

Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.



Public GRC API

**Brian Laskey,
Senior Software Engineer, IBM**



Agenda

- Introduction
- Overview Java APIs
- REST APIs



What is the GRC API?

- Improved Application Programming Interfaces (API) for building custom solutions in the OpenPages application
 - Improve productivity by making coding simpler
 - Same functionality for custom solutions
 - Easier to learn, train developers with better documentation and examples
 - Developed with Java
- Brand new functionality for remote clients to interact with OpenPages (RESTful Webservices)
 - External Systems for Data Integration
 - Decoupled User Interfaces
 - Developed with open technologies, HTTP, REST, JSON

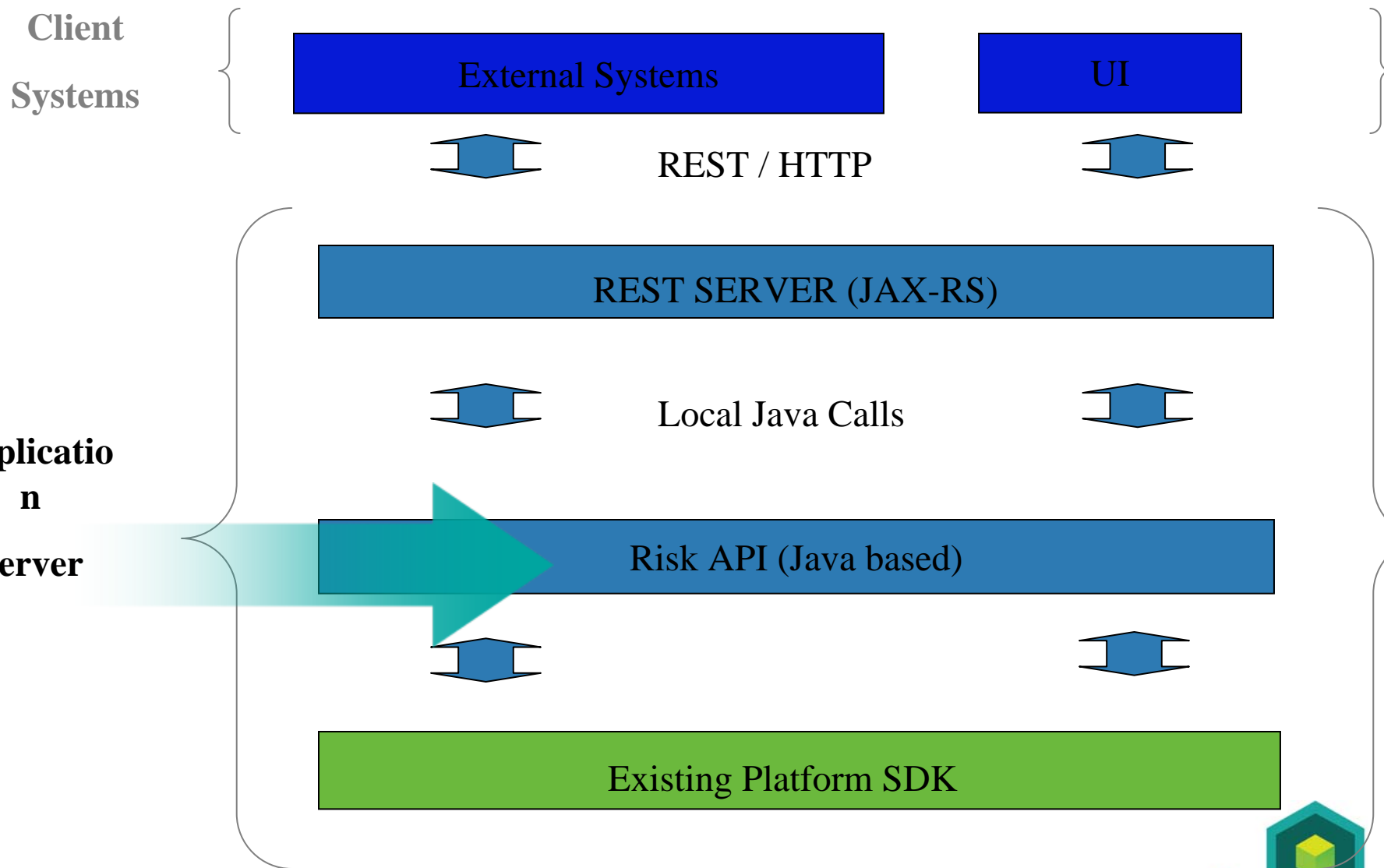


Target Audiences

- To use the GRC API in custom solutions you must be a Java developer and familiar with how to package and deploy custom deliverables, following existing practices.
 - Skillsets: Java, J2EE, Databases, XML
- To use the REST API you must be familiar with HTTP and technologies that support client-server programming patterns
 - Skillsets: JSON, RESTful webservice, HTTP, client-side development using one or more: Java, Javascript, VBA, Mobile platforms, etc...



Technology Stack – In the Labs



Usage of New API On the Platform

- With new API Custom Solutions may be developed using the new Services of the Risk API
- Same customizations as in prior release are possible
 - UI “Helpers” – written with Java/JSP
 - OpenPages Triggers – Custom Business Logic
 - Workflow Java Actions
- Existing Code is not impacted
- Do not need to rewrite existing customizations



Core Services

- API both Java and REST support working with following areas
- Metadata
- Resources
- Security
- Configuration
- Application
- Workflow
- Audit
- Query



Review Java API Services and methods.

Basic operations



Basic Operations

- Context and ServiceFactory
- MetadataService
 - (Object Types, Fields, Dependencies)
- ResourceService
 - (CRUD Operations on Objects, Folders and Associations)
- ConfigurationService
 - (Profiles, Currencies, Registry)



Getting Started – The Context

- Context contains information needed to use API services in your code
- To start a session with the GRC API you need to provide create a Context for the current user. To create a new Context without an existing OpenpagesSession provided by existing OpenPages GRC platform you must provide user credentials.

```
Context context = new Context();  
//add connection information for the environment, and user credentials  
context.put(Context.SERVICE_USER_NAME, "jsmith");  
context.put(Context.SERVICE_USER_PASSWORD, "*****");
```

- Note other information is retrieved from `aurora.properties`



Getting Started – The Context

- To create a new Context where you already have an existing session:

```
Context context = new Context();
```

```
//add connection with an existing OpenpagesSession  
context.put(Context.SERVICE_SESSION,openpagesSession);
```

- While above is generally true, in certain deliverables you can retrieve the current Context for that session.



