LF AI & DATA

The EGERIA Metadata Show

Egeria Webinar

WRITING INTEGRATION CONNECTORS

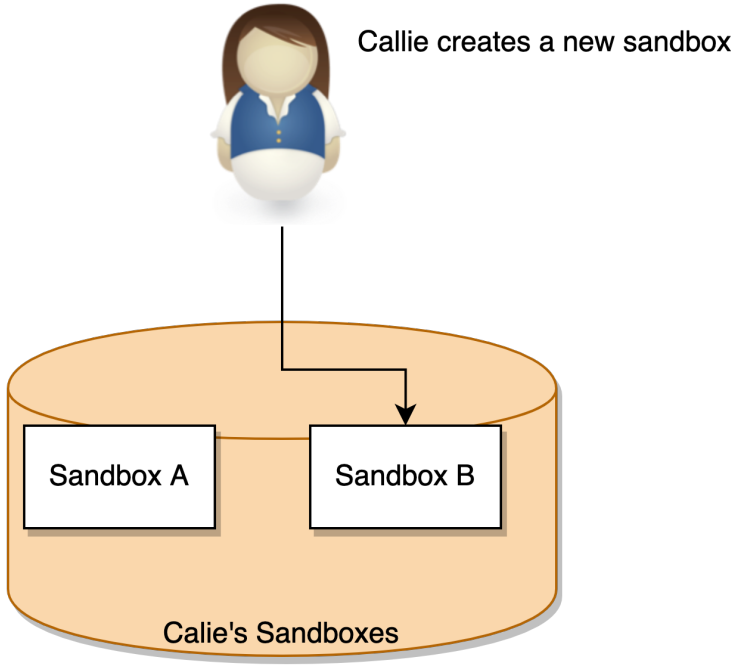
Mandy Chessell CBE FREng
Egeria Open Source Project Lead

Egeria's webinar series

7th February 2022	15:00 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 CMDBs. - 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 March 2022**	~~15:00~~ 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 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/523629111	**Mandy Chessell**
~~**4th**~~ **April 2022**	~~15:00~~ UTC	**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. Zoom Conference https://zoom.us/j/523629111	**Chris Grote**

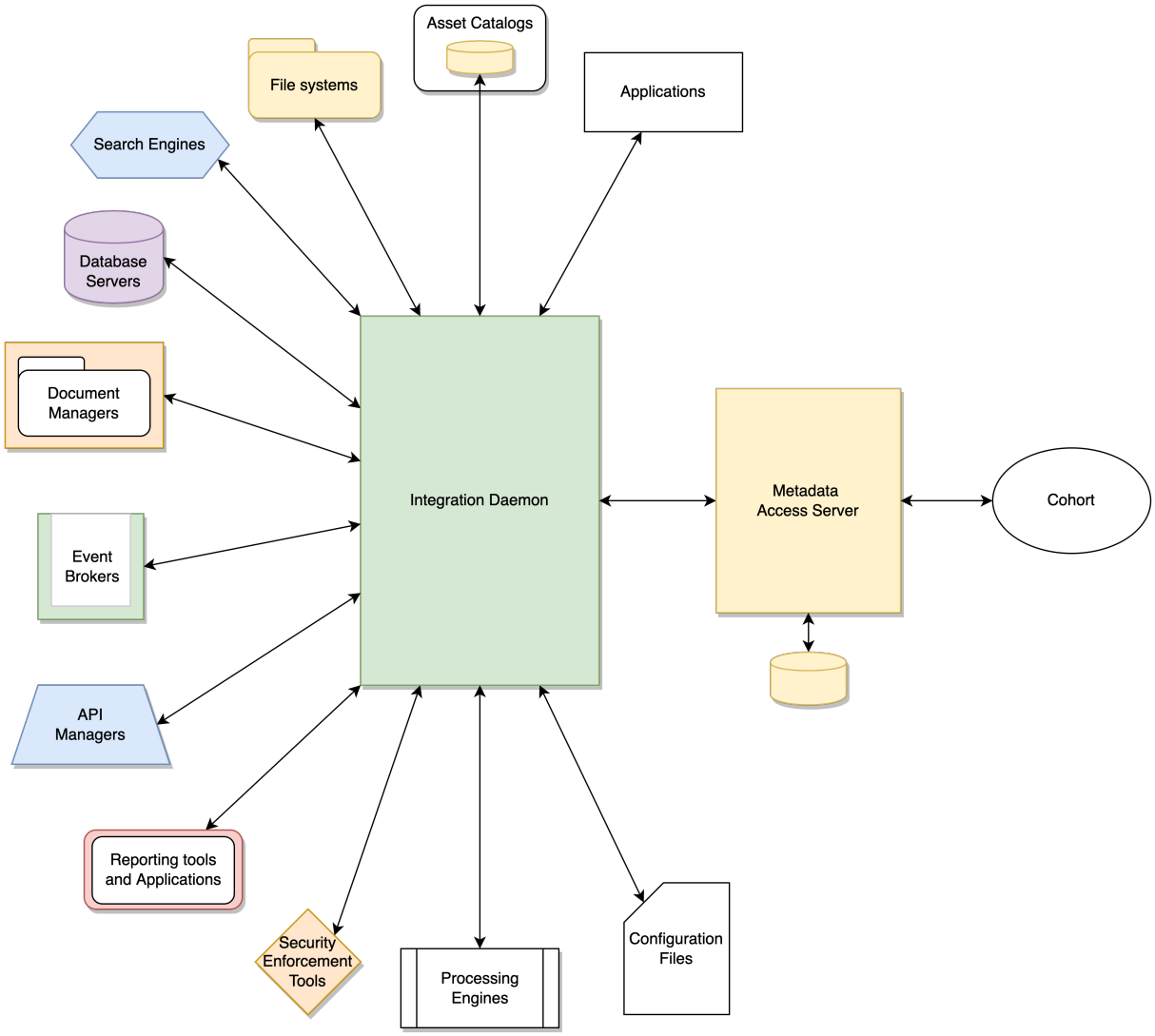
Integrated cataloguing

- 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.

