

**Oracle® Healthcare Data Repository**

FHIR User's Guide

Release 8.1

**F25715-01**

January 2020

Copyright © 2018, 2020 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

---

---

# Contents

<b>Preface</b> .....	v
Audience .....	v
Documentation Accessibility .....	v
Related Documents .....	v
<b>1 Getting Started</b>	
1.1 Introduction .....	1-1
1.2 Platform Requirements .....	1-1
<b>2 Installation</b>	
<b>3 HDR FHIR Server Architecture</b>	
<b>4 Deployment Details</b>	
4.1 Install files .....	4-1
4.2 FHIR Server Base URL and REST Endpoints.....	4-1
4.3 Configuration Files .....	4-1
4.3.1 Properties File.....	4-1
4.3.2 Log Configurations.....	4-4
<b>5 Using the OAuth 2.0 protected API</b>	
5.1 Prerequisites .....	5-1
5.2 How It Works .....	5-1
5.3 Obtaining the Access Token from the OAuth Server .....	5-2
5.4 Calling the HDR FHIR API with an Access Token.....	5-2
5.5 Error Messages .....	5-2
<b>6 Auditing</b>	
6.1 Audit Interceptor Execution Flow.....	6-1
6.2 Audit Record Format.....	6-2
6.3 Settings .....	6-2

## 7 FHIR Command-Line Utility

7.1	Prerequisites .....	7-1
7.2	Commands.....	7-1
7.2.1	export-conceptmap-to-csv .....	7-2
7.2.2	import-csv-to-conceptmap .....	7-3
7.2.3	upload-definitions and upload-examples .....	7-4
7.2.4	upload-terminology.....	7-4

## 8 Working with FHIR REST APIs

---

---

# Preface

Welcome to the Installation Guide for Oracle Healthcare Data Repository (HDR) Version 8.1.

## Audience

This document is intended for an audience of Oracle Applications Database Administrators who plan to install the Oracle Healthcare Data Repository (HDR) either locally or through a VPN connection to the servers.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

## Related Documents

The following set of documents have been referenced in this installation guide. All of these documents should be downloaded or printed from My Oracle Support and used as your guide during all HDR installations. You will find that you will add your own comments and corrections to these documents. Should you find errors, inconsistencies or missing information in any of these documents, please log an SR via My Oracle Support so that the documentation error can be corrected. It is recommended that you return to My Oracle Support and check for updates to these documents on a regular basis.

Note that you may obtain the following documents by either downloading these from <https://edelivery.oracle.com/> or from the appropriate media (CD or DVD) in the physical media pack. For downloading a document from *My Oracle Support*, use My Oracle Support Article ID to search for the particular document.

**Integration and Other Product References:**

Oracle Healthcare Data Repository Javadoc. This document describes the HDR Application Programming Interface. It defines all of the Classes and Interfaces included in the release.

Oracle Healthcare Data Repository Implementation Guide. This document details all of the steps required to implement the variable functional components of the HDR Platform. Once analysis has been performed to determine which parts of the HDR Platform will be utilized, this document will detail the prerequisites and process steps needed to implement the functionality.

Oracle Healthcare Data Repository Programmer's Guide. This document is organized around code samples that address common setup functions and application features, with emphasis on application functionality typically used in healthcare settings. The examples include code samples targeted to developers writing code for HDR setup and application development.

---

---

# Getting Started

This section provides information about the platform requirements for setting up an Oracle Healthcare Data Repository HL7 FHIR server.

## 1.1 Introduction

Fast Healthcare Interoperability Resources (FHIR) is a standard for healthcare data exchange that is published by HL7 ([www.hl7.org/fhir/](http://www.hl7.org/fhir/)).

Oracle Healthcare Data Repository (HDR) Release 8.1 supports the HL7 FHIR specification version R4. The FHIR server module is distributed as a web application, which can be deployed to standard web containers such as WebLogic. FHIR resources are exposed as a set of REST APIs that can be accessed by REST-based applications.

## 1.2 Platform Requirements

The following software is required for Oracle Healthcare Data Repository and the FHIR server module:

- Operating System: Oracle Enterprise Linux 6.x or 7.x (64 bit)
- Oracle Database 12c Release 1 (12.1.0.2.0) or Release 2 (12.2.1.2.0) or Release 19c. Download from the Oracle Software Delivery Cloud at <https://edelivery.oracle.com>.
- WebLogic Server 12.2.1.3/12.2.1.4 with the Coherence option. Download from the Oracle Software Delivery Cloud at <https://edelivery.oracle.com>.
- JDK (Java Development Kit) 8u121 and later. Download from My Oracle Support. See Section 3.1, "Download and install Oracle Java Development Kit".

Download Oracle Healthcare Data Repository 8.1 from the Oracle Software Delivery Cloud at <https://edelivery.oracle.com>.





# 2

---

---

## Installation

Refer to the Oracle Oracle Healthcare Data Repository 8.1 installation guide for information on the prerequisites and steps to install the HDR components.



---

## HDR FHIR Server Architecture

Healthcare functional domains (administrative, clinical, and financial) and core services are exposed as RESTful endpoints.

