncrepl        rid=002
                provider=ldap://10.48.19.213:20005/
                binddn="cn=Manager,o=edds"
                bindmethod=simple
                credentials=password
                searchbase="o=edds"
                schemachecking=on
                type=refreshAndPersist
                retry="60 +"

mirrormode      on
overlay         syncprov
```

Change the entry "10.48.19.213:20005" to the IP address or hostname and port of the LDAP master server that will be running on the backup server. Change the entry "cn=Manager,o=edds" and "password" to match the username and password of the LDAP master server running on the backup server.

H.3.2 LDAP Master Backup

The second step is to configure the edds-deploy.properties file for the LDAP master on the backup server and use the normal deployment script "ant deploy-ldap-master" to deploy the LDAP master on the backup server as before. Next navigate to the directory to where LDAP has been deployed and use the "stop" script to stop the server (e.g. in EDDS_RUNTIME/LDAP/scripts).

Edit the file config/master.conf in the LDAP directory where the LDAP master server was deployed. Before the section that starts “Consumer Proxy that pulls in data via Syncrepl and pushes out via slapd-ldap” copy and paste the same Mirror Mode configuration text in the previous section:

Change the entry “10.48.19.213:20005” to the IP address or hostname and port of the LDAP master server that you just deployed on the prime server. Change the entry “cn=Manager,o=edds” and “password” to match the username and password of the LDAP master server running on the prime server.

H.3.3 LDAP Slave Prime

The third step is to configure the edds-deploy.properties file for the LDAP slave on the prime server chain and use the normal deployment script “ant deploy-ldap-slave” to deploy the LDAP slave on the prime server. No additional configuration is needed. You can now start the LDAP slave server using the “start” script followed by the LDAP master server deployed on the prime.

H.3.4 LDAP Slave Backup

The fourth and final step is to configure the edds-deploy.properties file for the LDAP slave on the backup server chain and use the normal deployment script “ant deploy-ldap-slave” to deploy the LDAP slave on the backup server. No additional configuration is needed. You can now start the LDAP slave server using the “start” script followed by the LDAP master server deployed on the backup.

Once LDAP replication has been configured, EDDS will work with LDAP should the prime server go down without any changes to configuration.

Appendix I Using EDDS with an Existing Central LDAP Installation

Instead of authenticating users against the EDDS LDAP Server, EDDS can authenticate users against a separate LDAP installation, referred to as the “Central LDAP”. EDDS does this by allowing EDDS Administrators to add a user that exists in the Central LDAP server as a user within EDDS by ticking a checkbox on the new user creation form within the EDDS MMI. When an EDDS Administrator does this, the EDDS Web Server verifies that the username exists in the central LDAP server. If it does, then the user is created within the EDDS LDAP to store all the user’s EDDS specific information but with a flag set that indicates that the password must be verified using the central LDAP server.

When the user logs on, the EDDS Web Server will check the central LDAP flag, and if it is set, it will attempt to perform a bind operation on the central LDAP server with the username and password provided. If the bind operation succeeds, then the username and password is verified and the user can proceed to use EDDS. The EDDS permissions for the user are set in the same way as for a regular EDDS user. The only difference for the user is that they cannot change their password using the EDDS MMI and the EDDS Administrator cannot change the user’s password or set any password policy. Instead, the password must be managed centrally.

To configure EDDS to use a central LDAP installation, it is necessary to have the following information:

- The username (bind DN) to connect to the LDAP server
- The password
- The location in the LDAP structure where the users are (e.g. o=People,o=esoc,dn=esa,dn=int)

If you do not have a username and password to connect to the LDAP server, and instead must connect anonymously (anonymous look-ups must be enabled) then the following changes must be made to the EDDS Web Server context file after deployment:

1) Go to where Apache Tomcat is installed
2) Go into the webapps/edds/WEB-INF directory
3) Edit the file eddscontext.xml
4) Search for the bean called “centralLdapContextSource”
5) Remove the two lines:
   ```
   <property name="userDn" value="" />
   <property name="password" value="" />
   ```
6) Add the following line in its place:
   ```
   <property name="anonymousReadOnly" value="true" />
   ```

Appendix J Example Requests for the Request Submitter

Batch requests can be created and saved in the EDDS MMI. These XML files can then be used as templates to be submitted to the Request Submitter. The request file of previously submitted requests can also be downloaded from the EDDS MMI. As it is not possible to download the XML request for cancel, suspend and resume requests, this appendix includes examples that can be copied as a template. The examples can also be found in the EDDS source code under `edds/edds-request-submitter/src/test/resources/EDDSRequests`

J.1 Cancel Requests