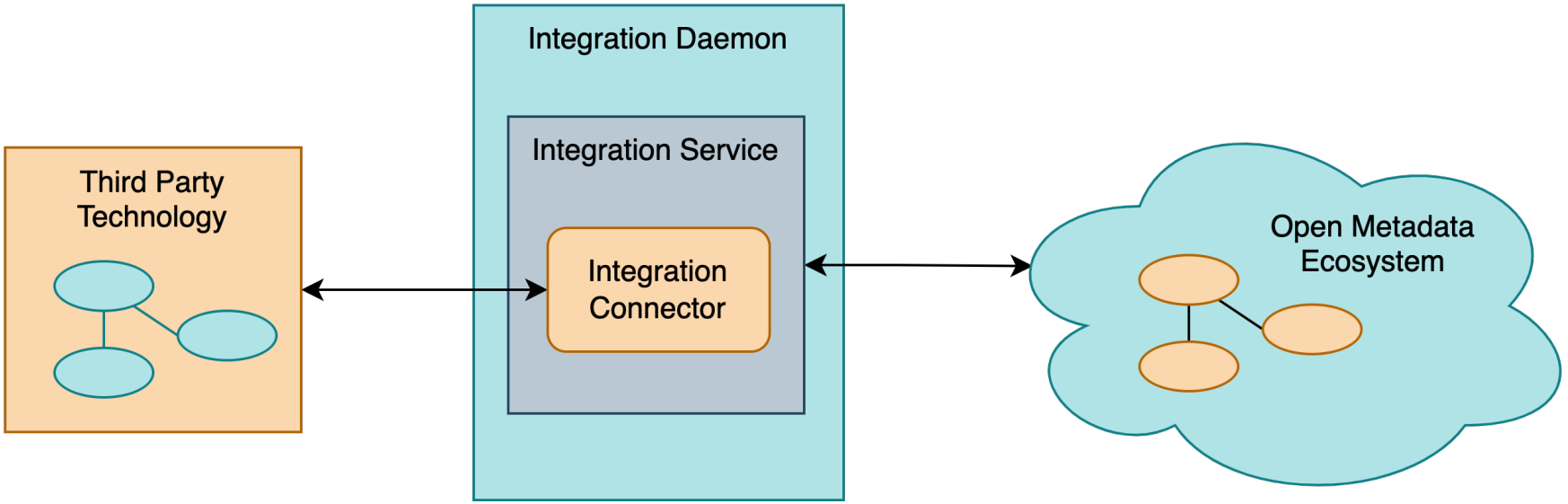


The integration daemon

- A type of OMAG Server
- Metadata extraction, capture and delivery



The integration connector



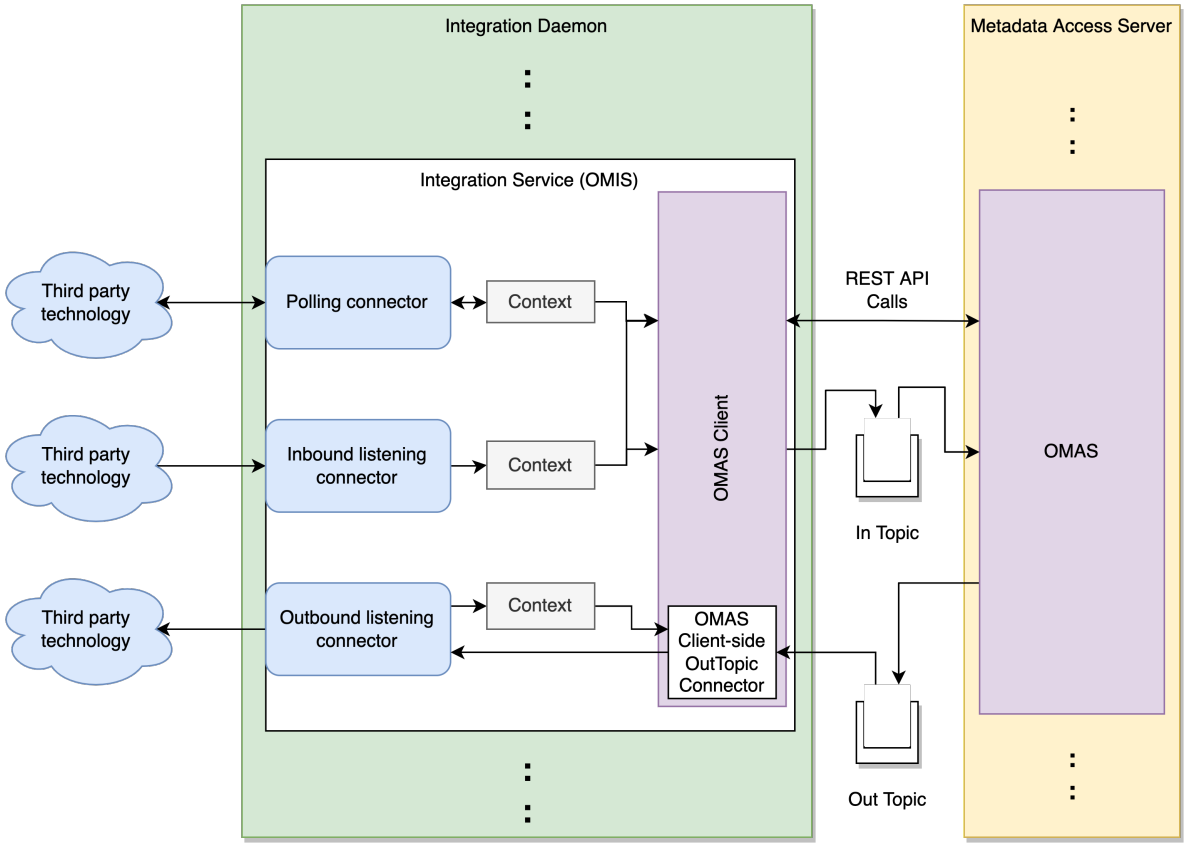
Open Metadata Integration Services (OMIS)

The integration services available today are:

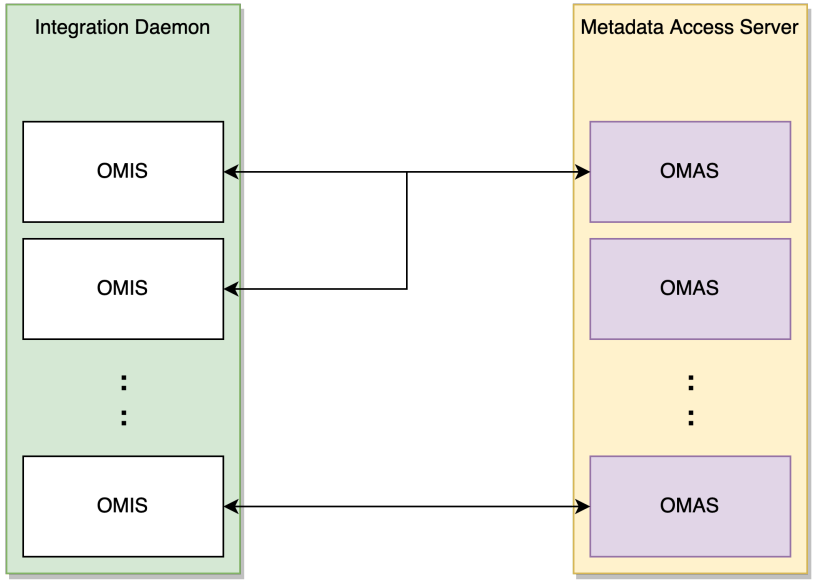
- API Integrator - provides cataloguing for APIs.
- Analytics Integrator - provides cataloguing for Analytics tools.
- Catalog Integrator - provides a two-way synchronization for data catalogs.
- Database Integrator - provides metadata extraction from relational databases.
- Display Integrator - provides 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

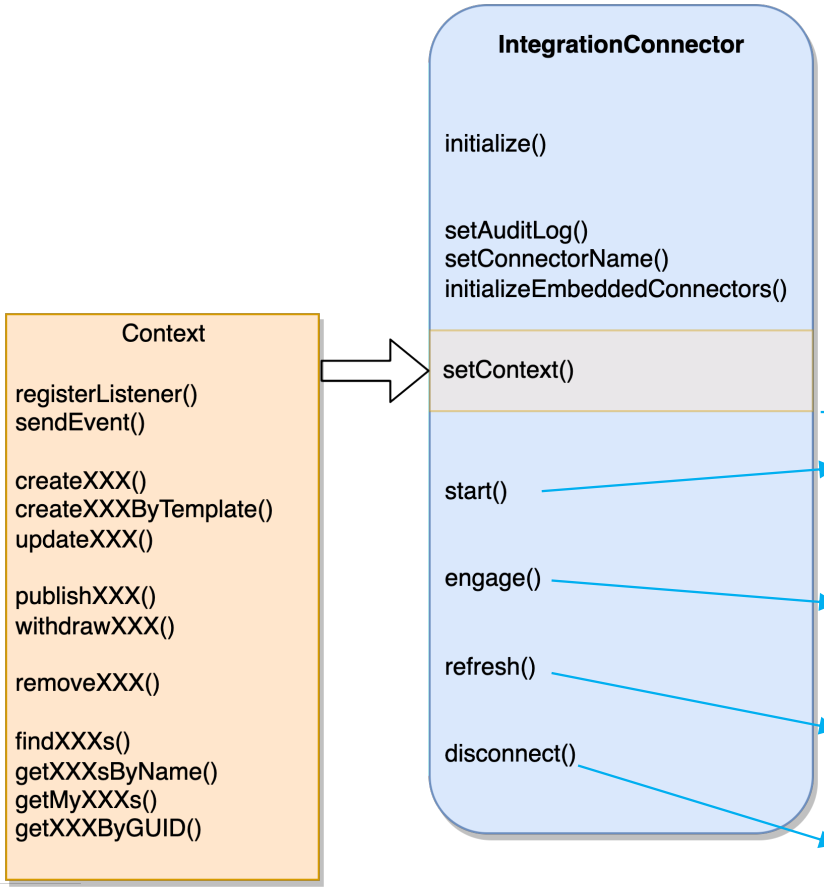
```
/**
 * Creates a new file asset and links it to the folder structure implied in the path name.  If the folder
 * structure is not catalogued already, this is created automatically using the createFolderStructureInCatalog() method.
 * For example, a pathName of "one/two/three/MyFile.txt" potentially creates 3 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".
 *
 * @param dataFileProperties details of the data file to add to the catalog as an asset
 * @param connectorProviderName class name of connector provider for connector to access this asset
 *
 * @return list of GUIDs from the top level to the root of the pathname
 *
 * @throws InvalidParameterException one of the parameters is null or invalid
 * @throws PropertyServerException problem accessing property server
 * @throws UserNotAuthorizedException security access problem
 */
public List<String> addDataFileToCatalog(DataFileProperties dataFileProperties,
                                          String            connectorProviderName) throws InvalidParameterException,
                                                                                          UserNotAuthorizedException,
                                                                                          PropertyServerException
```

OMAS

```
/**
 * Creates a new file asset and links it to the folder structure implied in the path name.  If the folder
 * structure is not catalogued already, this is created automatically using the createFolderStructureInCatalog() method.
 * For example, a pathName of "one/two/three/MyFile.txt" potentially creates 3 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".
 *
 * @param userId calling user
 * @param fileManagerCapabilityGUID unique identifier of the software server capability representing an owning external file manager or null
 * @param fileManagerCapabilityName unique name of the software server capability representing an owning external file manager or null
 * @param dataFileProperties details of the data file to add to the catalog as an asset
 * @param connectorProviderName class name of connector provider for connector to access this asset
 *
 * @return list of asset GUIDs from the top level to the root of the pathname
 *
 * @throws InvalidParameterException one of the parameters is null or invalid
 * @throws PropertyServerException problem accessing property server
 * @throws UserNotAuthorizedException security access problem
 */
List<String> addDataFileToCatalog(String            userId,
                                  String            fileManagerCapabilityGUID,
                                  String            fileManagerCapabilityName,
                                  DataFileProperties dataFileProperties,
                                  String            connectorProviderName) throws InvalidParameterException,
                                                                                  UserNotAuthorizedException,
                                                                                  PropertyServerException;
```

EGERIA

Integration Connector Implementation



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

Handled by base class

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.