Getting Started – ServiceFactory

- Factory Pattern which uses Context to create API Services
- With the Context get a reference to the **IServiceFactory** which allows you to create any of the other services to interact with the OpenPages platform features

```
//gets the reference IServiceFactory  
IServiceFactory serviceFactory =  
ServiceFactory.getServiceFactory(context);
```

```
//create the services you need to use  
IMetaDataService metaService =  
serviceFactory.createMetaDataService();  
//etc...
```

See all available services interfaces in

Package com.ibm.openpages.api.service



MetaDataService

- Interface: `com.ibm.openpages.api.service.IMetaDataService`
- Access to the MetaData or “Schema” information in the OpenPages Object Model
 - Object Types
 - Field Definitions
 - Associations between object types
- Package for related public interfaces
 - **`com.ibm.openpages.api.metadata`**



MetaData – Object Types

- Object Types are represented by Interface: ITypeDefinition
- Retrieve from MetadataService
 - **getType**(Id id)
 - **getType**(java.lang.String systemName)
 - **getTypes**()
- System Name = “SOXBusEntity”, “SOXIssue”, “LossEvent”
- Id = SDK/Platform ContentTypeId
- Note **com.ibm.openpages.api.metadata.Id** used in above method signature is used through out the rest of the GRC API for unique identifiers of every object.
- The System Name is used in QueryService as an object identifier.
- From ITypeDefinition retrieve
 - Name, description, localized label
 - Associations to other Object Types
 - Field Definitions
 - FileType – (file extension, useful for SOXDocument)
 - Root folder



MetaData – Object Types and Fields

- No “BundleType” or Field Group. In terms of development the construct does not add much value. All Fields associated to an Object Type through Field Groups are provided
- Fields now referenced with the Field Group name to provide a unique name.
 - “Field Group:Field Name”
- Examples:
 - “OPSS-BE:Executive Owner”
- Getting a ITypeDefinition and fields:

```
ITypeDefinition issueType = metaDataService.getType("SOXIssue");  
fieldsList = issueType.getFieldsDefinition();
```



MetaData – Fields

- Fields are represented by Interface: IFieldDefinition
- Retrieve from MetaDataService
 - **getField**(java.lang.String systemName)
- Retrieve from an Object type's ITypeDefinition
 - **getField**(Id id)
 - **getField**(java.lang.String systemName)
 - **getFieldsDefinition**()
- System Name = FieldGroup:Field Name e.g. “OPSS-Iss:Status”
- Id = SDK/Platform Propertypeld



MetaData – Fields

- From IFieldDefinition
 - What DataType – See DataType enum: DATA_TYPE, CURRENCY_TYPE, etc...
 - Id, Name, Description and Local Label and other attributes
 - For Enum Types: Possible Values as IEnumValue interface
 - Field Dependencies: **getDependencies()** and **isController()**
- Field Dependencies are provided so that the Dependent fields and know what other field values matter based on their controllers.
- Single interface IFieldDependency represents the dependent field for both Dependencies and Dependent Picklists. For each FieldDependencyType there will be a listing of the Controlling Fields and Values per the configured dependencies.
 - For Dependent Picklists – mapping of controller values to dependent values, all IEnumValue.



MetaData – Associations

- Definitions of Associations between Object types are represented by Interface: IAssociationDefinition
- From IAssociationDefinition
 - **getChild()** – information on the child as a ITypeDefinition
 - **getParent()** - information on the parent as a ITypeDefinition
 - **isEnabled()** – whether the association is enabled in the schema or not.



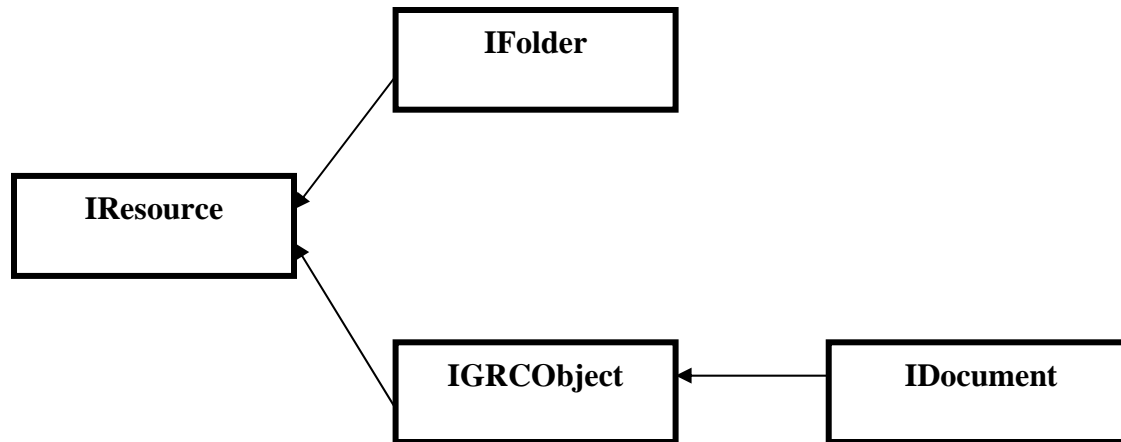
ResourceService

- Interface: `com.ibm.openpages.api.service.IResourceService`
- Access to the object instances or “Resources” from OpenPages data.
 - GRC Objects and their values
 - Folders
 - Documents
- Perform updates to all objects
 - Create
 - Update
 - Delete
 - Change associations between GRC Objects
- Package for related public interfaces
 - **`com.ibm.openpages.api.resource`**
- GRC Object is an instance of any of the typical Object Types in OpenPages UI



Resource - class hierarchy

- Everything is a Resource
- Specializations have different methods exposed in the interfaces