```
<?xml version="1.0" encoding="UTF-8"?>
<p:CancelPartList xmlns:p="http://edds.egos.esa/model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://edds.egos.esa/model ../../../../../edds-ws-
common/src/main/wsdl/batchrequest.xsd">
	<CancelPart>
		<User>
			<UserName>aresuser</UserName>
		</User>
		<JobIdPart>

	<Job>BatchRequest.Param.TEST_MISSION.1.2014.031.10.52.47.692</Job>
		</JobIdPart>
	</CancelPart>
</p:CancelPartList>
```

J.2 Suspend Requests

```
<?xml version="1.0" encoding="UTF-8"?>
<p:SuspendPartList xmlns:p="http://edds.egos.esa/model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://edds.egos.esa/model ../../../../../edds-ws-
common/src/main/wsdl/batchrequest.xsd">
	<SuspendPart>
		<User>
			<UserName>aresuser</UserName>
		</User>
		<JobIdPart>

	<Job>BatchRequest.Param.TEST_MISSION.1.2014.031.10.52.47.692</Job>
		</JobIdPart>
	</SuspendPart>
</p:SuspendPartList>
```

J.3 Resume Requests

```
<?xml version="1.0" encoding="UTF-8"?>
<p:ResumePartList xmlns:p="http://edds.egos.esa/model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://edds.egos.esa/model ../../../../../edds-ws-
common/src/main/wsdl/batchrequest.xsd">
      <ResumePart>
            <User>
                  <UserName>aresuser</UserName>
            </User>
            <JobIdPart>