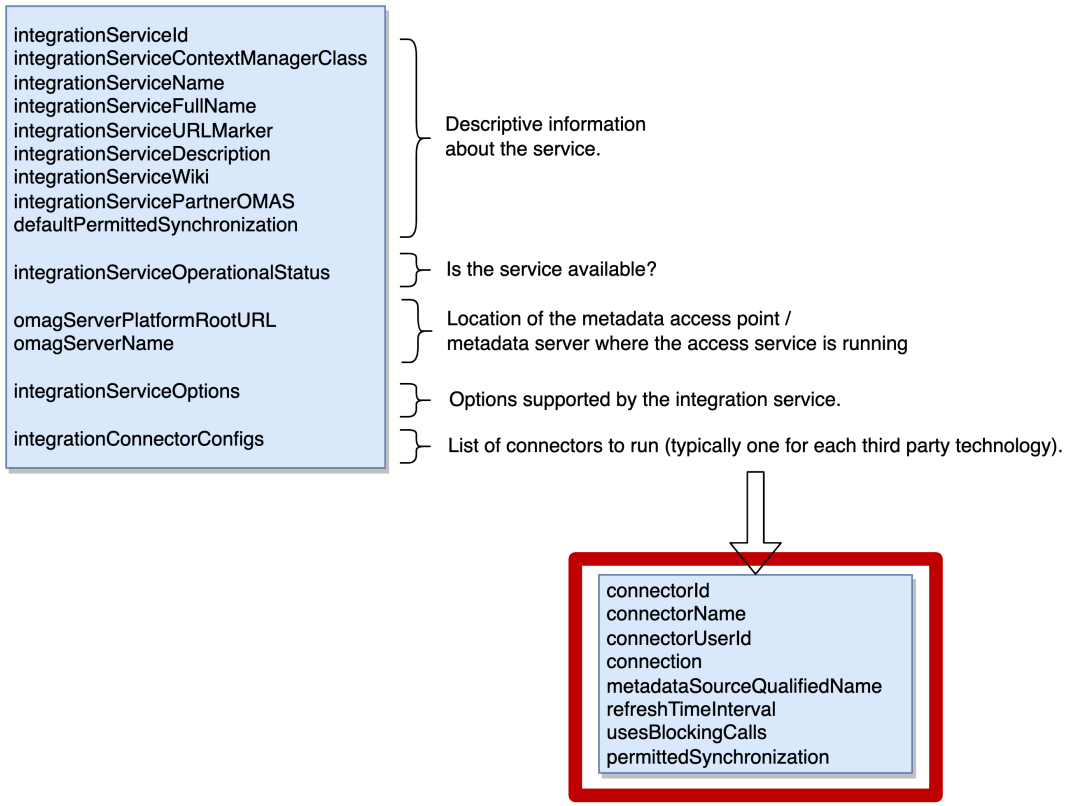
Used for blocking calls to wait for new metadata. It is called from its own thread iff the connector is configured to have its own thread. It is recommended that the engage() method returns when each blocking call completes. The integration daemon will pause a second and then call engage() again. This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

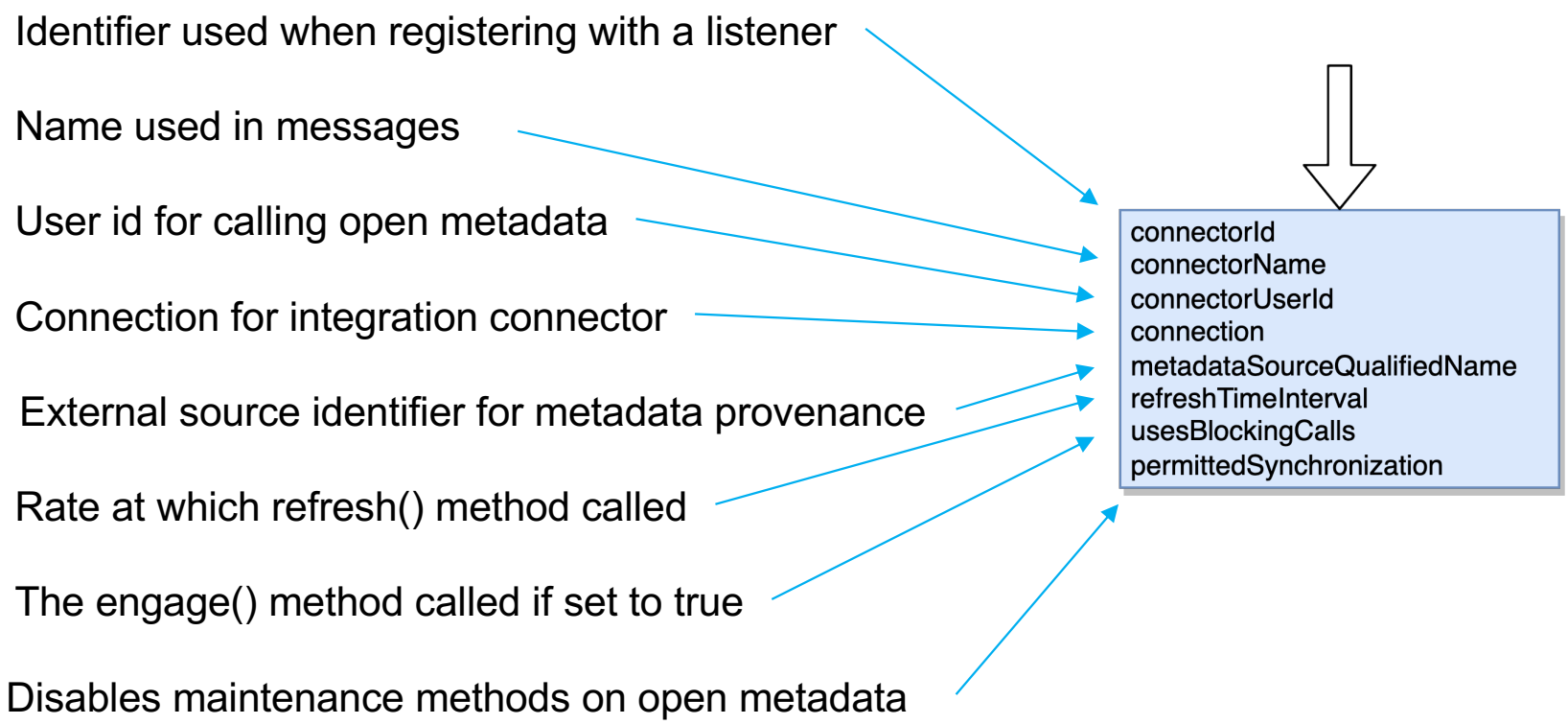
Free up any resources held since the connector is no longer needed.

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



Connector Provider

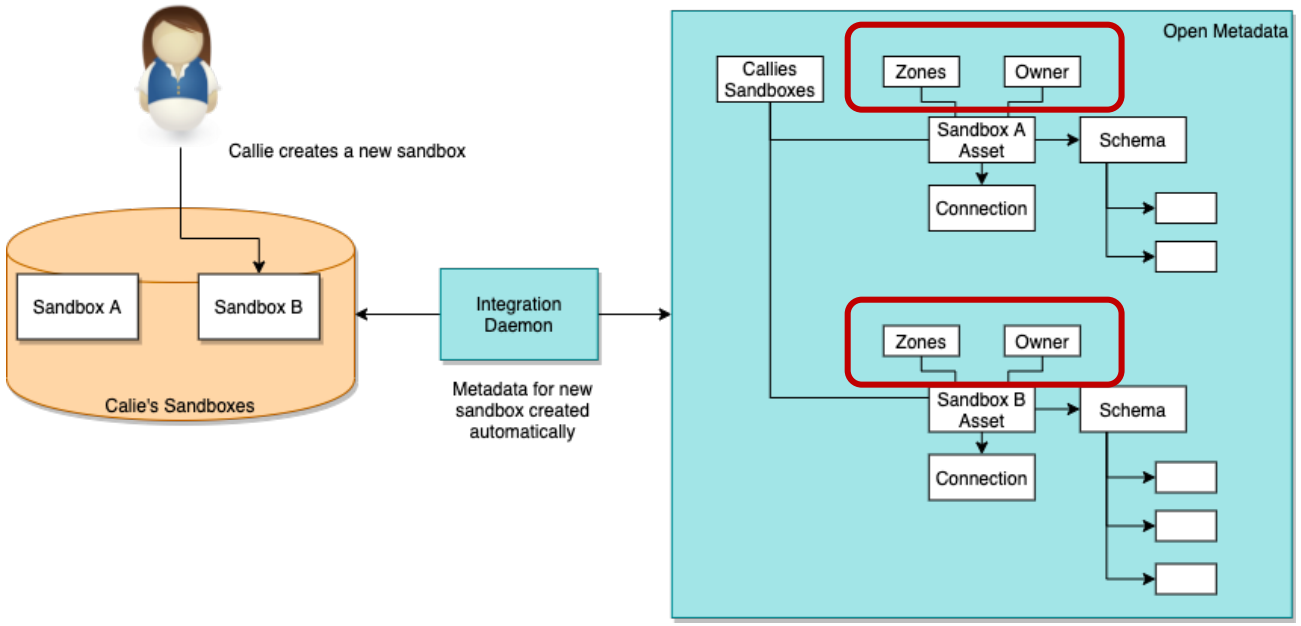
- Set up connector class name
- Build connector type
- Set up audit log component identifier

