WRITING INTEGRATION CONNECTORS

Mandy Chessell CBE FREng
Egeria Open Source Project Lead
# Egeria’s webinar series

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Session Title</th>
<th>Description</th>
<th>Speaker</th>
</tr>
</thead>
</table>
| 7th Feb 2022 | 16:00 UTC  | Using an integration connector   | Automated metadata capture and distribution is the only way to ensure accuracy and consistency of metadata in your digital landscape. This webinar uses example scenarios to show how Egeria’s integration daemon manages integration connectors to enable:  
- dynamic cataloguing of data files, documents, databases, events and APIs  
- distribution and synchronization of technical metadata between data platforms.  
- exchange of metadata between metadata repositories such as data catalogs and CMDIIs.  
- notification to stewards when exceptions are detected.  
- configuring security managers such as Apache Ranger.  
- onboarding organization data - people, roles, userIDs, team structures into the open metadata ecosystem and maintaining access information in LDAP.  
- capture and exchange of lineage metadata.  
All of the metadata captured is managed and exchanged using Egeria’s open metadata schemas and benefits from Egeria’s metadata governance capabilities. | Mandy Chessell|
| 7th - 15th Mar 2022 | 14:00 UTC | How to build an integration connector | This session covers how to extend Egeria's automated cataloguing to include metadata from a new technology. It describes how automated cataloguing works and the role of the integration connector. It covers the design of the integration connector using examples to illustrate the different approaches and their benefits and challenges. It shows how to set up a project for a new connector, how to build and package it and finally it shows the new connector running in Egeria.  
Zoom Conference [https://zoom.us/j/623629111](https://zoom.us/j/623629111) | Mandy Chessell|
| 4th Apr 2022  | 10:00 UTC  | Using a repository connector      | This session covers how to use Repository Connectors to connect technologies into Egeria, focussing on XTD6 (formerly known as Crox). Ever wanted to know what the state of your metadata was at some specific time in the past? This session will introduce the XTD6 open metadata repository that supports these historical metadata queries.  
Zoom Conference [https://zoom.us/j/623629111](https://zoom.us/j/623629111) | Chris Grote |
Integrated cataloguing

- Callie has a database server that she uses to analyze relational data.
- She creates a new sandbox for each type of analysis.
- However, she often forgets to catalog her sandboxes.
The integration daemon

- A type of OMAG Server

- Metadata extraction, capture and delivery
The integration connector
Open Metadata Integration Services (OMIS)

The integration services available today are:

- **API Integrator** - provides cataloguing for APIs.
- **Analytics Integrator** - provides cataloguing for Analytics tools.
- **Catalog Integrator** - provides a two-way synchronization for data catalogs.
- **Database Integrator** - provides metadata extraction from relational databases.
- **Display Integrator** - provides metadata extraction from systems that provide user displays and forms to capture new data values.
- **Files Integrator** - collects metadata about files stored in a filesystem or file manager.
- **Infrastructure Integrator** - supports the extraction of metadata from IT infrastructure artifacts as well as the use of metadata to maintain IT infrastructure artifacts.
- **Lineage Integrator** - collects metadata about processes, their internal logic and the data assets they work with.
- **Organization Integrator** - imports details of an organization's structure - such as teams and departments.
- **Security Integrator** - distributes security properties to access control enforcement points.
- **Stewardship Integrator** - exchanges requests for stewardship action (and results) with a human task manager.
- **Topic Integrator** - provides cataloguing of topics and event schema for event brokers.
Inside the integration daemon

- The purpose of the integration daemon and its integration services is to minimise the effort required to integrate a third-party technology into the open metadata ecosystem. They handle:
  - Management of configuration - including user security information.
  - Starting and stopping of your integration logic.
  - Thread management and polling.
  - Access to the open metadata repositories for query and maintenance of open metadata.
  - Ability to write to audit log and maintain measurements for performance metrics.
  - Metadata provenance.