Resource - IGRObject

- Main interface for working with instances of data
 - Interface: **com.ibm.openpages.api.resource.IGRObject**
 - System fields inherited from IResource
 - Custom fields in **getFields()** / **getField(java.lang.String systemName)**
 - Field values represented by IField, values can be get/set on the IField
 - Get TypeDefinition,
 - Getters / Setters for Parent Folders, Primary Parent (Id)
 - Get info about direct parents or children
-
- **Version is no longer exposed explicitly. Just work with the object and its fields.**



Resource – Creating Objects

- Process is to create an object from IResourceFactory.
- The reference to the Object is in memory only. Make changes to it, set values etc.
- Call ResourceService.saveResource() to persist.

```
// Get the Resource Factory
IResourceFactory rss = resourceService.getResourceFactory();
// Get the type Definition of the Object being created
ITypeDefinition objectTypeDef = metaDataService.getType(type);
// Create AutoNamed Child Object
IGRCObject newObject = rss.createAutoNamedGRCObject(objectTypeDef);
//Set Description
newObject.setDescription(childDescription);
//Set Parent Association
newObject.setPrimaryParent(parentObject.getId());
//Create the Object
newObject = resourceService.saveResource(newObject);
```



Resource – Creating Objects Notes

- Specifying a parent folder explicitly is optional:
- Folder Path defaults to Object Type Root Folder
 - If Primary Parent is specified before create, new Object is created under the Primary Parent's relative folder path.
- Do not need to provide an extension for GRCOBJECTS' names
- Note Most Behavior of saveResource() is from SDK's ObjectService.create();



Resource – Retrieving Objects

- Use ResourceService to get the object from either it's path or Id (ResourceId)
- Optional - Can use the GRCObjectFilter for telling the ResourceService what data to bring back
 - Reporting Period – Default - current
 - Fields – Default - all
 - Which Direct Associations (for getChildren(), getParents()) calls. – Default None

```
//get GRC Object Filter for current and certain fields
GRCObjectFilter myFilter = new
GRCObjectFilter(confService.getCurrentReportingPeriod());
//These are IFieldDefinitions from MetaDataService
myFilter.setFieldFilters(l1TextAreaFieldDef,l1MediumRTFFieldDef);
//get the GRC Object with Full Path and Filter
res1 = resourceService.getGRCObject(fullPath, myFilter);
```



Resource – Updating Objects

- After Retrieving a GRCObject, make changes to it, or it's Fields
- Call ResourceService.saveResource() to persist.

```
//get the GRC Object with Full Path and Filter  
res1 = resourceService.getGRCObject(fullPath, myFilter);  
IStringField someTextField = (IStringField) res1.getField("LevelOne:Text  
Area");  
someTextField.setValue("New value");  
//update the object  
updatedObject = resourceService.saveResource(newObject);
```



Resource – Updating Objects Notes

- If you change the parent folder using `setParentFolder()`, this will Move the GRObject to the new Folder. Associations are unchanged.
 - For Hierarchical Moves refer to `IApplicationService`



Resource – Fields

- All Field values are represented by IField interface
- This provides the Field Name, Data Type, FieldDefinition, isNull
- Depending on the Data Type, each instance of the IField will be a subclass:
 - IBooleanField, IFloatField, IStringField etc..
- Which you may cast the IField to. You may know this explicitly, or you can make a programmatic check using DataType or instanceof

```
if (field instanceof IDateField) {  
    IDateField dateField = (IDateField)field;  
    Date date = dateField.getValue();  
}
```

- The field values can now be explicitly set based on the type. Less guessing on what to cast the value to. Examples:
 - **IEnumField.setEnumValue**(IEnumValue enumValue)
 - **IntegerField.setValue**(java.lang.Integer value)



Resource – Utilities

- **com.ibm.openpages.api.resource.util.ResourceUtil**
- Constructs / derives Full Paths from Relative Paths and vice versa
- Get list of modified fields for a GRCOObject before saving. (Can be used in Triggers)
- Field value setter
 - **setFieldValue**(IField field, java.lang.Object value)
- Can be used as another way to set values. Refer to the JavaDocs for what format it accepts in the value argument.



Resource – Getting Parents or Children

- From IGRCOBJECT you can get the Direct Parents or Direct Children
- Depends on what is requested in GRCOBJECTFilter.
- Example, for a child of Entity, request to retrieve only Parents, of type SOXBusEntity

```
GRCObjectFilter includeParentEnt = new GRCObjectFilter(
cs.getCurrentReportingPeriod());

includeParentEnt.getAssociationFilter().setIncludeAssociations(
IncludeAssociations.PARENT);

includeParentEnt.getAssociationFilter().setTypeFilters(entityType);

IGRCObject childWithParents = rs.getGRCObject(child.getId(), includeParentEnt);
List<IAssociationNode> parentsOfL1Associations = childWithParents.getParents();
```

- Each IAssociationNode contains the parent Id, and Path and Type, which could be used to retrieve the full GRCOBJECT
- **Best Practice – Do not use for “tree walking”. If going up many levels, use QueryService.**
- **Always try to bring back the minimal data you require with each call.**



ConfigurationService

- Interface: `com.ibm.openpages.api.service.IConfigurationService`
- Access to the configuration / administration information in OpenPages
 - Settings or “Registry”
 - Currencies
 - Profiles
 - Application Text
 - Reporting Period
- Enable / Disable
 - SAM (System Admin Mode)
 - Profile
 - Currencies
- Package for related public interfaces
 - **`com.ibm.openpages.api.configuration`**



Configuration - Settings

- OpenPages “Registry” settings available in IConfigProperties

```
IConfigProperties settings = configService.getConfigProperties() =  
String childTypes =  
    settings.getProperty("/OpenPages/Applications/GRCM/Locked Objects/Lock  
    Child Types/");
```



Configuration - Currencies

- Currency and Exchange rate information
 - **getBaseCurrency()**
 - **getCurrency**(ISOCurrencyCode currencyCode)
 - **getCurrencies**(boolean onlyEnabled)
 - **getExchangeRate**(ICurrency currency, java.util.Date date)
 - **getCurrentExchangeRate**(ICurrency currency)
- Currencies identified by their 3-letter ISO 4217 code which is represented by the ISOCurrencyCode Enum
- IExchangeRate is a double that has the start and end dates when the rate was effective.
- Can enable or disable a Currency