      <Job>BatchRequest.Param.TEST_MISSION.1.2014.031.10.52.47.692</Job>
            </JobIdPart>
      </ResumePart>
</p:ResumePartList>
```

Appendix K Development Environment Resources

K.1 Formatting.xml

The formatting template for Java source code that needs to be imported into Eclipse.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<profiles version="11">
<profile kind="CodeFormatterProfile" name="EDDS" version="11">
<setting id="org.eclipse.jdt.core.formatter.comment.insert_new_line_before_root_tags" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_annotation" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_type_parameters" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_type_declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_type_arguments" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_anonymous_type_declaration" value="next_line"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_case" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_brace_in_array_initializer" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_annotation_declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_closing_brace_in_array_initializer" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_field" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_while" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_in_annotation_type_member_declaration"
value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_else_in_if_statement" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_prefix_operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.keep_else_statement_on_same_line" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_ellipsis" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.insert_new_line_for_parameter" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_annotation_type_declaration" value="do
not insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_breaks_compare_to_cases" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_at_in_annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_multiple_fields" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_expressions_in_array_initializer" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_conditional_expression" value="80"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_for" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_binary_operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_question_in_wildcard" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_array_initializer" value="end_of_line"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_in_enum_constant" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_finally_in_try_statement" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_on_local_variable" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_catch_in_try_statement" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_while" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_after_package" value="1"/>
```

```
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_type_parameters" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.continuation_indentation" value="2"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_postfix_operator" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_method_invocation" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_angle_bracket_in_type_arguments" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_superinterfaces" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_new_chunk" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_binary_operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_package" value="0"/>
<setting id="org.eclipse.jdt.core.compiler.source" value="1.5"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_enum_constant_arguments" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_constructor_declaration" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_closing_angle_bracket_in_type_arguments" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.format_line_comments" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_enum_declarations" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.join_wrapped_lines" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_block" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_explicit_constructor_call" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_method_invocation_arguments" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_member_type" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.align_type_members_on_columns" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_enum_constant" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_for" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_method_declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_selector_in_method_invocation" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_switch" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_unary_operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_case" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.indent_parameter_description" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_method_declaration" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_switch" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_enum_declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_angle_bracket_in_type_parameters" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_type_declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.clear_blank_lines_in_block_comment" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.lineSplit" value="140"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_if" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_brackets_in_array_type_reference" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_parenthesized_expression" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_explicitconstructorcall_arguments" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_constructor_declaration" value="insert"/>
```

```
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_first_class_body_declaration" value="0"/>
<setting id="org.eclipse.jdt.core.formatter.indentation.size" value="4"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_in_method_declaration" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_enum_constant" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_superclass_in_type_declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_assignment" value="0"/>
<setting id="org.eclipse.jdt.core.compiler.problem.assertIdentifier" value="error"/>
<setting id="org.eclipse.jdt.core.formatter.tabulation.char" value="space"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_constructor_declaration_parameters"
value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_prefix_operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_statements_compare_to_body" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_method" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.format_guardian_clause_on_one_line" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_for" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_cast" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_parameters_in_constructor_declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_labeled_statement" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_annotation_type_declaration" value="next_line"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_method_body" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_method_invocation" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_bracket_in_array_allocation_expression" value="do
not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_enum_constant" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_at_in_annotation_type_declaration" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_method_declaration_throws" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_if" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_switch" value="next_line"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_method_declaration_throws" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_parenthesized_expression_in_return" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_question_in_conditional" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_question_in_wildcard" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_bracket_in_array_allocation_expression" value="do
not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_parenthesized_expression_in_throw" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_type_arguments" value="do not insert"/>
<setting id="org.eclipse.jdt.core.compiler.problem.enumIdentifier" value="error"/>
<setting id="org.eclipse.jdt.core.formatter.indent_switchstatements_compare_to_switch" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_ellipsis" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_block" value="next_line"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_for_inits" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_method_declaration" value="next_line"/>
```

```xml
<setting id="org.eclipse.jdt.core.formatter.compact_else_if" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_array_initializer" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_for_increments" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_bracket_in_array_reference" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_enum_constant" value="next_line"/>
<setting id="org.eclipse.jdt.core.formatter.comment.indent_root_tags" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_enum_declarations" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_explicitconstructorcall_arguments"