- This means you can focus on interacting with the third-party technology and mapping its metadata to open metadata in your integration connector.
Supporting the metadata needs of different technologies

**OMIS**

```
// creates a new file asset and links it to the folder structure implied in the path name. If the folder structure is not
cataloged already, this is created automatically using the createFolderStructureInCatalog() method.

// For example, a pathname of "one/two/three/myFile.txt" potentially creates 2 new folder assets, one called "one",
// the next called "one/two" and the last one called "one/two/three" plus a file asset called
// "one/two/three/myFile.txt".

public List<String> addDataFileToCatalog(String dataFileProperties, String connectorProviderName) throws 

```

**OMAS**

```
// creates a new file asset and links it to the folder structure implied in the path name. If the folder structure is not
cataloged already, this is created automatically using the createFolderStructureInCatalog() method.

// For example, a pathname of "one/two/three/myFile.txt" potentially creates 2 new folder assets, one called "one",
// the next called "one/two" and the last one called "one/two/three" plus a file asset called
// "one/two/three/myFile.txt".

public List<String> addDataFileToCatalog(String dataFileProperties, String connectorProviderName) throws 

```
Integration Connector Implementation

IntegrationConnector

- initialize()
- setAuditLog()
- setConnectorName()
- initializeEmbeddedConnectors()
- setContext()
- start()
- engage()
- refresh()
- disconnect()

For saving connector instance identifier and connection object. Called from the ConnectorBroker.

Provides the logging destination. Provides the name of the connector for logging. Initializes embedded connectors

Set up the integration service specific context.

Indicates that the connector is completely configured and can begin processing. This call can be used to register with non-blocking services. For example, it can register a listener with the OMAS Out Topic with the context.

Used for blocking calls to wait for new metadata. It is called from its own thread if the connector is configured to have its own thread. It is recommended that the engage() method returns when each blocking call completes. The integration daemon will pause a second and then call engage() again. This pattern enables the calling thread to detect the shutdown of the integration daemon server.

Requests that the connector does a comparison of the metadata in the third party technology and open metadata repositories. Refresh is called when the integration connector first starts and then at intervals defined in the connector’s configuration as well as any external REST API calls to explicitly refresh the connector.

Free up any resources held since the connector is no longer needed.

Handled by base class
Integration Connector configuration

- The configuration provides the integration daemon with the information it needs to control the lifecycle and runtime support needed by the connector.
Integration Connector configuration

- Identifier used when registering with a listener
- Name used in messages
- User id for calling open metadata
- Connection for integration connector
- External source identifier for metadata provenance
- Rate at which refresh() method called
- The engage() method called if set to true
- Disables maintenance methods on open metadata
### Connector Provider

- Set up connector class name
- Build connector type
- Set up audit log component identifier

Configuration properties are added to the connection

![Connector Provider Class](https://egeria-project.org)
Using templates

- Governance metadata added through templates
Connections for integration connectors

- Connection
- ConnectorType: Implementation of the integration connector
- Endpoint: Location of the system to catalog (Optional)
Locating endpoints

- An integration connector can listen for events from open metadata to retrieve the endpoint for the third-party technology.
Connections for integration connectors