The client application that uses the HDR FHIR RESTful APIs must properly configure the system in a secure environment to avoid unauthorized access of those APIs. To access protected FHIR APIs, the client application must provide a valid access token. For more information on this, see the HDR 8.1 *Secure Configuration Guide* and *Secure Development Guide*.

The FHIR Server can be deployed in a single or multi-node cluster as illustrated below:

**Figure 3–1 Single Node**

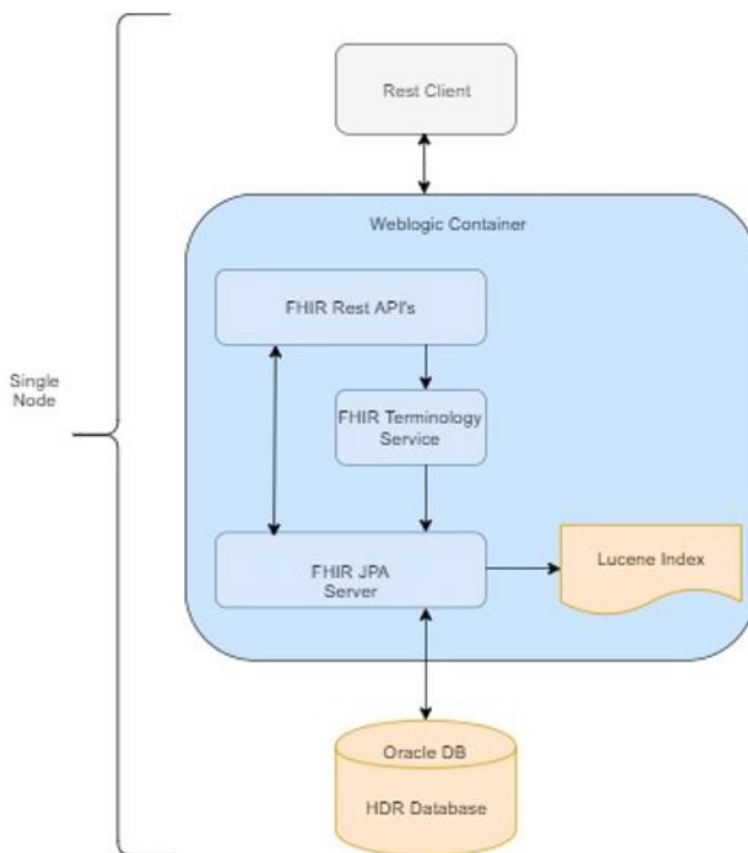
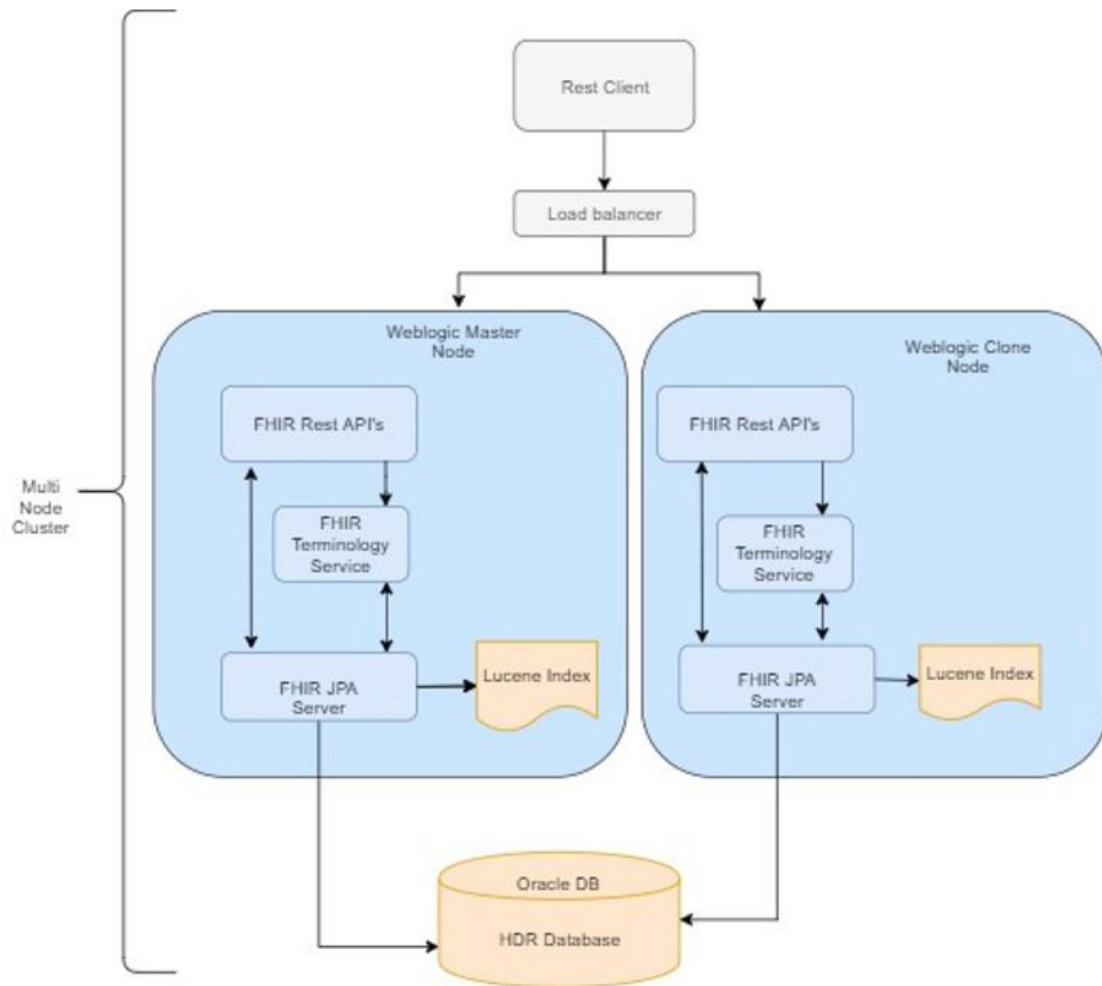