value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_switch" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_superinterfaces" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_method_declaration_parameters" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_allocation_expression" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.tabulation.size" value="4"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_bracket_in_array_type_reference" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_opening_brace_in_array_initializer" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_closing_brace_in_block" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_bracket_in_array_reference" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_enum_constant" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_angle_bracket_in_type_arguments" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_constructor_declaration" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_if" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_constructor_declaration_throws" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.clear_blank_lines_in_javadoc_comment" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_throws_clause_in_constructor_declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_assignment_operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_assignment_operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_empty_lines" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_synchronized" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_closing_paren_in_cast" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_method_declaration_parameters" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_block_in_case" value="next_line"/>
<setting id="org.eclipse.jdt.core.formatter.number_of_empty_lines_to_preserve" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_method_declaration" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_catch" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_constructor_declaration" value="do not
insert"/>
```

```
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_method_invocation" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_bracket_in_array_reference" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_and_in_type_parameter" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_qualified_allocation_expression" value="16"/>
<setting id="org.eclipse.jdt.core.compiler.compliance" value="1.5"/>
<setting id="org.eclipse.jdt.core.formatter.continuation_indentation_for_array_initializer" value="2"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_brackets_in_array_allocation_expression" value="do
not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_at_in_annotation_type_declaration" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_allocation_expression" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_cast" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_unary_operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_angle_bracket_in_parameterized_type_reference"
value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_anonymous_type_declaration"
value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.keep_empty_array_initializer_on_one_line" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_enum_declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.keep_imple_if_on_one_line" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_constructor_declaration_parameters" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_closing_angle_bracket_in_type_parameters" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_at_end_of_file_if_missing" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_for" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_labeled_statement" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_parameterized_type_reference" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_superinterfaces_in_type_declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_binary_expression" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_enum_declaration" value="next_line"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_while" value="do not insert"/>
<setting id="org.eclipse.jdt.core.compiler.codegen.inlineJsrBytecode" value="enabled"/>
<setting id="org.eclipse.jdt.core.formatter.put_empty_statement_on_new_line" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_on_parameter" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_angle_bracket_in_type_parameters" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_in_method_invocation" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_while_in_do_statement" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_enum_constant" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.comment.format_javadoc_comments" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.comment.line_length" value="140"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_between_import_groups" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_enum_constant_arguments" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_semicolon" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_constructor_declaration" value="next_line"/>
```

```
<setting id="org.eclipse.jdt.core.formatter.number_of_blank_lines_at_beginning_of_method_body" value="0"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_conditional" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_body_declarations_compare_to_type_header" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_annotation_type_member_declaration"
value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.wrap_before_binary_operator" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.indent_body_declarations_compare_to_enum_declaration_header" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_between_type_declarations" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_synchronized" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_statements_compare_to_block" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_superinterfaces_in_enum_declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.join_lines_in_comments" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_question_in_conditional" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_multiple_field_declarations" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_compact_if" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_for_inits" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_switchstatements_compare_to_cases" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_array_initializer" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_default" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_and_in_type_parameter" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_in_constructor_declaration" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_assert" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_imports" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.comment.format_html" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_throws_clause_in_method_declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_angle_bracket_in_type_parameters" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_bracket_in_array_allocation_expression" value="do
not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_anonymous_type_declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_conditional" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_angle_bracket_in_parameterized_type_reference"
value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_for" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_postfix_operator" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.format_source_code" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_synchronized" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_allocation_expression" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_constructor_declaration_throws" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_parameters_in_method_declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_brace_in_array_initializer" value="insert"/>
<setting id="org.eclipse.jdt.core.compiler.codegen.targetPlatform" value="1.5"/>
<setting id="org.eclipse.jdt.core.formatter.use_tabs_only_for_leading_indentations" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_on_member" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.format_header" value="false"/>
```

```
<setting id="org.eclipse.jdt.core.formatter.comment.format_block_comments" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_enum_constant" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_enum_constants" value="0"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_block" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_body_declarations_compare_to_annotation_declaration_header"
value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_parenthesized_expression" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_parenthesized_expression"
value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_catch" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_multiple_local_declarations" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_switch" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_for_increments" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_method_invocation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_assert" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_type_declaration" value="next_line"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_in_array_initializer" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_braces_in_array_initializer" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_in_method_declaration" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_semicolon_in_for" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_in_catch" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_angle_bracket_in_parameterized_type_reference"
value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_multiple_field_declarations" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_in_annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_parameterized_type_reference" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_method_invocation_arguments" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_after_imports" value="1"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_multiple_local_declarations" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_body_declarations_compare_to_enum_constant_header" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_semicolon_in_for" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.never_indent_line_comments_on_first_column" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_angle_bracket_in_type_arguments" value="do not
insert"/>
<setting id="org.eclipse.jdt.core.formatter.never_indent_block_comments_on_first_column" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.keep_then_statement_on_same_line" value="false"/>
</profile>
</profiles>
```