Accessed through the `embeddedConnectors` variable
Start method

1. Call super class
2. Extract configuration
3. Log start
4. Register listeners

```java
/**
 * Indicates that the connector is completely configured and can begin processing.
 * This call can be used to register with non-blocking services.
 * @throws ConnectorCheckedException if there is a problem within the connector.
 */
@override
public void start() throws ConnectorCheckedException {
    super.start();
    final String methodName = "start";
    /**
     * Extract the configuration
     */
    EndpointProperties endpoint = connectionProperties.getEndpoint();
    if (endpoint != null) {
        fileDirectoryName = endpoint.getAddress();
    }
    Map<String, Object> configurationProperties = connectionProperties.getConfigurationProperties();
    if (configurationProperties != null) {
        if (configurationProperties.containsKey(BasicFileMonitorIntegrationProviderBase.LOCK_FILE_MONITORING_ENABLED)) {
            allowLocking = true;
        }
        templateQualifiedName = configurationProperties.get(BasicFileMonitorIntegrationProviderBase.TEMPLATE_DIRECTORY_CONFIG_PARAMETER) + File.separator + methodName;
    }
    /**
     * Record the configuration
     */
    if (auditing != null) {
        auditing.logMessage(methodName,
            BasicFilesIntegrationConnectorsAuditCode.CONNECTOR_CONFIGURATION, getOperationDefinition().getName(),
            fileDirectoryName,
            Boolean.valueOf(allowLocking),
            templateQualifiedName);
    }
    /**
     * Start listening
     */
    this.initiateDirectoryMonitoring(this.getRootDirectoryFile(), methodName);
}
```
Register open metadata listener

```java
/**
 * Indicates that the connector is completely configured and can begin processing.
 * This call can be used to register with non-blocking services.
 * @throws ConnectorCheckedException there is a problem within the connector.
 * @Override
 * public void start() throws ConnectorCheckedException {
 *     super.start();
 *     final String methodName = "start";
 *     if (connectionProperties.getUserId() != null) {
 *         clientId = connectionProperties.getUserId();
 *     }
 *     myContext = super.getContext();
 *     try {
 *         myContext.registerListener(this);
 *         /*
 *         * Record the start
 *         */
 *         if (auditLog != null) {
 *             auditLog.logMessage(methodName,
 *                                 EgeriaInfrastructureConnectorAuditCode.CONNECTOR_START.getMessageDefinition(connectorName, clientId));
 *         }
 *     } catch (Exception error) {
 *         throw new ConnectorCheckedException(EgeriaInfrastructureConnectorErrorCode.UNEXPECTED_EXCEPTION.getMessageDefinition(connectorName,
 *                                                                         error.getClass().getName(),
 *                                                                         methodName,
 *                                                                         error.getMessage()));
 *     }
 * }
 */
```
Example solution

- Metadata synchronization can be inbound or outbound and this affects the design of your integration connector.
Refresh method: Third-party metadata source

Metadata source

Third Party Technology

retrieve each element catalogued in third party technology

Integration Connector

Sweep 1

Compare

Sweep 2

retrieve equivalent element

create or update element as needed

retrieve elements catalogued for third party technology

delete elements no longer defined in third party technology

Metadata destination

Context
Refresh method: Third-party metadata destination

**Metadata destination**

- Third Party Technology
  - Retrieve equivalent element
  - Create or update element as needed
  - Retrieve elements catalogued on third party technology
  - Delete elements no longer defined in open metadata

**Integration Connector**

- Compare

**Sweep 1**

- Retrieve each element catalogued in open metadata

**Sweep 2**

**Metadata source**

- Context
  - Retrieve equivalent element
Two sweeps

```java
/**
 * Requests that the connector does a comparison of the metadata in the third party technology and open metadata repositories.
 * Refresh is called when the integration connector first starts and then at intervals defined in the connector's configuration
 * as well as any external REST API calls to explicitly refresh the connector.
 * This method performs two sweeps. It first retrieves the files in the directory and validates that are in the
 * catalog - adding or updating them if necessary. The second sweep is to ensure that all of the assets catalogued
 * in this directory actually exist in the file system.
 * @throws ConnectorCheckedException there is a problem with the connector. It is not able to refresh the metadata.
 * @Override
 * public void refresh() throws ConnectorCheckedException
 * {
 *     final String methodName = "refresh";
 *     File directory = this.getRootDirectoryFile();
 *     if (directory != null)
 *     {
 *         /* Sweep one - cataloguing all files
 *         */
 *         File[] filesArray = directory.listFiles();
 *         if (filesArray != null)
 *         {
 *             for (File file : filesArray)
 *             {
 *                 if (file != null)
 *                 {
 *                     this.catalogFile(file, methodName);
 *                 }
 *             }
 *         }
 *         /* Sweep two - ensuring all catalogued files still exist. Notice that if the folder does not exist, it is
 *         * ignored. It will be dynamically created when a new file is added.
 *         */
 *         try
 *         {
 *             catch (Exception error)
 *             {
 *                 if (auditlog != null)
 *                 {
 *                     auditLog.logException(methodName,
 *                     BasicFilesIntegrationConnectorsAuditCode.UNEXPECTED_EXC_DATA_FILE_UPDATE.getMessageDefinition(error.getClass().getName(),
 *                     connectorName,
 *                     directory.getAbsolutePhysicalPath(),
 *                     error.getMessage()));
 *                     error;
 *                 }
 *                 throw new FileException(
 *                     BasicFilesIntegrationConnectorsErrCode.UNEXPECTED_EXC_DATA_FILE_UPDATE.getMessageDefinition(error.getClass().getName(),
 *                     connectorName,
 *                     directory.getAbsolutePhysicalPath(),
 *                     error.getMessage()),
 *                     error.getClass(),
 *                     methodName,
 *                     error,
 *                     directory.getAbsolutePath());
 *             }
 *         }
 *     }
 * }
 */
```
Refresh method - sweep 2