Figure 3-2 Multi-Node Cluster



---

---

## Deployment Details

This section describes the details of the HDR FHIR Server deployment.

### 4.1 Install files

The HDR FHIR Server module is packaged as a deployable web application (`.war`) file. The war file is distributed along with the HDR 8.1 installation binary. Once installed, FHIR resources are exposed as a set of REST endpoints.

The other component, the FHIR command line tool, is available under the HDR HOME directory on the middle tier. Refer to [Chapter 7, "FHIR Command-Line Utility"](#) section for more details.

### 4.2 FHIR Server Base URL and REST Endpoints

FHIR REST APIs can be accessed using the base URL as show below:

```
http://HOSTNAME:PORT/oracle-fhir-server/fhir
```

A specific resource can be accessed using the URL format:

```
<BASE_URL>/<resourceName>
```

For example, to access the 'Patient' resource, use the following URL:

```
http://HOSTNAME:PORT/oracle-fhir-server/fhir/Patient
```

For the complete list of resources and their corresponding URLs, see [Chapter 8, "Working with FHIR REST APIs"](#).

### 4.3 Configuration Files

Runtime behavior and logging are controlled using configuration files.

#### 4.3.1 Properties File

The runtime behavior of the FHIR JPA server can be managed using the `hdr_fhir.properties` file: `<HDR_DOMAIN>/config/fhir/hdr_fhir.properties`.

**Example 4-1** *hdr\_fhir.properties* file

```
# Supported FHIR version
fhir_version=R4
#below entries are for capability statement
metadata.implementation.description=Oracle FHIR Server
metadata.software.name=Oracle FHIR Server
```

```

metadata.publisher=Oracle FHIR Server
# true or false
metadata.resource.count.enabled=false

# This is the address that the FHIR server will report as its own address.
# If this server will be deployed (for example) to an internet accessible
# server, put the DNS name of that server here.

server_address=http://HOSTNAME:PORT/oracle-fhir-server/fhir
server.base=/oracle-fhir-server/fhir

default_encoding=JSON
#resource compression setting - if no value set, server defaults to compression
enabled.
resource_compression_enabled=true
etag_support=ENABLED
reuse_cached_search_results_millis=-1
default_page_size=20
max_page_size=200
allow_override_default_search_params=true
allow_contains_searches=true
allow_multiple_delete=true
allow_external_references=true
allow_cascading_deletes=true
allow_placeholder_references=true
expunge_enabled=true
persistence_unit_name=FHIR_PU
logger.name=fhirtest.access
logger.format=Path[${servletPath}] Source[${requestHeader.x-forwarded-for}]
Operation[${operationType} ${operationName} ${idOrResourceName}]
UA[${requestHeader.user-agent}] Params[${requestParameters}]
ResponseEncoding[${responseEncodingNoDefault}]
logger.error_format=ERROR - ${requestVerb} ${requestUrl}
logger.log_exceptions=true

server.name=HDR FHIR Server
server.id=home
test.port=

#####
# Validation
#####
# Should all incoming requests be validated
validation.requests.enabled=false
# Should outgoing responses be validated
validation.responses.enabled=false

#####
# Search Features
#####
filter_search.enabled=true
graphql.enabled=true

#####
# Supported Resources
#####
# Enable the following property if you want to customize the
# list of resources that is supported by the server (i.e. to
# disable specific resources)
#supported_resource_types=Patient,Observation,Encounter