K.2 Codetemplate.xml

The code templates for Java source code that needs to be imported into Eclipse.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><templates><template autoinsert="true"
context="constructorbody_context" deleted="false" description="Code in created constructor stubs" enabled="true"
id="org.eclipse.wst.jsdt.ui.text.codetemplates.constructorbody" name="constructorbody">${body_statement}
// ${todo} Auto-generated constructor stub</template><template autoinsert="false" context="newtype_context"
deleted="false" description="Newly created files" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.newtype"
name="newtype">${filecomment}
${package_declaration}

${typecomment}
${type_declaration}
//////
// $$Log: $$
//</template><template autoinsert="true" context="setterbody_context" deleted="false" description="Code in created
setters" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.setterbody" name="setterbody">${field} =
${param};</template><template autoinsert="true" context="getterbody_context" deleted="false" description="Code in created
getters" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.getterbody" name="getterbody">return
${field};</template><template autoinsert="true" context="methodcomment_context" deleted="false" description="Comment for
non-overriding methods" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.methodcomment" name="methodcomment">/**
 * ${tags}
 */</template><template autoinsert="true" context="catchblock_context" deleted="false" description="Code in new catch
blocks" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.catchblock" name="catchblock">// ${todo} Auto-generated
catch block
${exception_var}.printStackTrace();</template><template autoinsert="true" context="methodcomment_context" deleted="false"
description="Comment for non-overriding function" enabled="true"
id="org.eclipse.wst.jsdt.ui.text.codetemplates.methodcomment" name="methodcomment">/**
 * ${tags}
 */</template><template autoinsert="true" context="methodbody_context" deleted="false" description="Code in created method
stubs" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.methodbody" name="methodbody">// ${todo} Auto-generated
method stub
${body_statement}</template><template autoinsert="true" context="settercomment_context" deleted="false"
description="Comment for setter function" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.settercomment"
name="settercomment">/**
 * @param ${param} the ${bare_field_name} to set
 */</template><template autoinsert="true" context="fieldcomment_context" deleted="false" description="Comment for fields"
enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.fieldcomment" name="fieldcomment">/**
 *
 */</template><template autoinsert="true" context="fieldcomment_context" deleted="false" description="Comment for vars"
enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.fieldcomment" name="fieldcomment">/**
 *
 */</template><template autoinsert="true" context="interfacebody_context" deleted="false" description="Code in new
interface type bodies" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.interfacebody" name="interfacebody">
</template><template autoinsert="true" context="filecomment_context" deleted="false" description="Comment for created
JavaScript files" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.filecomment" name="filecomment">/**
 *
 */</template><template autoinsert="true" context="typecomment_context" deleted="false" description="Comment for created
types" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.typecomment" name="typecomment">/**
 * @author ${user}
 *
 * ${tags}
 */</template><template autoinsert="true" context="gettercomment_context" deleted="false" description="Comment for getter
function" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.gettercomment" name="gettercomment">/**
 * @return the ${bare_field_name}
 */</template><template autoinsert="true" context="settercomment_context" deleted="false" description="Comment for setter
method" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.settercomment" name="settercomment">/**
 * @param ${param} the ${bare_field_name} to set
 */</template><template autoinsert="false" context="overridecomment_context" deleted="false" description="Comment for
overriding methods" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.overridecomment" name="overridecomment">/**
 * ${see_to_overridden}
 */</template><template autoinsert="true" context="enumbody_context" deleted="false" description="Code in new enum type
bodies" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.enumbody" name="enumbody">
</template><template autoinsert="true" context="constructorcomment_context" deleted="false" description="Comment for
created constructors" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.constructorcomment"
name="constructorcomment">/**
 * ${tags}
 */</template><template autoinsert="true" context="delegatecomment_context" deleted="false" description="Comment for
delegate methods" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.delegatecomment" name="delegatecomment">/**
 * ${tags}
 * ${see_to_target}
 */</template><template autoinsert="true" context="gettercomment_context" deleted="false" description="Comment for getter
method" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.gettercomment" name="gettercomment">/**
 * @return the ${bare_field_name}
 */</template><template autoinsert="false" context="typecomment_context" deleted="false" description="Comment for created
types" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.typecomment" name="typecomment">/**
 * TODO write here a description of the class
 *
 * @author     edds team
*/</template><template autoinsert="true" context="classbody_context" deleted="false" description="Code in new class type
bodies" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.classbody" name="classbody">
</template><template autoinsert="true" context="classbody_context" deleted="false" description="Code in new class type
bodies" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.classbody" name="classbody">
</template><template autoinsert="true" context="setterbody_context" deleted="false" description="Code in created setters"
enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.setterbody" name="setterbody">${field} =
${param};</template><template autoinsert="true" context="annotationbody_context" deleted="false" description="Code in new
annotation type bodies" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.annotationbody" name="annotationbody">
</template><template autoinsert="true" context="catchblock_context" deleted="false" description="Code in new catch blocks"
enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.catchblock" name="catchblock">// ${todo} Auto-generated
catch block
${exception_var}.printStackTrace();</template><template autoinsert="true" context="getterbody_context" deleted="false"
description="Code in created getters" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.getterbody"
name="getterbody">return ${field};</template><template autoinsert="true" context="constructorbody_context" deleted="false"
description="Code in created constructor stubs" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.constructorbody"
name="constructorbody">${body_statement}
```