Configuration properties are added to the connection

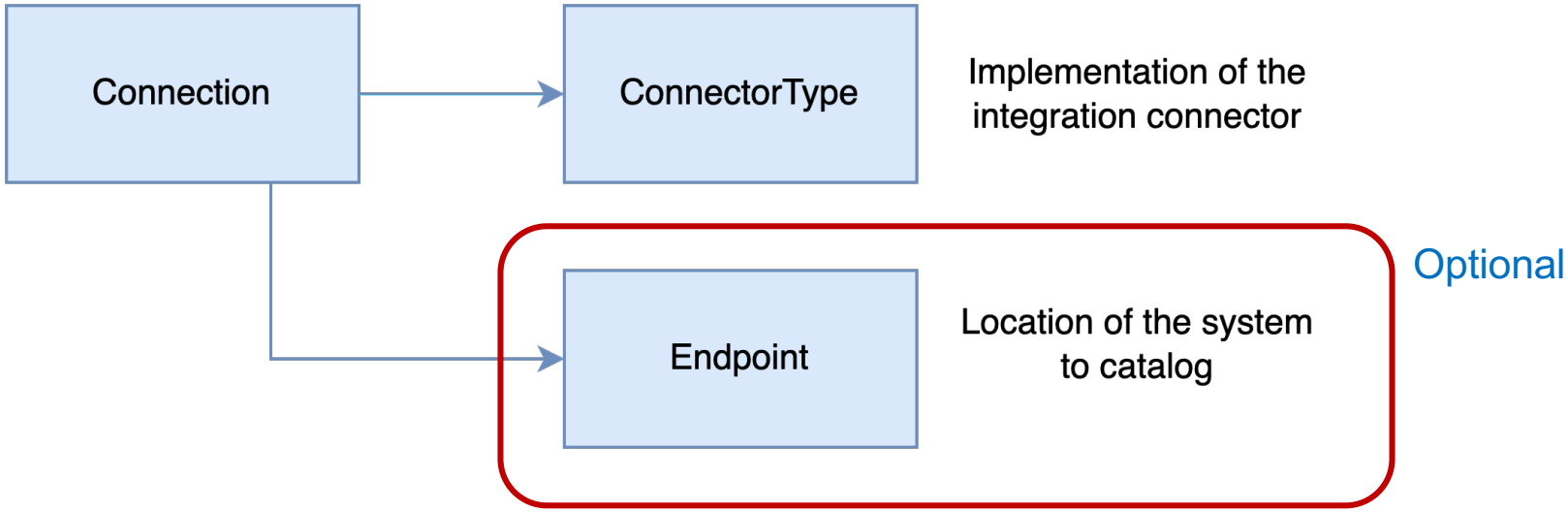
```
/**
 * BasicFilesMonitorIntegrationProviderBase is the base class provider for the basic files integration connectors.
 */
class BasicFilesMonitorIntegrationProviderBase extends IntegrationConnectorProvider
{
    static final String TEMPLATE_QUALIFIED_NAME_CONFIGURATION_PROPERTY = "templateQualifiedName";
    static final String ALLOW_CATALOG_DELETE_CONFIGURATION_PROPERTY    = "allowCatalogDelete";

    /**
     * Constructor used to initialize the ConnectorProviderBase with the Java class name of the specific
     * store implementation.
     *
     * @param connectorTypeGUID the unique identifier for the connector type
     * @param connectorComponentId the component id used by the connector in logging
     * @param connectorQualifiedName the unique name for this connector
     * @param connectorDisplayName the printable name for this connector
     * @param connectorDescription the description of this connector
     * @param connectorWikiPage the URL of the connector page in the connector catalog
     * @param connectorClass the name of the connector class that the connector provider creates
     */
    BasicFilesMonitorIntegrationProviderBase(String   connectorTypeGUID,
                                             int      connectorComponentId,
                                             String   connectorQualifiedName,
                                             String   connectorDisplayName,
                                             String   connectorDescription,
                                             String   connectorWikiPage,
                                             Class<?> connectorClass)
    {
        super();

        /*
         * Set up the class name of the connector that this provider creates.
         */
        super.setConnectorClassName(connectorClass.getName());

        /*
         * Set up the connector type that should be included in a connection used to configure this connector.
         */
        ConnectorType connectorType = new ConnectorType();
        connectorType.setType(ConnectorType.getConnectorTypeType());
        connectorType.setGUID(connectorTypeGUID);
        connectorType.setQualifiedName(connectorQualifiedName);
        connectorType.setDisplayName(connectorDisplayName);
        connectorType.setDescription(connectorDescription);
        connectorType.setConnectorProviderClassName(this.getClass().getName());

        List<String> recognizedConfigurationProperties = new ArrayList<>();
        recognizedConfigurationProperties.add(TEMPLATE_QUALIFIED_NAME_CONFIGURATION_PROPERTY);
        recognizedConfigurationProperties.add(ALLOW_CATALOG_DELETE_CONFIGURATION_PROPERTY);

        connectorType.setRecognizedConfigurationProperties(recognizedConfigurationProperties);

        super.connectorTypeBean = connectorType;

        /*
         * Set up the component description used in the connector's audit log messages.
         */
        AuditLogReportingComponent componentDescription = new AuditLogReportingComponent();

        componentDescription.setComponentId(connectorComponentId);
        componentDescription.setComponentDevelopmentStatus(ComponentDevelopmentStatus.TECHNICAL_PREVIEW);
        componentDescription.setComponentName(connectorQualifiedName);
        componentDescription.setComponentDescription(connectorDescription);
        componentDescription.setComponentWikiURL(connectorWikiPage);

        super.setConnectorComponentDescription(componentDescription);
    }
}
```