```

```
#####
# Database Settings
#####
hibernate.dialect=org.hibernate.dialect.Oracle12cDialect
hibernate.search.model_mapping=ca.uhn.fhir.jpa.search.LuceneSearchMappingFactory
hibernate.format_sql=false
hibernate.show_sql=false
#hibernate.hbm2ddl.auto=update
hibernate.hbm2ddl.auto=none
hibernate.jdbc.batch_size=20
hibernate.cache.use_query_cache=false
hibernate.cache.use_second_level_cache=false
hibernate.cache.use_structured_entries=false
hibernate.cache.use_minimal_puts=false
hibernate.search.default.directory_provider=filesystem
hibernate.search.default.indexBase=target/lucenefiles
hibernate.search.lucene_version=LUCENE_CURRENT
tester.config.refuse_to_fetch_third_party_urls=false
hibernate.search.autoregister_listeners=false
hibernate.search.indexing_strategy>manual
hibernate.search.default.worker.execution=async

#####
# Binary Storage Operations
#####
binary_storage.enabled=true

#####
# CORS Settings
#####
cors.enabled=true
cors.allowCredentials=true
# Supports multiple, comma separated allowed origin entries
# cors.allowed_
origin=http://localhost:8080,https://localhost:8080,https://fhirtest.uhn.ca
cors.allow_origin=*

#####
# Allowed Bundle Types for persistence (defaults are: COLLECTION, DOCUMENT, MESSAGE)
#####
#allowed_bundle_
types=COLLECTION, DOCUMENT, MESSAGE, TRANSACTION, TRANSACTIONRESPONSE, BATCH, BATCHRESPONSE, HISTORY, SEARCHSET

#####
# Subscriptions
#####

# Enable REST Hook Subscription Channel
subscription.resthook.enabled=true

# Enable Email Subscription Channel
subscription.email.enabled=false
email.enabled=false
email.from=some@test.com
email.host=
email.port=0
email.username=
email.password=
```

```
# Enable Websocket Subscription Channel
subscription.websocket.enabled=false
# Specify the JNDI data source name where HDR tables are created.
datasource.jndi_name=jdbc/HdrDataSource

# Enable/disable OAuth based API access (for easier dev/QA testing)
# Shouldn't use this property in production
oauth.access.enabled=false
#below properties are required for FHIR API access using oath token
oauth.token.issuer=https://dev-t9brtcqa.auth0.com/
oauth.token.audience=https://fhir-hdr.auth.com/api/v2/
#FHIR Server resource access settings
hdr.fhir.scopes=fhir.admin,fhir.users,fhir.users.restricted
#scopes and access rights
fhir.admin=read,create,update,delete
fhir.users=read,create
fhir.users.restricted=read
#note: add more if needed
#scopes and allowed fhir resources
fhir.admin.allowedapis=ALL
fhir.users.allowedapis=ResearchStudy
fhir.users.restricted.allowedapis=Patient, Observation, AllergyIntolerance,
Medication, Condition, Procedure, Immunization
#fhir.users.finance.allowedapis=Patient, Practitioner, Claim, ClaimResponse,
InsurancePlan
#fhir.users.clinical.allowedapis=
#note: add more if needed

#below properties that starts with audit.* are specific to auditing module of
oracle fhir infrastructure
audit.enabled=true
#audit storage type - for now FILE or DB. this needs to be extended for kafka etc
audit.datastore.type=FILE
audit.savemessagepayload.enabled=false
#auditing standard. this can be either a custom format or standard fhir
AuditEvent(ATNA equivalent) resource
audit.standard=CUSTOM
# Just to capture timing log. Only for Testing performace, not required to enable
this always.By defaults it is false.
response_timing_log_enabled=false
```

## 4.3.2 Log Configurations

HDR FHIR has several logging mechanisms that each serve a distinct purpose. These mechanisms are described in the table below. Oracle HDR FHIR uses the log4j logging framework to emit these logs. These logs are generated at runtime by all components of the FHIR. The location of the **log4j.properties** is: **<HDR\_DOMAIN>/config/fhir/log4j.properties**.



**Table 4–1 Log files**

<b>Log</b>	<b>File</b>	<b>Purpose</b>	<b>Retention</b>
Application log	<HDR_ DOMAIN>/logs/ hdr-fhir.log	Application Logging is a traditional file-based log of events and internal processing details of Oracle HDR FHIR. These logs are useful for troubleshooting. Application logs can be enabled and disabled at runtime by modifying the log4j properties file.	Logs are rotated and compressed on a Time basis, although this can be configured using the log4j.properties file.
Audit Log	<HDR_ DOMAIN>/logs/ audit-hdr-fhir.log	The audit log is intended to record actions taken by users. This log can be enabled or disabled using “audit.enabled” property defined in the hdr_ fhir.properties file.	Logs are rotated and compressed on a Time basis, although this can be configured using log4j.properties file.



---

---

## Using the OAuth 2.0 protected API

HDR FHIR offers a suite of REST APIs implemented per HL7 FHIR specification and secured using the OAuth 2.0 security framework. This article outlines the steps needed for clients/admin users to obtain a secure access token from HDR's OAuth Server and use the access token to invoke the HDR FHIR REST APIs.