```
// ${todo} Auto-generated constructor stub</template><template autoinsert="true" context="constructorcomment_context"
deleted="false" description="Comment for created constructors" enabled="true"
id="org.eclipse.jdt.ui.text.codetemplates.constructorcomment" name="constructorcomment">/**
 * ${tags}
 */</template><template autoinsert="false" context="newtype_context" deleted="false" description="Newly created files"
enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.newtype" name="newtype">${filecomment}
${package_declaration}

${typecomment}
${type_declaration}
</template><template autoinsert="false" context="overridecomment_context" deleted="false" description="Comment for
overriding functions" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.overridecomment"
name="overridecomment">/**
 * ${see_to_overridden}
 */</template><template autoinsert="true" context="methodbody_context" deleted="false" description="Code in created
function stubs" enabled="true" id="org.eclipse.wst.jsdt.ui.text.codetemplates.methodbody" name="methodbody">// ${todo}
Auto-generated function stub
${body_statement}</template><template autoinsert="true" context="delegatecomment_context" deleted="false"
description="Comment for delegate functions" enabled="true"
id="org.eclipse.wst.jsdt.ui.text.codetemplates.delegatecomment" name="delegatecomment">/**
 * ${tags}
 * ${see_to_target}
 */</template><template autoinsert="false" context="filecomment_context" deleted="false" description="Comment for created
Java files" enabled="true" id="org.eclipse.jdt.ui.text.codetemplates.filecomment" name="filecomment">/* ------------------
----------------------------------------------------------&#13;
 * (c) Copyright European Space Agency, 2016&#13;
 *             European Space Operations Centre&#13;
 *             Darmstadt Germany&#13;
 * The copyright of this document is vested in the European Space Agency. This&#13;
 * document may only be reproduced in whole or in part, stored in a retrieval&#13;
 * system, transmitted in any form, or by any means e.g. electronically,&#13;
 * mechanically or by photocopying, or otherwise, with the prior permission of&#13;
 * the Agency.&#13;
 * ----------------------------------------------------------------------------&#13;
 * System       : EDDS&#13;
 * Component     : ${project_name}&#13;
 * Classname    : ${package_name}.${file_name} &#13;
* ----------------------------------------------------------------------------&#13;
 */</template></templates>
```

K.3 JAutodoc

K.3.1 File Header

/* --

 * (c) Copyright European Space Agency, 2016