Archiving open metadata asset for file if the file no longer exists
Catalog file

Cataloguing file if it is not already in the catalog
Using templates

- Template qualified name supplied in connection configuration
- Integration connector looks it up to retrieve the unique identifier of the template
- The template GUID is used when the file is created

```
if (templateQualifiedName == null)
(...)
else 

    if (templateGUID == null)
    {
        DataFileElement templateElement = this.getContext().getByPathName(templateQualifiedName);
        if (templateElement != null)
        { 
            if ((templateElement.getElementHeader() != null) && (templateElement.getElementHeader().getGUID() != null))
            { 
                templateGUID = templateElement.getElementHeader().getGUID();
            }
        }
        else 
        {
            if (auditLog != null)
            {
                auditLog.logMessage(methodName,
                BasicFilesIntegrationConnectorsAuditCode.BAD_FILE_ELEMENT.getMessageDefinition(
                connectorName,
                templateElement.toString()));
            }
        }
    }
    else 
    {
        if (auditLog != null)
        {
            auditLog.logMessage(methodName,
            BasicFilesIntegrationConnectorsAuditCode.MISSING_TEMPLATE.getMessageDefinition(connectorName,
            templateQualifiedName));
        }
    }

    if (templateGUID != null)
    {
        TemplateProperties properties = new TemplateProperties();
        properties.setQualifiedName(file.getAbsolutePath());
        properties.setDisplayName(file.getName());
        properties.setNetworkAddress(file.getAbsolutePath());
        List<String> guids = this.getContext().addDataFileToCatalogFromTemplate(templateGUID, properties);
        if ((guids != null) && (!guids.isEmpty()) && (auditLog != null))
        {
            auditLog.logMessage(methodName,
            BasicFilesIntegrationConnectorsAuditCode.DATA_FILE_CREATED_FROM_TEMPLATE.getMessageDefinition(
            connectorName,
            properties.getQualifiedName(),
            guids.get(guids.size() - 1),
            templateQualifiedName,
            templateGUID));
        }
    }
```
Disconnect method

- Clear down resources
- Log shutdown

```java
/**
 * Shutdown file monitoring
 */
@throws ConnectorCheckedException something failed in the super class

@override
public void disconnect() throws ConnectorCheckedException {
    final String methodName = "disconnect";
    for (String fileName : monitors.keySet()) {
        this.stopDirectoryMonitoring(fileName, methodName);
    }
    if (auditLog != null)
        auditLog.logMessage(methodName,
            BasicFilesIntegrationConnectorsAuditCode.CONNECTOR_STOPPING.getMessageDefinition(connectorName));
    super.disconnect();
}
```
Metadata Provenance