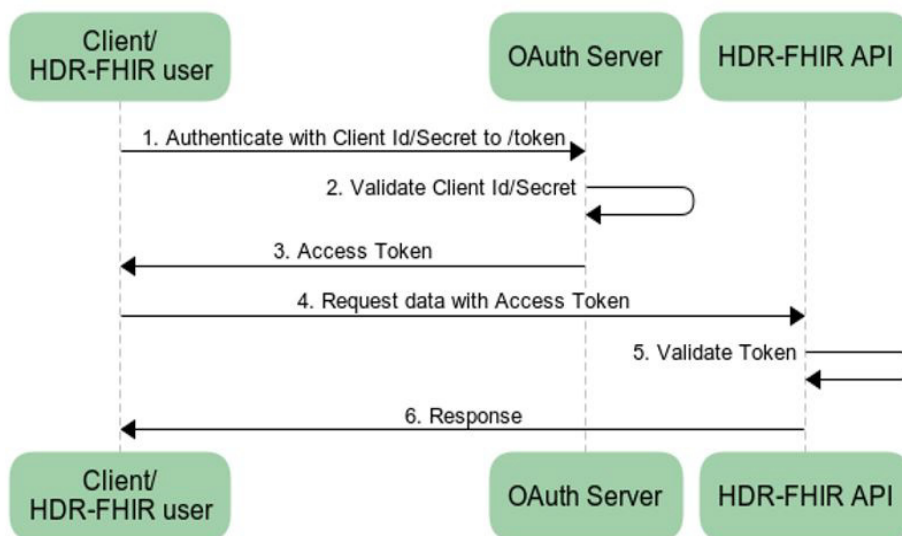
### 5.1 Prerequisites

Prerequisites for using the OAuth protected API are as follows:

- HDR FHIR is successfully registered with an OAuth Server as a Resource Server (that is protecting its endpoints).
- A client representing the HDR FHIR API admin user has been registered with OAuth Server as an OAuth Client and is authorized to invoke HDR FHIR APIs.

### 5.2 How It Works

Figure 5–1 Access Token Process



1. Client application or user authenticates with the OAuth Server (at say, the `/ms_oauth/oauth2/endpoints/tokens` endpoint) using the client ID and secret. The client ID and secret would have been obtained at the time of registering the OAuth client with OAuth Server.
2. OAuth Server validates the client ID and secret.
3. OAuth Server responds with an Access Token.
4. Client application or user uses the Access Token to call an HDR FHIR API.
5. HDR FHIR server intercepts the request and validates the Access Token.
6. HDR FHIR API responds with requested data.

## 5.3 Obtaining the Access Token from the OAuth Server

The client/user can ask the OAuth Server for tokens for any of the authorized applications by issuing the following API call:

```
curl --request POST \  
  --url https://example.oauthserver.com/ms_oauth/oauth2/endpoints/tokens \  
  --header 'content-type: application/json' \  
  --data '{"client_id":"CqwUDq2VQ6AH416sf7n42CZ2rNyElkDW","client_\  
secret":"iA6bJ9OQ-tMWhVNUZylx6Km1_\  
9tMuxVyKC4xNfWtPye72MjXyC3f1GJ38ttQ0oH9","audience":"hdr_fhir_api","grant_\  
type":"client_credentials"}'
```

In this example, `client_id` and `client_secret` are assigned random representative values. You should change these values with the actual client Id and secret, obtained after registering the client with OAuth Server.

```
{  
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsI.....N7KT4ig",  
  "token_type": "Bearer",  
  "expires_in":600  
}
```

You can now extract the `access_token` property value from the response to make authorized requests to your API.

## 5.4 Calling the HDR FHIR API with an Access Token

You can use this bearer token with an Authorization Header in your request to obtain authorized access to the HDR FHIR API.

```
curl --request GET \  
  --url http://<SERVER BASE URL>/fhir/Medication \  
  --header 'accept: application/json' \  
  --header 'authorization: Bearer  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsI.....N7KT4ig'
```

## 5.5 Error Messages

Here is a list of a few common OAuth-related error messages that can be thrown by HDR FHIR APIs and the associated remediation steps.

**Table 5-1 OAuth-Related Error Messages**

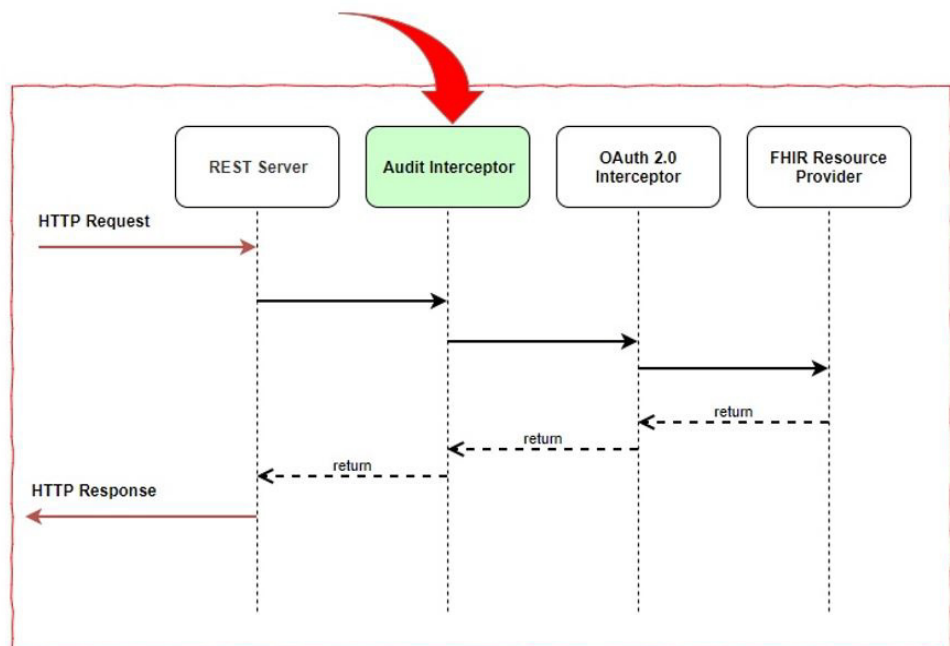
HTTP Status Code	Message	Meaning	Remediation
401	BAD_TOKEN: Invalid Algorithm. Algorithm is empty or not supported.	Signature algorithm is empty or not supported by the FHIR server.	Recommended algorithm is RS256. Make sure JWT header contains - "alg": "RS256".
200, 201	--	Success.	Authentication was successful. Operation was successful.
401	BAD_TOKEN: Invalid JWT token. Bad claims. Expired JWT	Unauthorized - expired OAuth token sent in request.	Current access token has expired. Obtain a fresh access token from OAuth Server and use it.
401	BAD_TOKEN: Invalid JWT token. Token is null or empty.	Unauthorized - no OAuth token sent in request.	Obtain a valid access token from OAuth Server. Pass it in request as bearer token in HTTP Auth header.
401	<Other error messages that start with "BAD_TOKEN: Invalid JWT token" >	Unauthorized - reason to be investigated.	Contact HDR administrator with the error message for further assistance.
401	BAD_TOKEN: Invalid JWT token. Bad claims. Invalid 'aud' attribute. Expected audience '<correct_audience>' does not exist in audience '<incorrect_audience>'	Unauthorized - token sent has incorrect audience value specified.	Ensure that you are using a correct audience value while requesting access token from OAuth Server.