Configuration – Reporting Periods

- Get the reference to the current reporting period as a convenience. Always has Id -1
 - **getCurrentReportingPeriod()**
- Also get the IReportingPeriod by label, or get a list of all Reporting periods in the system
- See GRCObjectFilter to filter the data retrieved.
- Also can be used to scope QueryService



Configuration – Profiles – Quickly Get Field Display Info

- From ConfigurationService
 - **getProfileFields**(java.lang.String profileName, java.lang.String objectName, java.lang.String viewName)
- IProfileFieldDefinition extends the MetaData's IFieldDefintion with additional info from specified Profile's view for that object.
- From IProfileFieldDefinition retrieve:
 - DisplayType
 - Display Order
 - Read Only / Required at Profile level
 - Profile Field Id and Profile Type Definition Id (distinct from Metadata Ids)
- IDisplayType
 - Name – indicates what it is: Rich Text, Drop Down, User Selector
 - Parameters – is Map of display type specific parameters
 - com.ibm.openpages.api.configuration.DisplayTypes has constants for all current DisplayType names and their related Parameters e.g. TEXT_AREA_COLUMNS_KEY



Configuration – Profiles

- Can retrieve full profile as IProfile object
 - **getDefaultProfile()**
 - **getProfile**(java.lang.String name)
 - **getPreferredProfile()** – current user's profile
 - **getPreferredProfile**(IUser user) – get profile for User
 - **getProfiles()** - all profiles
- Can Retrieve the Id, Name, fall back or default
- Definition of entire profile is retrieved as XML document
- Can enable / disable the Profile



Review API Services and methods.

Queries



Query Service

- Specification and execution of complex queries against the OpenPages model
- Query Syntax: Subset of SQL- 92
 - SELECT Only
- Queries are expressed by this SQL-like syntax rather than through complex Java structures.
- Queries are specified in terms of object model types:
 - RiskAssessment, SOXBusEntity, SOXRisk, etc.
- Query Service converts the SQL expression and translates to physical DB tables
 - [FieldGroup:Fields] → columns
 - [Object Types] → tables
- API is used to consume result sets row by row.
- Minimize memory consumption of returned data by the client (paging, iterators)



Query Service Capabilities

- Select which data (Fields) to retrieve, only retrieve that data
- Hierarchical queries may span multiple associated types (support for JOIN)
- Support for recursive types (Business Entity)
- Return all field types, fields from different objects in a single row
- Filter on all field types, i.e. Currency, Drop Downs, Strings
 - Which operators are available depends on data type
- Sorting on field
- Filter By Primary Associations
- Like/Contains, Long String search
- Paging of Results



Query Service Implementation

- New Service not based on OPSDK directly.
- **Below is for information only, not exposed to API developers**
- Queries are parsed using ANTLR 3.4 (3rd party library)
- Parsing results into “Query Object” that describes the request based on the tokens and syntax we defined. Query Object is then used to dynamically construct query on physical DB tables.
 - API define our custom SQL in ANTLR syntax
 - API internally defines Query Object and mapping of Query Object to physical model.
- Uses the “RV” Reporting Schema views for filtering **and** retrieval of values
 - Modeled after SQL from hierarchical findResources
 - Respects OpenPages security, including Record Level Security Rules



Query Syntax

- Like a SQL **SELECT** statement

SELECT [Columns]... identified by [Object Type].[Field Name]
Or [Object Type].* for all fields

FROM [Object Type]

Optional use JOINS for parents/children:

JOIN [Parent Object Type] ON PARENT([Object Type])

Optional any filters on fields

WHERE

[Object Type].[Field Name] LIKE 'abc' AND
[Object Type].[Field Name] = 123

Optional a sort order

ORDER BY



Query Example

- For Controls with Ineffective Controls:

- Query Control

SELECT

[SOXControl].[Resource ID],

[SOXControl].[Name],

[SOXControl].[OPSS-Ctl:Operating Effectiveness]

FROM [SOXControl]

WHERE

[SOXControl].[OPSS-Ctl:Operating Effectiveness] = 'Ineffective'



Old API vs New API: Search for child Issues, two levels below an Entity

```
// get ContentTypeIds
ContentTypeId entityId = ms.getContentTypeId("SOXBusEntity");
ContentTypeId riskEntityTypeId = ms.getContentTypeId("SOXProcess");
ContentTypeId riskTypeId = ms.getContentTypeId("SOXIssue");
// set id
ResourceSet set = (ResourceSet) rs.getLabelId(currentRP = rs.getLabel("CurrentPeriod").getLabelId());
// use hierfindresources to get Issues
// (Entity -> Process -> Issue)
HierFindResourceOptions hierFindResourceOptions = new HierFindResourceOptions();
hierFindResourceOptions.setIncludes(hierFindResourceOptions.getIncludes() + riskEntityTypeId);
hierFindResourceOptions.setIncludes(hierFindResourceOptions.getIncludes() + riskTypeId);
hierFindResourceOptions.setIncludeV(hierFindResourceOptions.getIncludeV() + riskEntityTypeId);
hierFindResourceOptions.setReturnTypes(hierFindResourceOptions.getReturnTypes() + riskEntityTypeId);
hierFindResourceOptions.setIsTopDown(hierFindResourceOptions.getIsTopDown() + riskEntityTypeId);
hierFindResourceOptions.setHonorPriority(hierFindResourceOptions.getHonorPriority() + riskEntityTypeId);
hierFindResourceOptions.setScopeResources(hierFindResourceOptions.getScopeResources() + riskEntityTypeId);
hierFindResourceOptions.setLabelId(hierFindResourceOptions.getLabelId() + riskEntityTypeId);
hierFindResourceOptions.setSetId(hierFindResourceOptions.getSetId() + riskEntityTypeId);
// set the content type criteria for Risk from
// Entity
ContentTypeCriteria entityObjectCriteria = new ContentTypeCriteria(entityTypeId);
entityObjectCriteria.setRecurseContentTypes(true);
hierFindResourceOptions.addCriteria(entityObjectCriteria);
```

```
//Process
```

```
ContentTypeCriteria reObjectCriteria = new
ContentTypeCriteria(
    riskEntityTypeId);
reObjectCriteria.setRecurseContentType(false);
```

```
String statement = "SELECT [SOXIssue].* " +
    "FROM [SOXIssue] " +
    "JOIN [SOXRisk] ON CHILD([SOXIssue]) " +
    "JOIN [SOXBusEntity] ON CHILD([SOXRisk]) " +
    "WHERE [SOXBusEntity].[Resource ID] = "+entityId +
    " ORDER BY [SOXIssue].[Name]";
IQuery query = queryService.buildQuery(statement);
ITabularResultSet resultset = query.fetchRows(SKIP NONE);
```

```
List<Resource> issues = (List<Resource>) rs
    .findResources(hierFindResourceOptions);
```