- Identifies source of open metadata
- Controls who can update open metadata

<table>
<thead>
<tr>
<th>Integration Service</th>
<th>Method to control external source metadata provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytics Integrator OMIS</td>
<td>Call setAnalyticsIsHome() method to set toggle. Default is true.</td>
</tr>
<tr>
<td>API Integrator OMIS</td>
<td>Call setAPIManagerIsHome() method to set toggle. Default is true.</td>
</tr>
<tr>
<td>Catalog Integrator OMIS</td>
<td>Use assetManagerIsHome property on method calls.</td>
</tr>
<tr>
<td>Database Integrator OMIS</td>
<td>External source metadata provenance always enabled.</td>
</tr>
<tr>
<td>Display Integrator OMIS</td>
<td>Call setApplicationIsHome() to set toggle. Default is true.</td>
</tr>
<tr>
<td>Files Integrator OMIS</td>
<td>Local cohort metadata provenance is always enabled.</td>
</tr>
<tr>
<td>Infrastructure Integrator OMIS</td>
<td>Call setInfrastructureManagerIsHome() method to set toggle. Default is true.</td>
</tr>
<tr>
<td>Lineage Integrator OMIS</td>
<td>Use assetManagerIsHome property on method calls.</td>
</tr>
<tr>
<td>Organization Integrator OMIS</td>
<td>Local cohort metadata provenance is always enabled.</td>
</tr>
<tr>
<td>Search Integrator OMIS</td>
<td>Not applicable - outbound only</td>
</tr>
<tr>
<td>Security Integrator OMIS</td>
<td>Local cohort metadata provenance is always enabled.</td>
</tr>
<tr>
<td>Topic Integrator OMIS</td>
<td>Call setEventBrokerIsHome() method to set toggle. Default is true.</td>
</tr>
</tbody>
</table>
Inside the Lineage Integrator OMIS
Methods from the Lineage Integration OMIS context

```java
/** *
 * Register a listener to receive open lineage events.
 */

/** *
 * The listener is implemented by the integration connector. Once it is registered with the context, its processOpenLineageRunEvent() method is called each time an open lineage event is published to the Lineage Integrator OMIS.
 * @param listener listener to call
 */
public void registerListener(OpenLineageRunEventListener listener) {
    openLineageListenerManager.registerListener(listener);
}

/** *
 * Called each time an integration connector wishes to publish an open lineage run event. The event is formatted and passed to each of the registered open lineage event listeners.
 * @param payload payload to send for the event
 */
public void publishOpenLineageRunEvent(String payload) {
    openLineageListenerManager.publishOpenLineageRunEvent(payload);
}

/** *
 * Called each time an open lineage run event is published to the Lineage Integrator OMIS. The integration connector is able to work with the formatted event using the Egeria beans or refer to the open lineage run event using the supplied open lineage backend beans or another set of beans.
 * @param event bean for the event
 */
public void publishOpenLineageRunEvent(OpenLineageRunEvent event) {
    openLineageListenerManager.publishOpenLineageRunEvent(event);
}

/** *
 * Register for inbound events from the Asset Manager OMIS OutTopic
 */

/** *
 * Register a listener object that will be passed each of the events published by the Asset Manager OMIS.
 * @param listener listener object
 */
public void registerListener(AssetManagerRunEventListener listener) throws InvalidParameterException, ConnectionCheckedException, PropertyServerException, UnauthorizedException {
    openLineageListenerManager.registerListener(listener);
}
```

https://egeria-project.org
Examples of granularity challenge

External identifiers are supported by Catalog Integrator OMIS and Lineage Integrator OMIS
/**
 * Create a new metadata element to represent the root of an asset.
 * 
 * @param assetManagerIsHome ensure that only the asset manager can update this asset
 * @param assetExternalIdentifier unique identifier of the asset in the external asset manager
 * @param assetExternalIdentifierName name of property for the external identifier in the external asset manager
 * @param assetExternalIdentifierUsage optional usage description for the external identifier when calling the external asset manager
 * @param assetExternalIdentifierKeyPattern pattern for the external identifier within the external asset manager (default is LOCAL_KEY)
 * @param mappingProperties additional properties to help with the mapping of the elements in the external asset manager and open metadata
 * @param assetProperties properties to store
 * 
 * @return unique identifier of the new metadata element
 * 
 * @throws IllegalArgumentException one of the parameters is invalid
 * @throws UnauthorizedException the user is not authorized to issue this request
 * @throws PropertyServerException there is a problem reported in the open metadata server(s)
 */