This module is responsible for collecting and storing audit data from incoming REST request and response. Key details such as user id, IP address, resource name, HTTP request type and request URL etc. are collected from the incoming request and stored in a secure location. Audit records can be stored either in a database table or in a file.

## 6.1 Audit Interceptor Execution Flow

Figure 6-1 Audit Interceptor Execution Flow



The audit module (rendered in green) design follows an interceptor pattern as shown in the above flow diagram. Here, incoming and outgoing REST API transactions are intercepted for extracting audit data elements.

Once the data is extracted, audit information goes to either a database table or a file (depending on storage settings defined in the FHIR server configuration file).

## 6.2 Audit Record Format

Audit record data format is as shown below.

Audit data element	Description
AUDIT_ID	Unique identifier for audit record
USER_ID	User ID
RESOURCE_NAME	FHIR resource name
HTTP_REQ_TYPE	HTTP request type - GET, POST, and so forth
REQUEST_URL	Incoming request URL
HTTP_RES_CODE	HTTP response code - 200, 201, 500, ...
SOURCE_IP_ADDRESS	Source system IP address
PROCESSING_TIME_MILLIS	Time taken to complete REST request
REQUEST_PAYLOAD	Payload
RESPONSE_PAYLOAD	Response payload
EVENT_TIMESTAMP	Timestamp
ATNA_AUDITEVENT	Audit record in the form of AuditEvent json

## 6.3 Settings

Audit service functionality can be controlled using a configuration file. The file is located at <HDR\_DOMAIN>/config/fhir/**hdr\_fhir.properties**.

For example, if there is a requirement to store message payload as part of an audit record, change “audit.savemessagepayload.enabled” property to true. Other important entries in the properties file is as shown below.

```
#audit enabled - true or false
```

```
audit.enabled=true
```

```
#audit storage type - FILE or DB
```

```
audit.datastore.type=FILE
```

If 'FILE' is selected as the storage type, audit data goes to a file named **audit-hdr-fhir.log**.

If 'DB' is selected as the storage type, audit data goes to a table called **OHF\_HDR\_FHIR\_AUDIT**. Refer to the FHIR eTRM document for more information about the Audit table.

```
#collect request/response payload message - true or false
```

```
audit.savemessagepayload.enabled=false
```



---

---

## FHIR Command-Line Utility

The HDR FHIR command-line interface (CLI) tool is a standalone command-line tool distributed with HDR 8.1 that contains a number of commands to ingest terminology data provided by the LOINC and SNOMED organizations.

The current version of the HDR FHIR command-line interface tool supports LOINC and SNOMED terminology data to be loaded into the HDR FHIR repository.

The tool is distributed in the form of zip file (**hdr-fhir-cli-app-8.1.0-SNAPSHOT.zip**) and is located in the `<HDR_HOME>/weblogic/hdrfhircli/` directory when the HDR 8.1 installation is complete.

Extract it into a directory where you will keep it, and add this directory to your path.

The zip file **hdr-fhir-cli-app-8.1.0-SNAPSHOT.zip** contains the following files:

File	Description
<b>lib</b>	All runtime executable jar file.
<b>hdr-fhir-cli.sh</b>	Linux script to manage the command line tool.

### 7.1 Prerequisites

The following prerequisites are required to use the command-line utility:

- Install the JDK 1.8
- Set the environment variable `JAVA_HOME`
- Extract the **hdr-fhir-cli-app-8.1.0-SNAPSHOT.zip** file into the `<HDR_HOME>/weblogic/hdrfhircli` directory
- Set the environment variable `CLI_HOME` to the location where **hdr-fhir-cli-app-8.1.0-SNAPSHOT.zip** file is extracted

### 7.2 Commands

The HDR FHIR command line tool has a number of supported functions, called commands. Each command has a name and a set of supported arguments.