Using templates

- Governance metadata added through templates

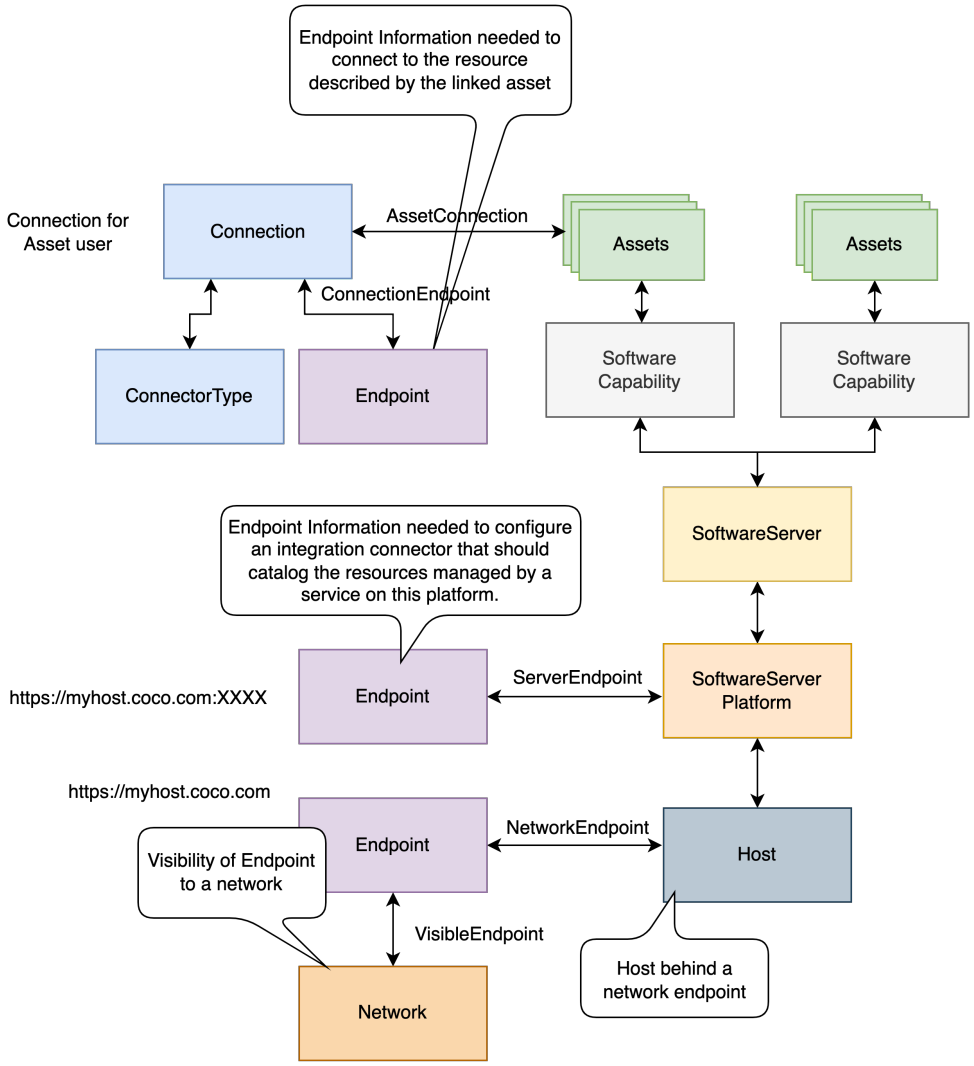


Connections for integration connectors

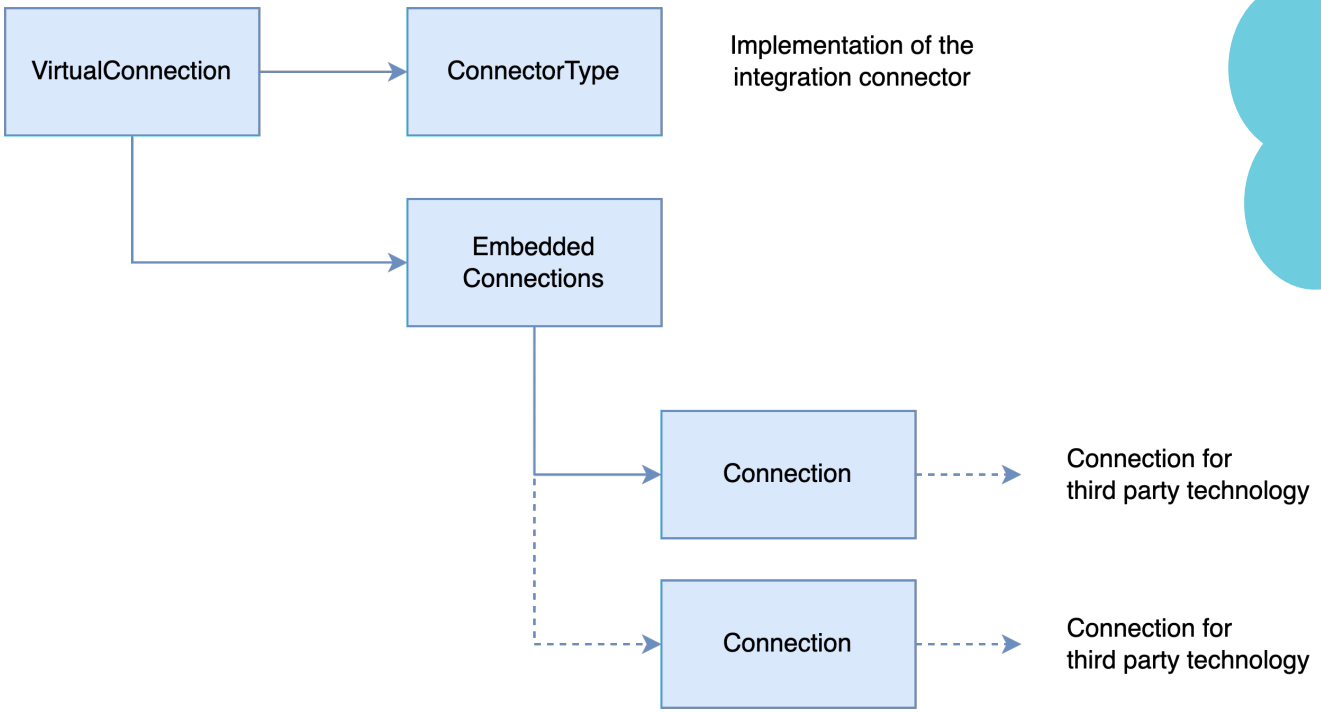


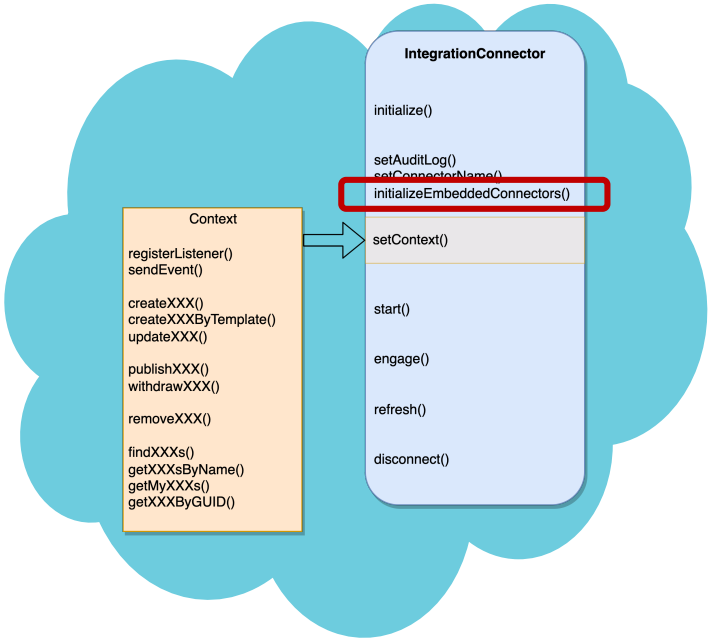
Locating endpoints

- 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

Call super class

Extract configuration

Log start

Register listeners

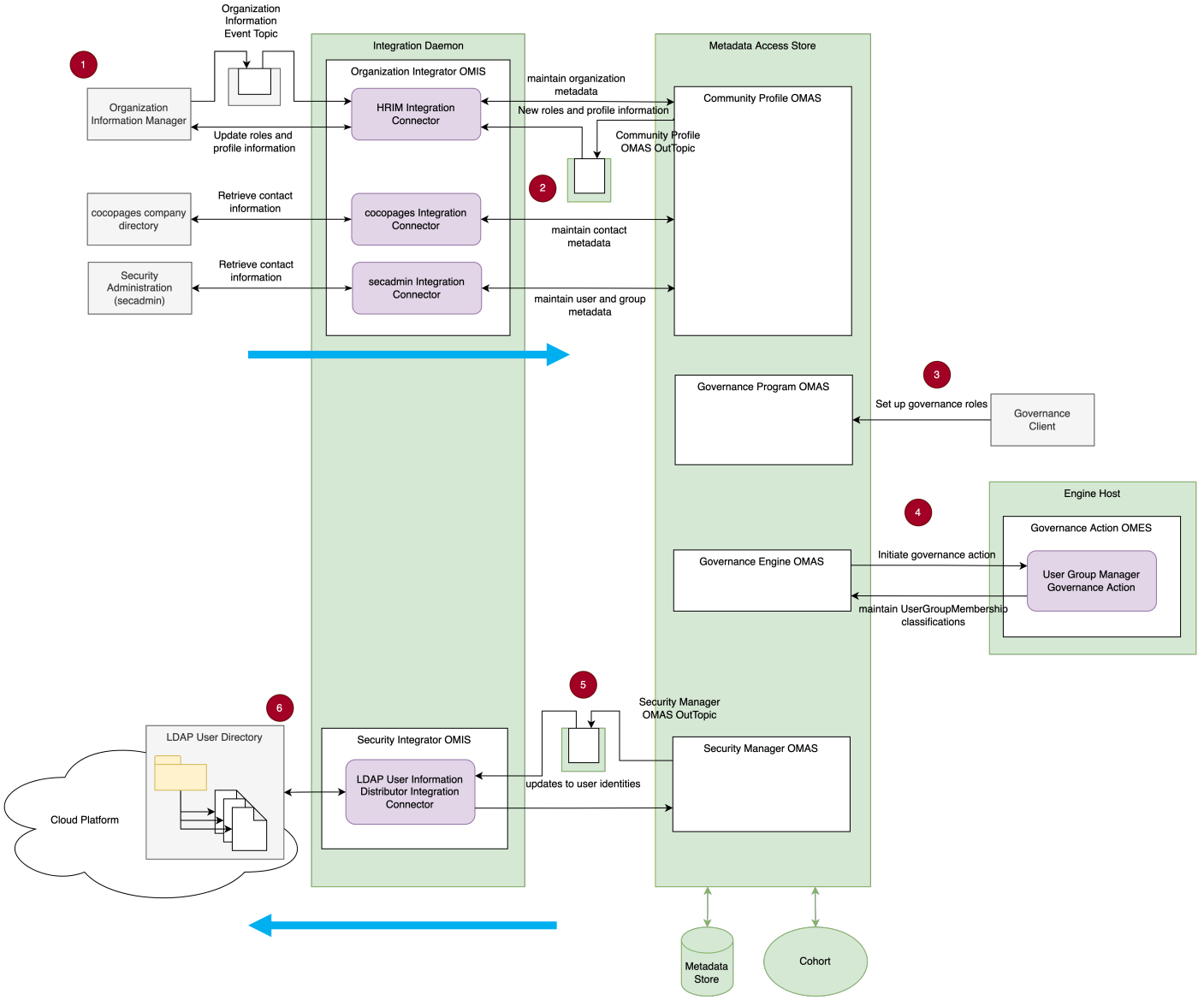
```java
/**
 * Indicates that the connector is completely configured and can begin processing.
 * This call can be used to register with non-blocking services.
 *
 * @throws ConnectorCheckedException there is a problem within the connector.
 */
@Override
public void start() throws ConnectorCheckedException
{
    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(BasicFilesMonitorIntegrationProviderBase.ALLOW_CATALOG_DELETE_CONFIGURATION_PROPERTY))
        {
            allowCatalogDelete = true;
        }

        templateQualifiedName = configurationProperties.get(BasicFilesMonitorIntegrationProviderBase.TEMPLATE_QUALIFIED_NAME_CONFIGURATION_PROPERTY).toString();
    }

    /*
     * Record the configuration
     */
    if (auditLog != null)
    {
        auditLog.logMessage(methodName,
                            BasicFilesIntegrationConnectorsAuditCode.CONNECTOR_CONFIGURATION.getMessageDefinition(connectorName,
                                                                                                                  fileDirectoryName,
                                                                                                                  Boolean.toString(allowCatalogDelete),
                                                                                                                  templateQualifiedName));
    }

    /*
     * Start listening
     */
    this.initiateDirectoryMonitoring(this.getRootDirectoryFile(), methodName);
}
```

Register open metadata listener

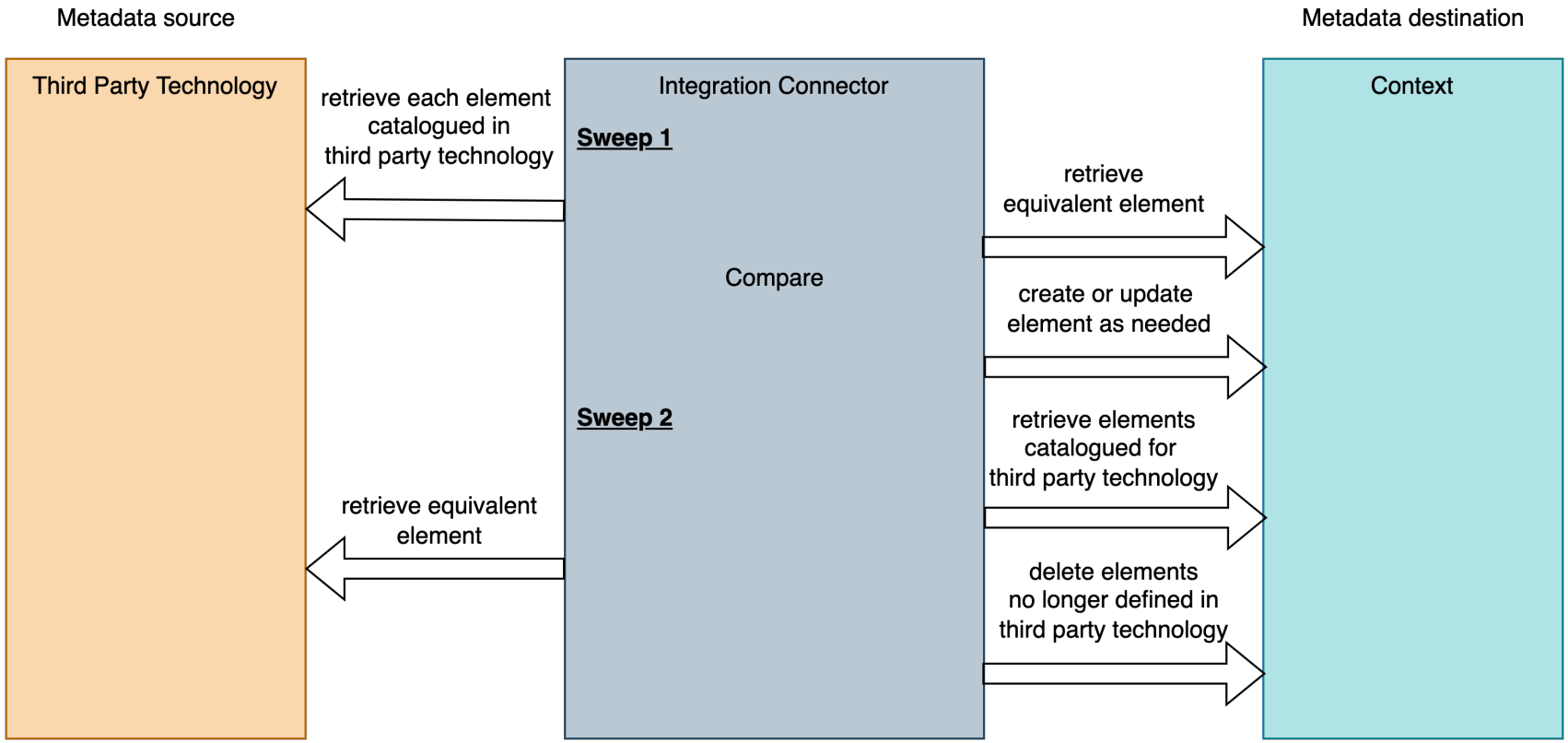

```java
/**
 * Indicates that the connector is completely configured and can begin processing.
 * This call can be used to register with non-blocking services.
 *
 * @throws ConnectorCheckedException there is a problem within the connector.
 */
@Override
public void start() throws ConnectorCheckedException
{
    super.start();

    final String methodName = "start";

    if (connectionProperties.getUserId() != null)
    {
        clientUserId = connectionProperties.getUserId();
    }

    myContext = super.getContext();

    try
    {
        myContext.registerListener(this);

        /*
         * Record the start
         */
        if (auditLog != null)
        {
            auditLog.logMessage(methodName,
                                EgeriaInfrastructureConnectorAuditCode.CONNECTOR_START.getMessageDefinition(connectorName, clientUserId));
        }
    }
    catch (Exception error)
    {
        throw new ConnectorCheckedException(EgeriaInfrastructureConnectorErrorCode.UNEXPECTED_EXCEPTION.getMessageDefinition(connectorName,
                                                                                                                              error.getClass().getName(),
                                                                                                                              error.getMessage()),
                                            this.getClass().getName(),
                                            methodName,
                                            error);
    }

}
```