When to use QueryService

- Need to query single object type that meets some straight forward criteria
 - Like FLV filters
 - Only need to bring back the data you need
- Need to get fields from several related object types at once
 - All the data you want to use is contained in a single row
 - Such as simple Cognos Reports



Using QueryService API

- Interface: `com.ibm.openpages.api.service.IQueryService`
- **buildQuery**(`java.lang.String sqlStatement`)
- Query defined in `sqlStatement` using a String that follows the SQL statement syntax we specify
- Package for related public interfaces
 - **com.ibm.openpages.api.query**
- `buildQuery()` returns an instance of `IQuery`
- Can set additional parameters on `IQuery` interface
- To execute the query, use `IQuery.fetchRows()`
- Consume the results using `ITabularResultSet`



Example QueryService Java usage

```
IServiceFactory serviceFactory = ServiceFactory.getServiceFactory(context);  
//create Query Service  
IQueryService queryService = serviceFactory.createQueryService();  
String query = "SELECT [Resource ID], [Name] FROM [SOXIssue]";  
IQuery query = queryService.buildQuery(query);  
ITabularResultSet resultset = query.fetchRows(0);  
for (IResultSetRow row : resultset) {  
    // process rows...  
    for (IField field : row) { // process fields  
        ... }  
    ... }  
}
```



QueryService – Result Sets

- ITabularResultSet implements Iterable<IResultSetRow> interface
 - Can use directly in for() loops.
 - Iterates over any IResultSetRow found until no rows remain.
 - Transparently gets the next Page if IQuery's page size > 0 (0 is all rows)
- IResultSetRow itself implements Iterable<IField>, every column of the results is an IField instance.
- Same IField interface from ResourceService, can be cast to the data type to retrieve the value.



QueryService – Other Query Parameters

- IQuery interface has parameters which impact the query results returned
- **setHonorPrimary**(boolean honorPrimary)
 - Whether to only traverse primary parent associations or all associations
 - Default is false
- **setReportingPeriod**(IReportingPeriod reportingPeriod)
 - Default to Current Reporting period
- **setPageSize**(int pageSize)
 - Default is 50 rows per page, 0 is no paging
- **setMaxRows**(int maxRows)
 - To restrict how many total rows to return i.e. “only get first 10”
 - Default is unlimited



QueryService – Paging

- Can also iterate over the ITabularResultSet page by page using **getPages()** which returns a list of IPage interfaces. IPage is iterable for IResultSetRow
- Example using Paging:

```
IQuery query = queryService.buildQuery(query);  
// Execute query and fetch results.  
ITabularResultSet resultset = query.fetchRows(0);  
for (IPage page : resultset.getPages()) {  
    // process page...  
    for (IResultSetRow row : page) {  
        // process each row...  
        for (IField field : row) { // process each field... }  
    }  
}
```



Query Service Syntax

Overview for Query Service SQL syntax



SELECT clause

- **SELECT**
- The SELECT must contain one of the following:
- A comma separated list of one or more field system names: the query result set include in its row set all the fields specified in the SELECT clause.
- *: The query result set include in it's row set all the fields of the qualifying object type or the object type specified in the FROM clause.
- In addition, all column names specified must be valid field names. – *(Case Sensitive)*
- Followed by FROM Clause

```
SELECT [SOXProcess].[Resource ID],  
[SOXProcess].[Name],  
[SOXRisk].[Resource ID],  
[SOXRisk].[Name],  
[SOXControl].[Resource ID],  
[SOXControl].[Name],  
[SOXControl].[OPSS-Ctl:Operating Effectiveness]
```



Naming Conventions

- When Specifying Field Names, use the System Name
- Also the name provided by MetaData
- Format is:
 - Field Group:Field Name
- Exception is System Fields i.e. Name, Resource ID. The following are the names to identify these fields in the query
 - Resource ID
 - Comment
 - Created By
 - Creation Date
 - Description
 - Last Modification Date
 - Last Modified By
 - Location
 - Name
- All identifiers referencing the name of a field need to be delimited by a []. The use of the square bracket allows for names to contain spaces.
- [Resource ID], [OPSS-ISS:Status]



Naming Conventions

- **Unqualified column names**

- A column name does not need to be qualified if there is only one type included in the SQL query.
- The following example demonstrates the column name (Resource ID) which is not qualified.

```
SELECT [Resource ID]  
FROM [SOXBusEntity]
```

- **Qualified column names**

- A column name can be formed by qualifying its type name.
- The following example demonstrates the column name (Resource ID) which is qualified by its type (SOXBusEntity).

```
SELECT [SOXBusEntity].[Resource ID]  
FROM [SOXBusEntity]
```



FROM clause

- **FROM**
- The FROM clause identifies which Object type the query will be run against. The FROM clause must contain only one object type name that is available from the model.
- Optional - Followed by JOIN Clause
- Example:
FROM [SOXBusEntity]