```
*           European Space Operations Centre
*           Darmstadt Germany
* The copyright of this document is vested in the European Space Agency. This
* document may only be reproduced in whole or in part, stored in a retrieval
* system, transmitted in any form, or by any means e.g. electronically,
* mechanically or by photocopying, or otherwise, with the prior permission of
* the Agency.
* ---------------------------------------------------------------------------
* System       : EDDS
* Component     : ${project_name}
* Classname    : ${package_name}.${file_name}
* ---------------------------------------------------------------------------
*/
```

Appendix L EDDS DevOps Project Configuration

EDDS DevOps pipeline can be triggered manually from the gitlab user interface
https://gitlab.esa.int/edds/EDDS

From the GitLab Project Menu, Navigate to Build -> Pipelines. This will list all the pipelines run for EDDS projects for different branches.

Click on "Run pipeline" button. Select the branch for which the pipeline should be run. For example: EDDS_3/development branch. No variables are needed. Run pipeline.

The pipeline is configured for the EDDS project. The EDDS pipeline is run specific to the source code branch.Multiple pipelines can be run for different branches at the same time. Refer to SUM section 6.5.1

L.1 Configuring Mission for EDDS deployment

EDDS deployment is mission specific. The EDDS project has to be configured to contain configurations for all missions. The EDDS Server, Delivery Manager, Request submitter has to be configured with the mission specific configuration. The EDDS deployment will be done per mission.

Create a directory with {MISSION_NAME} inside {EDDS_Source_Dir}/edds/edds-mission-config

L.1.1 Configuring EDDS Server

1) Go into the base directory of the mission configuration {EDDS_Source_Dir}/edds/edds-mission-config/{MISSION_NAME}
2) Create a directory 'server' inside it.
3) Go into the mission specific 'server' directory {EDDS_Source_Dir}/edds/edds-mission-config/{MISSION_NAME}/server. Copy the mission specific file "edds.properties" for edds server into 'server' directory.
4) Create a directory 'config' inside {EDDS_Source_Dir}/edds/edds-mission-config/{MISSION_NAME}/server. Go into the 'config' directory. Copy the mission specific files "testRSAPublicKey.txt" and "Aliases.conf" into the 'config' directory.

Refer to the example: {EDDS_Source_Dir}/edds/edds-mission-config/test-mission-1/server

L.1.2 Configuring Delivery Manager

1) Go into the base directory of the mission configuration {EDDS_Source_Dir}/edds/edds-mission-config/{MISSION_NAME}
2) Create a directory 'delivery-mission-specific-config' inside it.
3) Go into the mission specific 'server' directory {EDDS_Source_Dir}/edds/edds-mission-config/{MISSION_NAME}/delivery-mission-specific-config. Copy the mission specific file "edds.properties" for delivery manager into 'delivery-mission-specific-config' directory.
4) Create a directory 'mission-specific-config' inside {EDDS_Source_Dir}/edds/edds-mission-config/{MISSION_NAME}/delivery-mission-specific-config. Go into the 'mission-specific-config'' directory. Copy the mission specific xml configuration file for delivery manager into the 'mission-specific-config'' directory.

Refer to the example: {EDDS_Source_Dir}/edds/edds-mission-config/test-mission-1/delivery-mission-specific-config/

L.1.3 Configuring Request Submitter

More than one Request Submitter can be configured for a mission. For the deployment, only one request submitter configuration will be taken.

1) Go into the base directory of the mission configuration {EDDS_Source_Dir}/edds/edds-mission-config/{MISSION_NAME}
2) Create a directory 'request-submitter-{config-context}' inside it
3) Go into the mission specific 'server' directory {EDDS_Source_Dir}/edds/edds-mission-config/{MISSION_NAME}/request-submitter-{config-context}. Copy the mission specific file "config.properties" for request submitter into 'request-submitter-{config-context}' directory.
4) Repeat the above steps from step 2 to create multiple request submitter configuration for the mission based on the request submitter context {config-context}.

   Refer to the example: {EDDS_Source_Dir}/edds/edds-mission-config/test-mission-1/request-submitter-alpha/ and
   {EDDS_Source_Dir}/edds/edds-mission-config/test-mission-1/request-submitter-beta/

L.1.4 Configuring the deployment

EDDS is deployed into the Kubernetes cluster per mission based on their configuration.

The HELM deployment values.yaml contains the name of the mission, secret and the request submitter configuration. To configure these information , the following configuration in {EDDS_Source_Dir}/edds-helm-chart/values.yaml has to be changed before running the pipeline :

```
mission:
  missionName: {MISSION_NAME}
  secretName: {SECRET_NAME}
  requestSubmitterSpecific: {config-context}
```

missionName : The name of the mission. This should be same as the {MISSION_NAME} used to configure EDDS Server, delivery manager and request submitter in the above step.

secretName : The name of the secret which will contain the configuration of the mission, The mission configuration is stored as secret configuration in EDDS Kubernetes cluster. The secret name can be any name and the `{SECRET_NAME}` attribute value has to be configured in {EDDS_Source_Dir}/edds-helm-chart/values.yaml

requestSubmitterSpecific: The name of the request submitter context configuration which should be used for the mission. The value should be {config-context}' created in step 2 of section L.1.3.

Refer to the example: {EDDS_Source_Dir}/edds-helm-chart/values.yaml

After applying the above configuration, Run the EDDS pipeline manually. After the pipeline is completed, An EDDS instance will be deployed in the Kubernetes cluster for the mission with all the EDDS components running.