Example solution

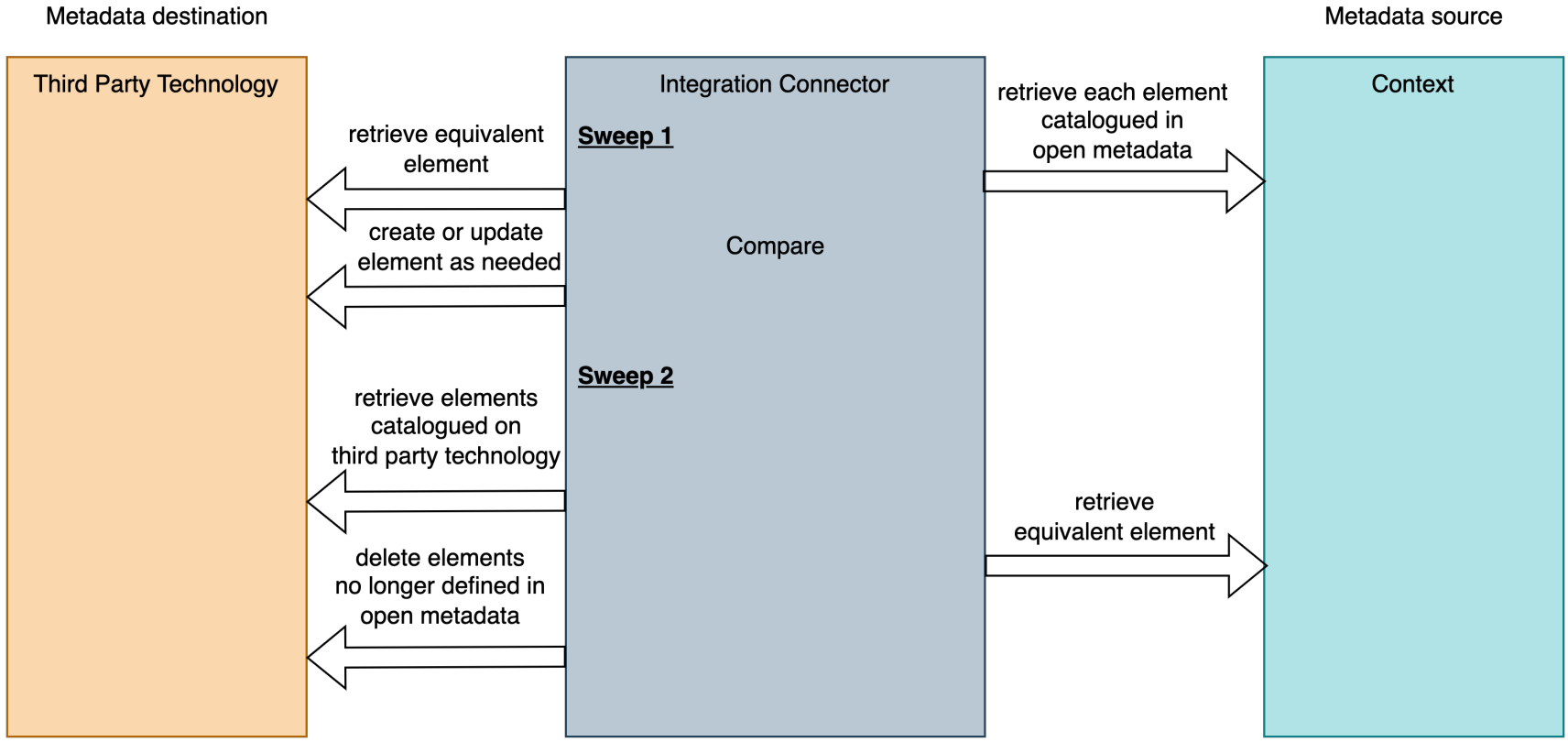
- Metadata synchronization can be inbound or outbound and this affects the design of your integration connector



Refresh method: Third-party metadata source

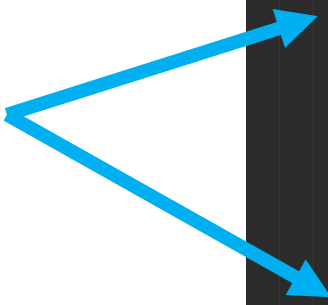


Refresh method: Third-party metadata destination



Refresh method

Two sweeps

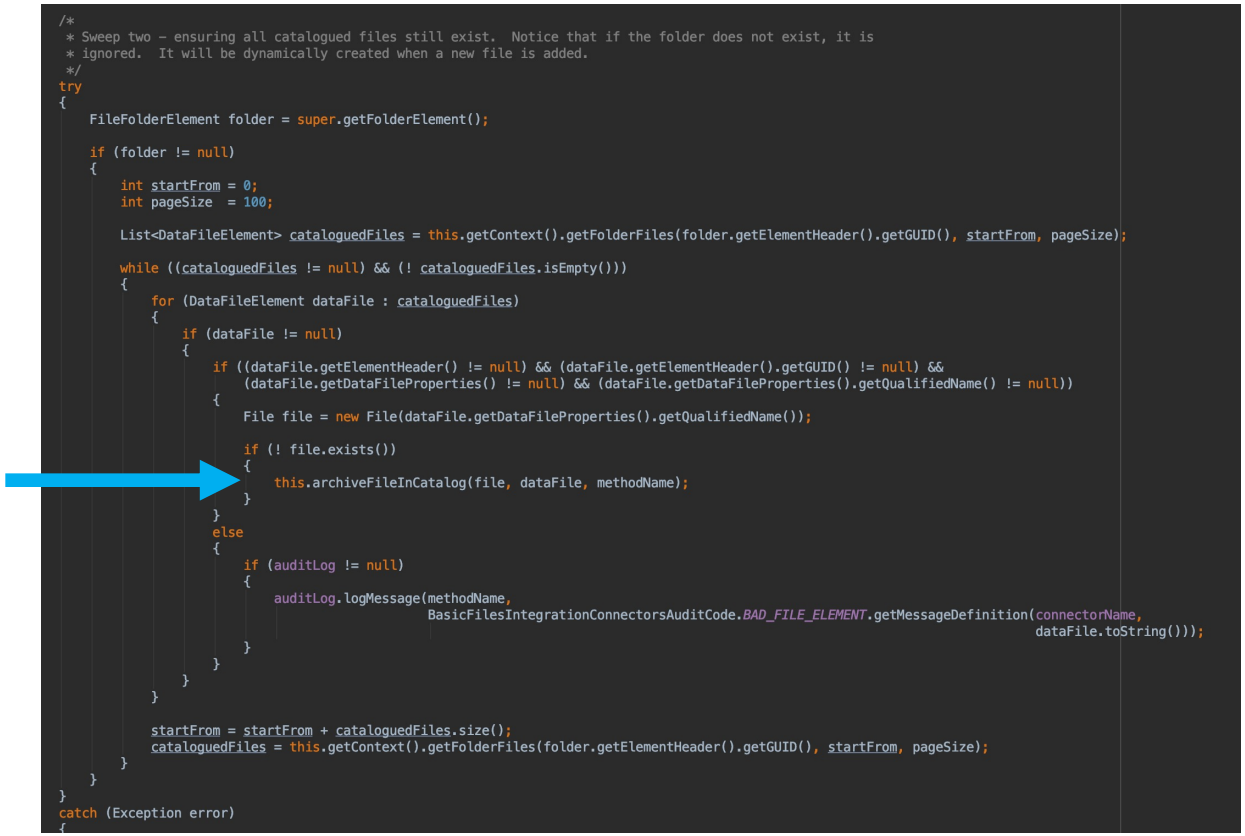


```java
/**
 * Requests that the connector does a comparison of the metadata in the third party technology and open metadata repositories.
 * Refresh is called when the integration connector first starts and then at intervals defined in the connector's configuration
 * as well as any external REST API 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 on 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.getAbsolutePath(),
                                                                                                                                    error.getMessage()),
                                      error);

            }

            throw new FileException(
                    BasicFilesIntegrationConnectorsErrorCode.UNEXPECTED_EXC_DATA_FILE_UPDATE.getMessageDefinition(error.getClass().getName(),
                                                                                                                  connectorName,
                                                                                                                  directory.getAbsolutePath(),
                                                                                                                  error.getMessage()),
                    error.getClass().getName(),
                    methodName,
                    error,
                    directory.getAbsolutePath());
        }
    }
}
```