JOIN clause

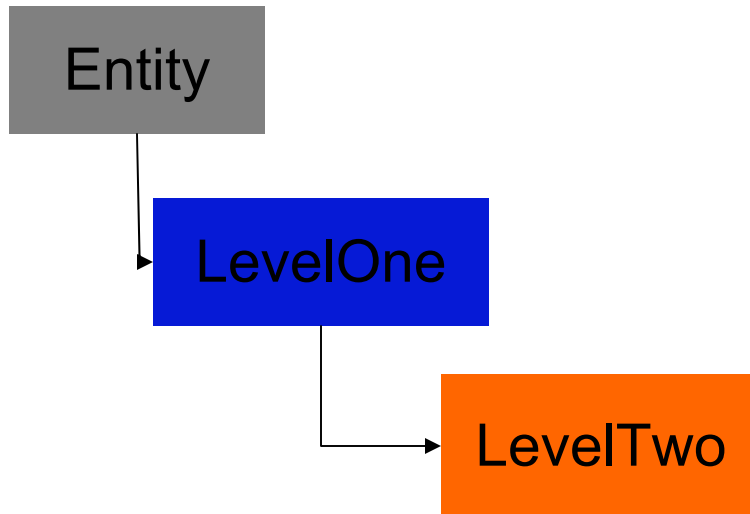
▪ JOIN

- The JOIN clause is used to combine fields from multiple objects types into a single result set. It is also how navigation between types is expressed. The JOIN clause differs slightly from the SQL-92 standard.
- The JOIN clause is expressed between object types with the ability to specify a CHILD/PARENT function to indicate the direction of traversal.
- Further more when defining a self JOIN (recursive), a level may be included in the function as an additional parameter.



JOIN Examples

- Simple three level hierarchy



Top down JOIN

- The top down traversal JOIN means you are joining starting from a parent to a child.
- Example:
- For the following example the SOXBusEntity type is a parent of the LevelOne type and LevelOne is the parent of LevelTwo

```
SELECT [SOXBusEntity].[Name], [LevelOne].[Name], [LevelTwo].[Name]  
FROM [SOXBusEntity]  
JOIN [LevelOne] ON PARENT([SOXBusEntity])  
JOIN [LevelTwo] ON PARENT([LevelOne])
```

FROM [SOXBusEntity]

- The FROM clause indicates the starting point in the traversal, SOXBusEntity
- JOIN [LevelOne] ON PARENT([SOXBusEntity])
- This join expresses the relationship between LevelOne and SOXBusEntity. In other words LevelOne is joining to SOXBusEntity via a PARENT association.
- JOIN [LevelTwo] ON PARENT([LevelOne])
- This join expresses the relationship between LevelTwo and LevelOne. In other words LevelTwo is joining with LevelOne via a PARENT association



Bottom Up JOIN

- The bottom up traversal JOIN means you are joining starting from a child to a parent.
- Example:
- For the following example the SOXBusEntity type is a parent of the LevelOne type and LevelOne is the parent of LevelTwo

```
SELECT [SOXBusEntity].[Name], [LevelOne].[Name], [LevelTwo].[Name]  
FROM [LevelTwo]  
JOIN [LevelOne] ON CHILD([LevelTwo])  
JOIN [SOXBusEntity] ON CHILD([LevelOne])
```

FROM [LevelTwo]

- The FROM clause indicates the starting point in the traversal, LevelTwo
- JOIN [LevelOne] ON CHILD([LevelTwo])
- This join expresses the relationship between LevelOne and LevelTwo. In other words LevelOne is joining with LevelTwo via a CHILD association.
- JOIN [SOXBusEntity] ON CHILD([LevelOne])
- This join expresses the relationship between SOXBusEntity and LevelOne. In other words SOXBusEntity is joining with LevelOne via a CHILD association.



Recursive Join

- The recursive traversal JOIN means you are joining performing a self join. In other words you are joining on a parent/child of the same type.
- Recursive traversal also extends the syntax of the PARENT/CHILD function.
- A level constraint may be included to specify which level to include in the join. When no values are specified in the function, it uses the default which is zero and will include **all levels**. Any other positive integer value will constrain the traversal for that level.

JOIN [SOXBusEntity] AS [child_be] ON PARENT([SOXBusEntity])

is equals to

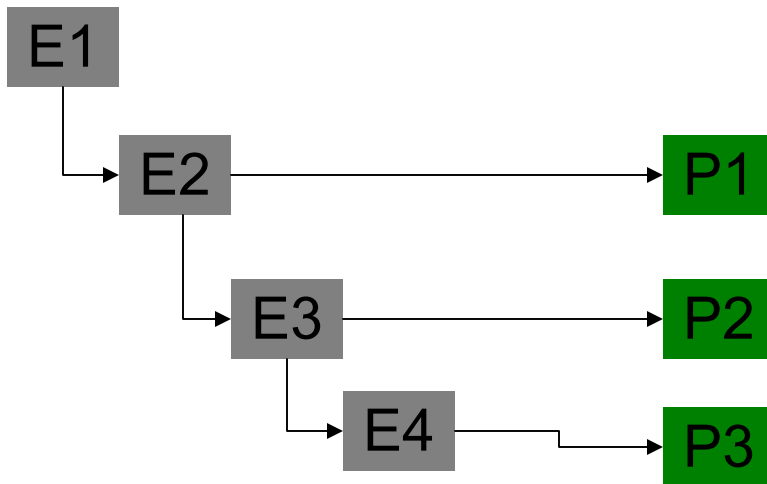
JOIN [SOXBusEntity] AS [child_be] ON PARENT([SOXBusEntity], 0)



Recursive – What are the levels?