public String createDataAsset( boolean assetManagerIsHome,
                          String assetExternalIdentifier,
                          String assetExternalIdentifierName,
                          String assetExternalIdentifierUsage,
                          KeyPattern assetExternalIdentifierKeyPattern,
                          Map<String, String> mappingProperties,
                          DataAssetProperties assetProperties) throws IllegalArgumentException,
                                  UnauthorizedException,
                                  PropertyServerException
Testing your connector

- Build and install jar in OMAG Server Platform’s lib directory
- Configure connector in integration daemon connected to a metadata access store.
Open forum

Questions?
## Egeria’s webinar series

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Session Title</th>
<th>Description</th>
<th>Speaker</th>
</tr>
</thead>
</table>
| 7th Feb   | 15:00 | Using an integration connector             | Automated metadata capture and distribution is the only way to ensure accuracy and consistency of metadata in your digital landscape. This webinar uses example scenarios to show how Egeria's integration daemon manages integration connectors to enable:  
- dynamic cataloging of data files, documents, databases, events and APIs  
- distribution and synchronization of technical metadata between data platforms.  
- exchange of metadata between metadata repositories such as data catalogs and CMDiBs.  
- notification to stewards when exceptions are detected.  
- configuring security managers such as Apache Ranger.  
- onboarding organization data - people, roles, userIDs, team structures into the open metadata ecosystem and maintaining access information in LDAP.  
- capture and exchange of lineage metadata.  
All of the metadata captured is managed and exchanged using Egeria’s open metadata schemas and benefits from Egeria’s metadata governance capabilities. | Mandy Chessell |
| 7th - 15th Mar | 14:00 | How to build an integration connector      | This session covers how to extend Egeria’s automated cataloguing to include metadata from a new technology. It describes how automated cataloguing works and the role of the integration connector. It covers the design of the integration connector using examples to illustrate the different approaches and their benefits and challenges. It shows how to set up a project for a new connector, how to build and package it and finally it shows the new connector running in Egeria. | Mandy Chessell |
| 4th Apr   | 15:00 | Using a repository connector                | This session covers how to use Repository Connectors to connect technologies into Egeria; focussing on XTDB (formerly known as crux).  
Ever wanted to know what the state of your metadata was at some specific time in the past? This session will introduce the XTDB open metadata repository that supports these historical metadata queries. | Chris Grote    |
Achievements

- 700 linked open metadata types demonstrating how the knowledge from many tools can be linked together.
- Open metadata repository interface proven for table, graph and hierarchical DB stores.
- Enterprise queries and replication across heterogeneous technologies
- Conformance test suite and mark
- Automated configuration of data virtualization technology and security as new data sets are added to a data lake
- Suite of persona-based labs and tutorial using Jupyter Notebooks.

- Virtual graph of metadata maintained across distributed heterogeneous metadata repositories.
- Frameworks, APIs and connectors for minimizing integration cost for different types of technologies
- Virtual repository explorer UI
- Instance based security
- Controlling visibility of assets through zones
- Scalable, secure platform configurable and customizable through connectors
- Purpose-based data access
- Metadata versioning and provenance
- Multi-tenant UI based on carbon
- W3C semantic standards pattern for data model exchange
- Automation of metadata acquisition through templates, daemons, discovery services and stewardship.
- Classification of assets
- Reference data management
- Multi-technology collaboration and feedback
- Multi-domain governance model
- Digital service lifecycle, from business design, development, devOps and use.
- Comprehensive open lineage services.
- Metadata deduplication