EGERIA

Refresh method - sweep 2

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



```
/*
 * 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
{
    FileFolderElement folder = super.getFolderElement();

    if (folder != null)
    {
        int startFrom = 0;
        int pageSize  = 100;

        List<DataFileElement> cataloguedFiles = this.getContext().getFolderFiles(folder.getElementHeader().getGUID(), startFrom, pageSize);

        while ((cataloguedFiles != null) && (! cataloguedFiles.isEmpty()))
        {
            for (DataFileElement dataFile : cataloguedFiles)
            {
                if (dataFile != null)
                {
                    if ((dataFile.getElementHeader() != null) && (dataFile.getElementHeader().getGUID() != null) &&
                        (dataFile.getDataFileProperties() != null) && (dataFile.getDataFileProperties().getQualifiedName() != null))
                    {
                        File file = new File(dataFile.getDataFileProperties().getQualifiedName());

                        if (! file.exists())
                        {
                            this.archiveFileInCatalog(file, dataFile, methodName);
                        }
                    }
                    else
                    {
                        if (auditLog != null)
                        {
                            auditLog.logMessage(methodName,
                                                BasicFilesIntegrationConnectorsAuditCode.BAD_FILE_ELEMENT.getMessageDefinition(connectorName,
                                                                                                                               dataFile.toString()));
                        }
                    }
                }
            }

            startFrom = startFrom + cataloguedFiles.size();
            cataloguedFiles = this.getContext().getFolderFiles(folder.getElementHeader().getGUID(), startFrom, pageSize);
        }
    }
}
catch (Exception error)
{
```

Catalog file

Cataloguing file if it is not already in the catalog

```java
/**
 * Create a catalog entry for a specific file.
 *
 * @param file Java File accessor
 * @param methodName calling method
 */
private void catalogFile(File   file,
                         String methodName)
{
    if (this.isActive())
    {
        try
        {
            DataFileElement cataloguedElement = this.getContext().getFileByPathName(file.getAbsolutePath());

            if (cataloguedElement == null)
            {
                if (templateQualifiedName == null)
                {
                    String fileExtension = FilenameUtils.getExtension(file.getAbsolutePath());

                    DataFileProperties properties = new DataFileProperties();

                    properties.setTypeName(this.getAssetTypeName(fileExtension));
                    properties.setQualifiedName(file.getAbsolutePath());
                    properties.setDisplayName(file.getName());
                    properties.setModifiedTime(new Date(file.lastModified()));

                    List<String> guids = this.getContext().addDataFileToCatalog(properties,  connectorProviderName: null);

                    if ((guids != null) && (!guids.isEmpty()) && (auditLog != null))
                    {
                        auditLog.logMessage(methodName,
                                            BasicFilesIntegrationConnectorsAuditCode.DATA_FILE_CREATED.getMessageDefinition(connectorName,
                                                                                                                            properties.getQualifiedName(),
                                                                                                                            guids.get(
                                                                                                                                    guids.size() - 1)));
                    }
                }
                else
                {...}
            }
        }
        catch (Exception error)
        {
            if (auditLog != null)
            {
                auditLog.logException(methodName,
                                      BasicFilesIntegrationConnectorsAuditCode.UNEXPECTED_EXC_DATA_FILE_UPDATE.getMessageDefinition(
                                              error.getClass().getName(),
                                              connectorName,
                                              file.getAbsolutePath(),
                                              error.getMessage()),
                                      error);

            }
        }
    }
}
```

EGERIA

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().getFileByPathName(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));
        }
    }
}
```

EGERIA

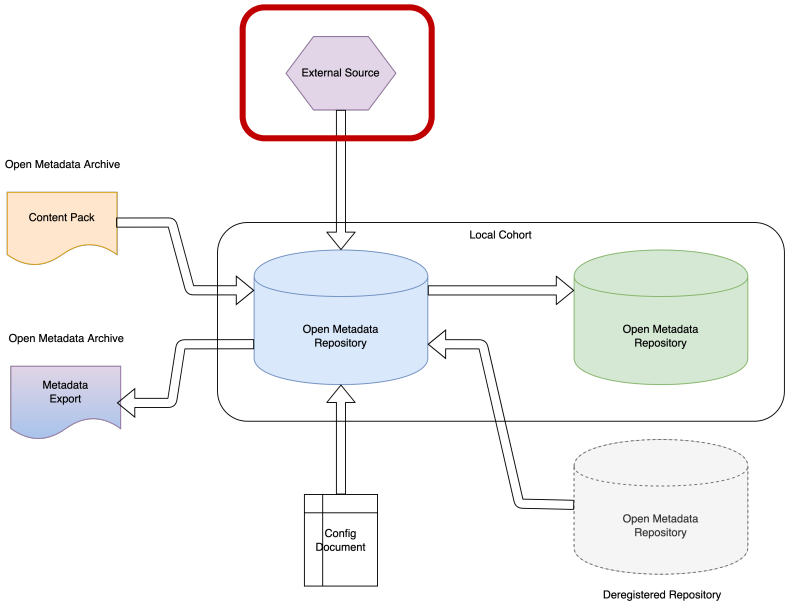
Disconnect method

- 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();
}
```

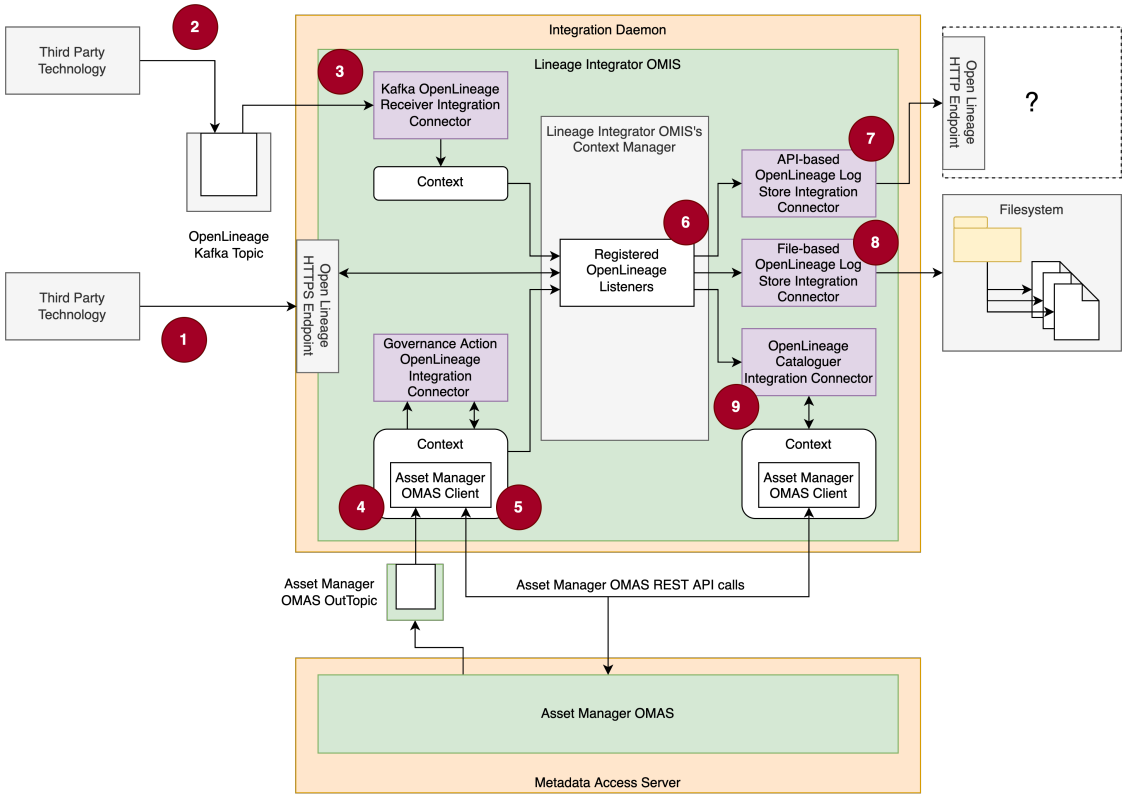
Metadata Provenance

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

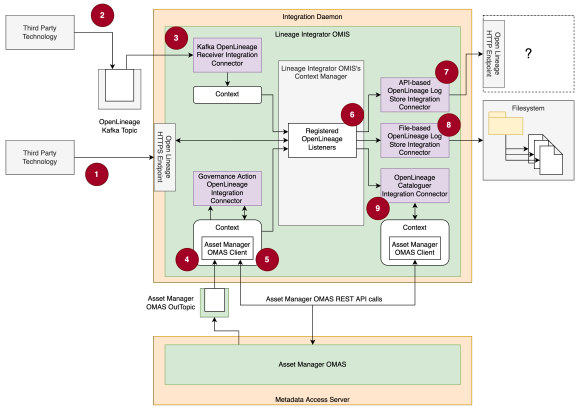


Integration Service	Method to control external source metadata provenance
Analytics Integrator OMIS	Call `setAnalyticsToolIsHome()` method to set toggle. Default is `true`.
API Integrator OMIS	Call `setAPIManagerIsHome()` method to set toggle. Default is `true`.
Catalog Integrator OMIS	Use `assetManagerIsHome` property on method calls.
Database Integrator OMIS	External source metadata provenance always enabled.
Display Integrator OMIS	Call `setApplicationIsHome()` to set toggle. Default is `true`.
Files Integrator OMIS	Local cohort metadata provenance is always enabled.
Infrastructure Integrator OMIS	Call `setInfrastructureManagerIsHome()` method to set toggle. Default is `true`.
Lineage Integrator OMIS	Use `assetManagerIsHome` property on method calls.
Organization Integrator OMIS	Local cohort metadata provenance is always enabled.
Search Integrator OMIS	Not applicable - outbound only
Security Integrator OMIS	Local cohort metadata provenance is always enabled.
Topic Integrator OMIS	Call `setEventBrokerIsHome()` method to set toggle. Default is `true`.