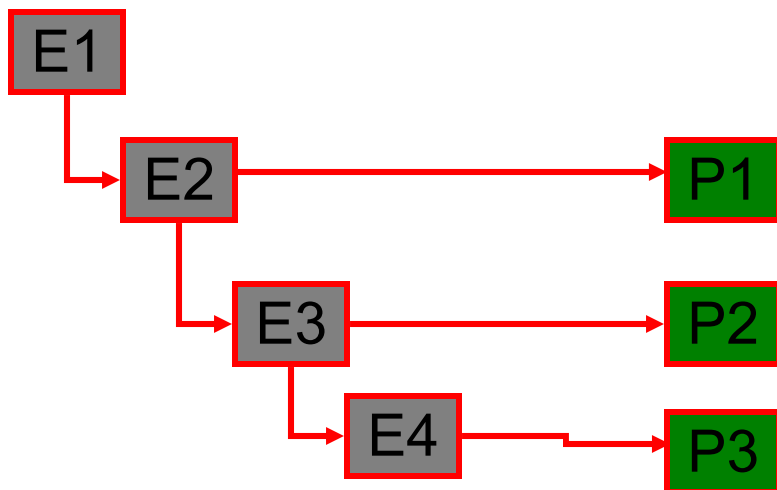
- Assume this hierarchy
- Go from E1 to Processes

```
SELECT [SOXProcess].[Resource ID]
FROM [SOXBusEntity]
...
WHERE [SOXBusEntity].[Name] = 'E1'
```



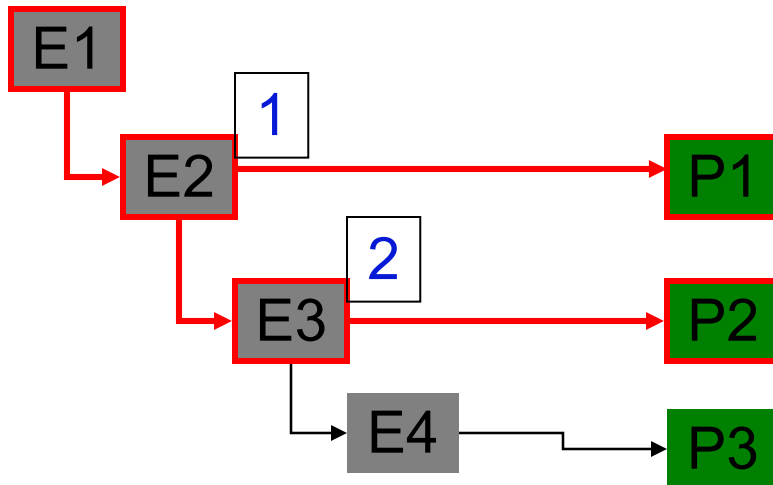
Recursive – What are the levels?

- JOIN [SOXBusEntity] AS [a] ON PARENT([SOXBusEntity], 0)
- All recursive entity levels considered



Recursive – What are the levels?

- JOIN [SOXBusEntity] AS [a] ON PARENT([SOXBusEntity], 2)
- Only two levels below E1 will be traversed



- This is the behavior of the SDK's FindResources setLevels()



WHERE clause

- **Comparisons permitted in the WHERE clause**
- Basic comparison predicate (=, <>, <, >, >=, <=) , IN predicate, LIKE predicate, IS NULL predicate,
- (AND), (OR), and negation (NOT) of predicates are also supported.
- The CONTAINS function is also supported for long text field when database indexing has been enabled.
- Depending on the Data Type of the Field, different predicates are supported. Refer to the Query package javadoc

WHERE [SOXIssue].[OPSS-Iss:Status] = 'Open' AND [SOXIssue].[OPSS-Iss:Additional Description] IS NOT NULL



LIKE Predicate

- The LIKE predicate is used in a WHERE clause to search for a specified pattern in a field value. Single quotes are used to enclose the search pattern. The "%" sign is used to define wildcards both before and after the pattern.
- For Date field type, LIKE predicate is not supported.
- For currency field type, the base amount of the currency is used for the pattern search.
- For enum/multi-enum field types, the enum value is used for the pattern search.
- All other field types are supported.

- Example:

```
SELECT [Resource ID], [Name]  
FROM [SOXBusEntity]  
WHERE [Name] LIKE '%entity%'
```



IN Predicate

- The IN predicate compares a value or values with a collection of values. The values for *expression1* and *expression2* in the IN predicate must be compatible. If NOT is specified, the result is reversed.
- For date field type, the Date string format needs to be specified.
 - Example:
[Creation Date] IN ('2013-05-16', '2013-05-18')
- For enum or multi-enum field types, a string representing the system name of the enum value delimited by single quotes may be used or the enum value identifier.
 - Example:
 - by enum value name:
[Shared:Shared Drop Down] IN ('New York', 'Rhode Island')
 - by enum value id:
[Shared:Shared Drop Down] IN (4215, 4218)
- For large text string field type, the IN clause is not supported.
- All other field types are supported.



Enum Type Fields

- As mentioned in the previous slides. For enum or multi-enum field types, a string representing the system name of the enum value delimited by single quotes may be used or the enum value identifier.



Currency

– Currency Field

- A Currency field is composite type that is composed of the following values:
 - base amount
 - base currency
 - exchange rate
 - local amount
 - local currency
- Result Column is represented as ICurrencyField with all these attributes.
- For the purpose of constraining the result set using a currency field, the base amount will be used as the constraining field. Other fields are ignored.
- Example:
- This example demonstrates the use of a currency field named Shared:Shared Currency field to apply a filter in the WHERE clause. The field base amount will be used for filtering.

```
SELECT [Shared:Shared Currency]  
FROM [LevelOne]  
WHERE [Shared:Shared Currency] = 5.0
```



Dates

- The Date string format supported by the Query Service is “yyyy-MM-dd”.

- Example:

```
SELECT [Creation Date]
FROM [SOXBusEntity]
WHERE [Creation Date] = '2013-05-16'
```



ORDER BY clause

- **ORDER BY**
- This clause **MUST** contain a comma separated list of one or more column names. All column names referenced in this clause must be valid “system name”.
- Only columns in the SELECT clause **MAY** be in the ORDER BY clause. Ordering is impacted by collation rules defined by the OpenPages schema.
- After each column name, may have the direction to sort: ASC , DESC

```
SELECT [SOXBusEntity].*  
FROM [SOXBusEntity]  
ORDER BY [SOXBusEntity].[Name] DESC
```



Advanced Features

- **UNION**
- Use this to union the results of two SELECT queries in a single execution. As in SQL, both SELECT clauses must return the same columns
- Example

```
SELECT [SOXIssue].[Resource ID]
FROM [SOXBusEntity]
JOIN [SOXIssue] ON PARENT([SOXBusEntity])
WHERE [SOXBusEntity].[Name] LIKE '%Test 01%'
UNION
SELECT [SOXIssue].[Resource ID]
FROM [SOXBusEntity]
JOIN [LevelOne] ON PARENT([SOXBusEntity])
JOIN [SOXIssue] ON PARENT([LevelOne])
WHERE [SOXBusEntity].[Name] LIKE '%Test 01%'
```