You can see a list of supported commands by simply executing the below command. For example:

```
$>${CLI_HOME}/hdr-fhir-cli.sh
```

Usage:

```
hdr-fhir-cli {command} [options]
```

Commands:

- `export-conceptmap-to-csv` - Exports a specific ConceptMap resource to a CSV file.
  
- `import-csv-to-conceptmap` - Imports a CSV file to a ConceptMap resource.
  
- `upload-definitions` - Uploads the conformance resources (StructureDefinition and ValueSet) from the official FHIR definitions.
- `upload-examples` - Downloads the resource example pack from the HL7.org FHIR specification website, and uploads all of the example resources to a given server.
  
- `upload-terminology` - Uploads a terminology package (e.g. a SNOMED CT ZIP file) to a server, using the `$upload-external-code-system` operation.

You can also see the list of supported arguments for a given command by issuing **command help [commandname]**. For example:

```
$>${CLI_HOME}/hdr-fhir-cli.sh help upload-terminology
```

Usage:

```
hdr-fhir-cli upload-terminology [options]
```

Uploads a terminology package (e.g. a SNOMED CT ZIP file) to a server, using the `$upload-external-code-system` operation.

Options:

- `-d,--data <arg>` Local file to use to upload (can be a raw file or a ZIP containing the raw file)
- `-l,--logging` If specified, verbose logging will be used.
- `-t,--target <target>` Base URL for the target server (e.g. " http://localhost:7001/oracle-fhir-server/fhir").
- `-u,--url <arg>` The code system URL associated with this upload (e.g. http://snomed.info/sct)
- `-v,--fhir-version <version>` The FHIR version being used. Valid values: r4

## 7.2.1 export-conceptmap-to-csv

The **export-conceptmap-to-csv** command can be used to export a ConceptMap resource as a CSV file of terminology mappings.

The first row of the CSV file will include the following headers:

- `SOURCE_CODE_SYSTEM` - ConceptMap.group.source
- `SOURCE_CODE_SYSTEM_VERSION` - ConceptMap.group.sourceVersion
- `TARGET_CODE_SYSTEM` - ConceptMap.group.target
- `TARGET_CODE_SYSTEM_VERSION` - ConceptMap.group.targetVersion
- `SOURCE_CODE` - ConceptMap.group.element.code
- `SOURCE_DISPLAY` - ConceptMap.group.element.display
- `TARGET_CODE` - ConceptMap.group.element.target.code
- `TARGET_DISPLAY` - ConceptMap.group.element.target.display
- `EQUIVALENCE` - ConceptMap.group.element.target.equivalence (ConceptMapEquivalence)
- `COMMENT` - ConceptMap.group.element.target.comment

Usage:

```
hdr-fhir-cli export-conceptmap-to-csv [options]
```

Exports a specific ConceptMap resource to a CSV file.

Options:

---

<b>-f,--filename &lt;filename&gt;</b>	The path and filename of the CSV file to be exported ( <b>./output.csv</b> ).
<b>-l,--logging</b>	If specified, verbose logging will be used.
<b>-t,--target &lt;target&gt;</b>	Base URL for the target server (e.g. <b>"http://localhost:7001/oracle-fhir-server/fhir"</b> ).
<b>u,--url &lt;url&gt;</b>	The URL of the ConceptMap resource to be exported (i.e. ConceptMap.url).
<b>-v,--fhir-version &lt;version&gt;</b>	The FHIR version being used. Valid values: <b>r4</b> .

---

These terminology mappings could then be exported with the following command:

```
./hdr-fhir-cli.sh export-conceptmap-to-csv --fhir-version R4 -t
http://localhost:8080//oracle-fhir-server/fhir -u
http://hl7.org/fhir/ConceptMap/cm-administrative-gender-v2 -f /u01/output.csv
```

## 7.2.2 import-csv-to-conceptmap

The **import-csv-to-conceptmap** command can be used to import a CSV file of terminology mappings and store it as a ConceptMap resource.

The first row of the CSV file is expected to include the following headers:

```
SOURCE_CODE_SYSTEM - ConceptMap.group.source
SOURCE_CODE_SYSTEM_VERSION - ConceptMap.group.sourceVersion
TARGET_CODE_SYSTEM - ConceptMap.group.target
TARGET_CODE_SYSTEM_VERSION - ConceptMap.group.targetVersion
SOURCE_CODE - ConceptMap.group.element.code
SOURCE_DISPLAY - ConceptMap.group.element.display
TARGET_CODE - ConceptMap.group.element.target.code
TARGET_DISPLAY - ConceptMap.group.element.target.display
EQUIVALENCE - ConceptMap.group.element.target.equivalence
(ConceptMapEquivalence)
COMMENT - ConceptMap.group.element.target.comment
```

An example CSV file that describes the mapping of FHIR to HL7v2 for Administrative Gender would appear as follows:

```
"SOURCE_CODE_SYSTEM", "SOURCE_CODE_SYSTEM_VERSION", "TARGET_CODE_SYSTEM", "TARGET_
CODE_SYSTEM_VERSION", "SOURCE_CODE", "SOURCE_DISPLAY", "TARGET_CODE", "TARGET_
DISPLAY", "EQUIVALENCE", "COMMENT"
"http://hl7.org/fhir/administrative-gender", "", "http://hl7.org/fhir/v2/0001", "", "m
ale", "Male", "M", "Male", "equal", ""
```

Usage:

```
hdr-fhir-cli import-csv-to-conceptmap [options]
```

Imports a CSV file to a ConceptMap resource.