Inside the Lineage Integrator OMIS

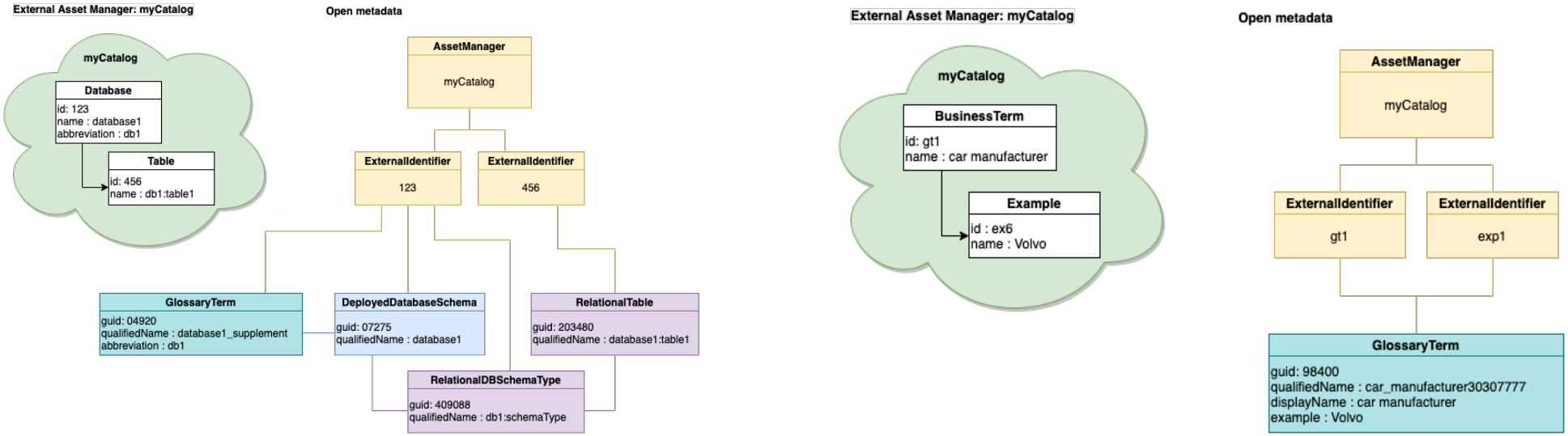


Methods from the Lineage Integration OMIS context




```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(OpenLineageEventListener 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 rawEvent json payload to send for the event
 */
public void publishOpenLineageRunEvent(String rawEvent) { openLineageListenerManager.publishOpenLineageRunEvent(rawEvent); }


/**
 * 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 reformat 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 OMAS OutTopic
 */

/**
 * Register a listener object that will be passed each of the events published by the Asset Manager OMAS.
 *
 * @param listener listener object
 *
 * @throws InvalidParameterException one of the parameters is null or invalid.
 * @throws ConnectionCheckedException there are errors in the configuration of the connection which is preventing
 *                                    the creation of a connector.
 * @throws ConnectorCheckedException there are errors in the initialization of the connector.
 * @throws PropertyServerException there is a problem retrieving information from the property server(s).
 * @throws UserNotAuthorizedException the requesting user is not authorized to issue this request.
 */
public void registerListener(AssetManagerEventListener listener) throws InvalidParameterException,
                                                                        ConnectionCheckedException,
                                                                        ConnectorCheckedException,
                                                                        PropertyServerException,
                                                                        UserNotAuthorizedException
```

Examples of granularity challenge



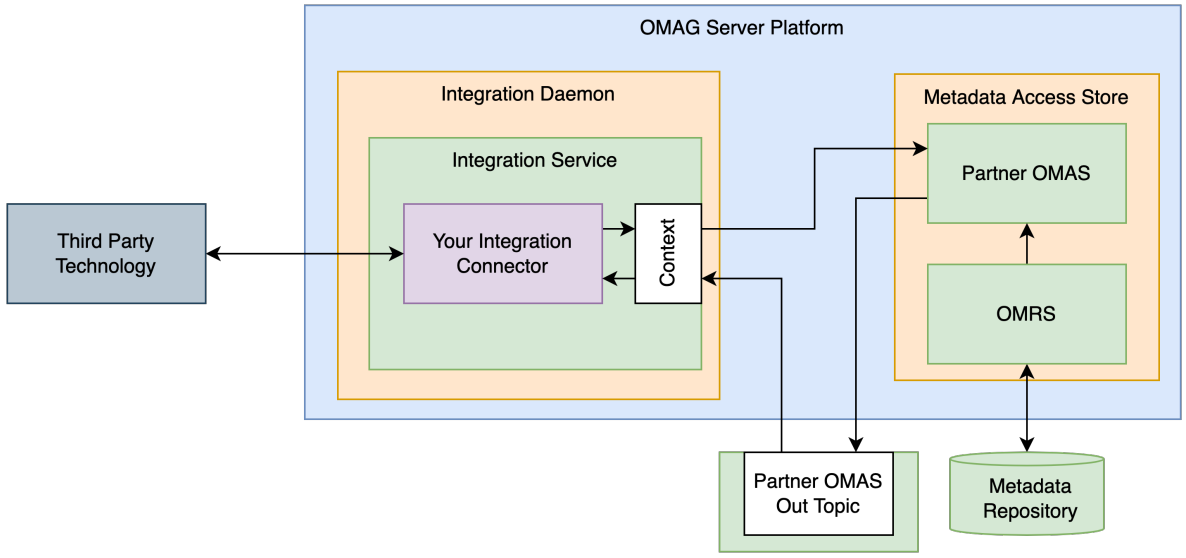
External identifiers are supported by Catalog Integrator OMIS and Lineage Integrator OMIS

Extract from the Catalog Integrator OMIS

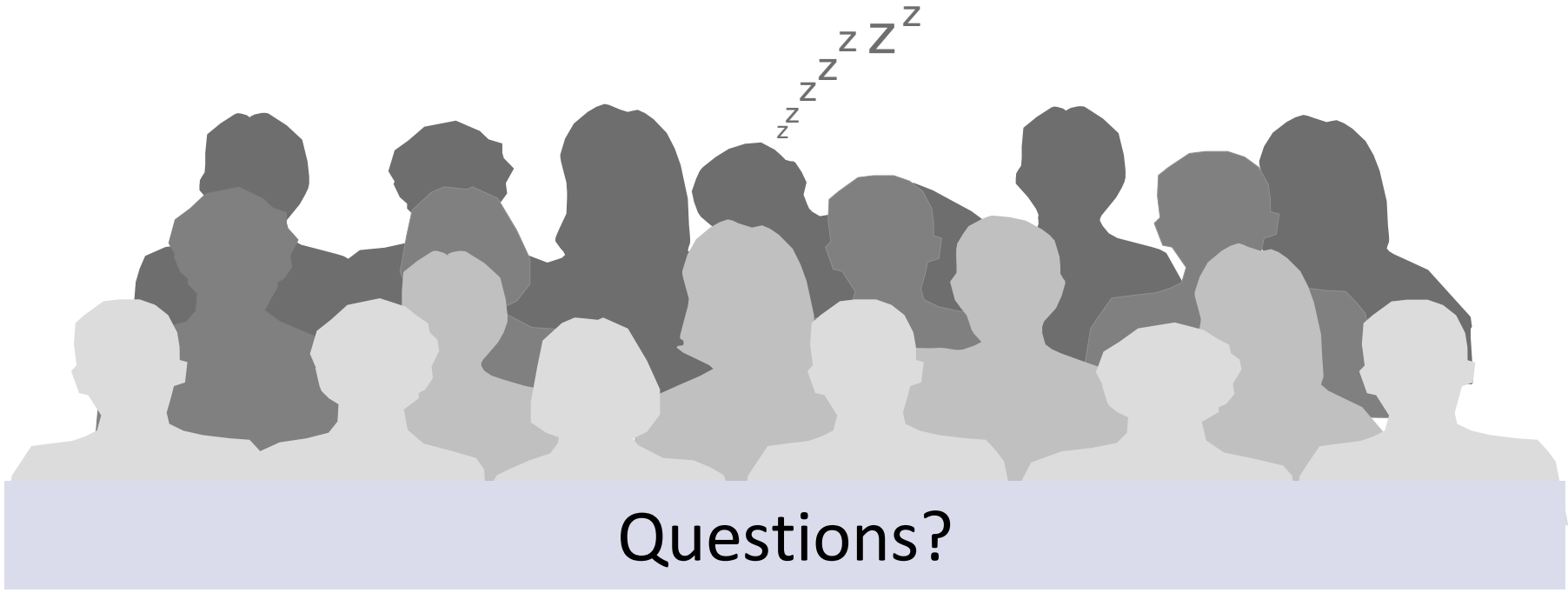
```
/**
 * Create a new metadata element to represent the root of an asset.
 *
 * @param assetManagerIsHome ensure that only the asset manager can update this asset
 * @param assetExternalIdentifier unique identifier of the asset in the external asset manager
 * @param assetExternalIdentifierName name 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 InvalidParameterException  one of the parameters is invalid
 * @throws UserNotAuthorizedException 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 InvalidParameterException,
                                                                          UserNotAuthorizedException,
                                                                          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



Egeria's webinar series

7th February 2022	15:00 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 CMDBs. - 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. Zoom Conference https://zoom.us/j/523629111	**Mandy Chessell**
~~**7th**~~ **15th March 2022**	~~15:00~~ 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 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/523629111	**Mandy Chessell**
4th April 2022	15:00 UTC	**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. Zoom Conference https://zoom.us/j/523629111	**Chris Grote**

THANK YOU!

EGERIA

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 heterogenous 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