Options:

---

<b>-f,--filename &lt;filename&gt;</b>	The path and filename of the CSV file to be imported (for example, <b>./input.csv</b> ).
---------------------------------------	------------------------------------------------------------------------------------------

---

---

<b>-i,--input &lt;input&gt;</b>	The source value set of the ConceptMap to be imported (i.e. ConceptMap.sourceUri).
<b>-l,--logging</b>	If specified, verbose logging will be used.
<b>-o,--output &lt;output&gt;</b>	The target value set of the ConceptMap to be imported (i.e. ConceptMap.targetUri).
<b>-t,--target &lt;target&gt;</b>	Base URL for the target server (e.g. <b>"http://localhost:7001/oracle-fhir-server/fhir"</b> ).
<b>-u,--url &lt;url&gt;</b>	The URL of the ConceptMap resource to be imported/exported (i.e. ConceptMap.url).
<b>-v,--fhir-version &lt;version&gt;</b>	The FHIR version being used. Valid values: <b>r4</b> .

---

These terminology mappings could then be imported with the following command:

```
./hdr-fhir-cli.sh import-csv-to-conceptmap --fhir-version R4 -t
http://localhost:8080//oracle-fhir-server/fhir -u
http://hl7.org/fhir/ConceptMap/cm-administrative-gender-v2 -i
http://hl7.org/fhir/ValueSet/administrative-gender -o
http://hl7.org/fhir/ValueSet/v2-0001 -f /u01/sampleInputFile.csv
```

### 7.2.3 upload-definitions and upload-examples

The **upload-definitions** command uploads the conformance resources (StructureDefinition and ValueSet) from the official FHIR definitions.

The **upload-examples** command uploads the example resources from the official FHIR definitions.

Usage:

```
hdr-fhir-cli upload-definitions/upload-examples [options]
```

The conformance rules are available at <https://www.hl7.org/fhir/conformance-rules.html>.

Options:

---

<b>-e,--exclude &lt;arg&gt;</b>	Exclude uploading the given resources, such as <b>"-e dicom-dcim,foo"</b> .
<b>-t,--target &lt;arg&gt;</b>	Base URL for the target server (such as <b>"http://localhost:7001/oracle-fhir-server/fhir"</b> ).
<b>-v,--fhir-version &lt;version&gt;</b>	The FHIR version being used. Valid values: <b>r4</b> .

---

Command usage:

```
./hdr-fhir-cli.sh upload-definitions --fhir-version R4 -t
http://localhost:8181/baseR4 "-e dicom-dcim,foo"

./hdr-fhir-cli.sh upload-examples --fhir-version R4 -t
http://localhost:8181/baseR4 "-e dicom-dcim,foo"
```

### 7.2.4 upload-terminology

The HDR FHIR server provides a terminology server, which supports ingestion of terminology data provided by LOINC and SNOMED. For more information on obtaining the above terminology data refer to the respective websites.

The HDR FHIR server provides a repository for terminology content used across the HDR platform, and an API suite to access the content.

The server provides a mechanism for ingestion of the terminology data via the **upload-terminology** command of the CLI tool. This command supports only LOINC and SNOMED terminologies in the current release.

---

**Note:** The path and exact filename of the terminology files will likely need to be adjusted for your local disk structure.

---

Usage:

**hdr-fhir-cli upload-terminology [options]**

Uploads a terminology package (such as a SNOMED CT ZIP file) to a server, using the \$upload-external-code-system operation.

Options:

---

<b>-d,--data &lt;arg&gt;</b>	Local file to use to upload (can be a raw file or a ZIP containing the raw file).
<b>-l,--logging</b>	If specified, verbose logging will be used.
<b>-t,--target &lt;target&gt;</b>	Base URL for the target server (such as " <a href="http://localhost:7001/oracle-fhir-server/fhir">http://localhost:7001/oracle-fhir-server/fhir</a> ").
<b>-u,--url &lt;arg&gt;</b>	The code system URL associated with this upload (such as <a href="http://snomed.info/sct">http://snomed.info/sct</a> ).
<b>-v,--fhir-version &lt;version&gt;</b>	The FHIR version being used. Valid values: <b>r4</b> .

---

Command usage:

```
$>${CLI_HOME}$. /hdr-fhir-cli.sh upload-terminology -d /scratch/fhir/Loinc_2.65.zip
-d /scratch/fhir/loincupload.properties --fhir-version R4 -t
http://localhost:8080//oracle-fhir-server/fhir -u http://loinc.org
```



---

---

## Working with FHIR REST APIs

Oracle Healthcare Data Repository 8.1-FHIR offers a suite of REST APIs implemented as per the HL7 FHIR specification and is secured using the OAuth 2.0 security framework. For more information, refer to the HDR FHIR REST API documentation.