Advanced Features

- **Outer Join**

- Use to perform left outer join, default JOIN clause is inner join, this returns rows in cases where multiple objects are being joined and some rows do not have all parents/children

- Example

```
SELECT [SOXBusEntity].[Name], [LossEvent].[Name]
FROM [SOXBusEntity]
OUTER JOIN [LossEvent] ON
PARENT([SOXBusEntity])
```

Entity Name	LossEvent Name
Global Financial Services	LE_00123
Commercial Banking	LE_00123
Retail Banking	<i>null</i>



Query Example

- Illustration of joins, returning multiple types, filtering on Enum field
- For Risk as Direct child of Process with Ineffective Controls:
 - Query Gets Process -> Risk -> Control

```
SELECT [SOXProcess].[Resource ID],  
[SOXProcess].[Name],  
[SOXRisk].[Resource ID],  
[SOXRisk].[Name],  
[SOXControl].[Resource ID],  
[SOXControl].[Name],  
[SOXControl].[OPSS-Ctl:Operating Effectiveness]  
FROM [SOXProcess]  
JOIN [SOXRisk] ON PARENT([SOXProcess])  
JOIN [SOXControl] ON PARENT([SOXRisk])  
WHERE  
[SOXControl].[OPSS-Ctl:Operating Effectiveness] = 'Ineffective'
```



Query Example

- 1. Basic query, get all fields for an object type. Use [*] to avoid having to list every field. Ideally you would only bring back data that you will use, so use with care.

```
SELECT [KeyRiskIndicatorValue].[*] FROM [KeyRiskIndicatorValue]
```

- b) Add a WHERE clause filter

```
SELECT [SOXIssue].[*] FROM [SOXIssue] WHERE [SOXIssue].[OPSS-Iss:Status] = 'Open'
```

- c) And add multiple filters with AND/OR operators

```
SELECT [SOXIssue].[*] FROM [SOXIssue] WHERE [SOXIssue].[OPSS-Iss:Status] = 'Open' AND [SOXIssue].[OPSS-Iss:Additional Description] IS NOT NULL
```



Query Example

- 2. Hierarcical Search. To retrieve multiple object types in one query based on the associations. For example get KRI Values under a specific Risk(s) where KRI Red Threshold > 20

```
SELECT [SOXRisk].[Location], [KeyRiskIndicator].[Name],[KeyRiskIndicator].[OPSS-KRI-Shared:Red Threshold], [KeyRiskIndicatorValue].[Name], [KeyRiskIndicatorValue].[OPSS-KRIVa:Value]
```

```
FROM [SOXRisk]
```

```
JOIN [KeyRiskIndicator] ON PARENT([SOXRisk])
```

```
JOIN [KeyRiskIndicatorValue] ON PARENT([KeyRiskIndicator])
```

```
WHERE
```

```
([SOXRisk].[Name] = 'RI-0019' AND
```

```
[KeyRiskIndicator].[OPSS-KRI-Shared:Red Threshold] >= 20)
```

```
OR ([SOXRisk].[Name] = 'RI-0020')
```



Query Example

3. Using LIKE to search for text (Simple String or Long String fields)

```
SELECT [SOXRisk].[Name], [SOXRisk].[Description]  
FROM [SOXRisk]  
WHERE [SOXRisk].[Description] LIKE '%Physical Assets%'
```



Query Example

- 4. Recursive types, SOXBusEntity and direct SOXBusEntity children under root level entity 'Global Financial Services'

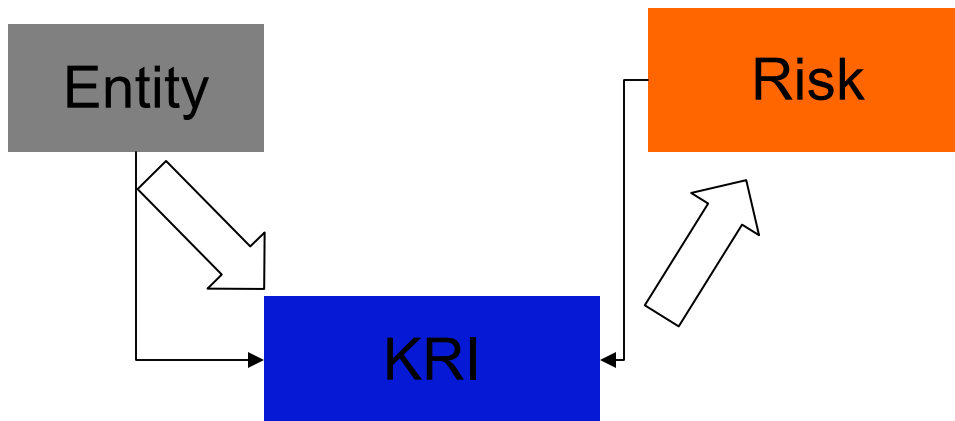
```
SELECT [SOXBusEntity].[Name], [child_be].[Name]
FROM [SOXBusEntity]
JOIN [SOXBusEntity] AS [child_be] ON PARENT([SOXBusEntity], 1)
WHERE [SOXBusEntity].[Name] = 'Global Financial Services'
```

- Note To return all levels, just remove '1' from the ON PARENT([SOXBusEntity]) clause.



Query Example

- 5. Multiple Direction Query
- KRI is child of Business Entity
- KRI is also child of Risk



Query Example

- Join Entity to KRI and KRI to Risk.

```
SELECT [SOXBusEntity].[Name], [KeyRiskIndicator].[Name],[SOXRisk].[Name]
FROM [SOXBusEntity]
JOIN [KeyRiskIndicator] ON PARENT([SOXBusEntity])
JOIN [SOXRisk] ON CHILD([KeyRiskIndicator])
WHERE [KeyRiskIndicator].[Name] LIKE 'KRI-0001'
```

Returns:

STRING_TYPE	STRING_TYPE	STRING_TYPE
Retail Banking	KRI-0001	RI-0019



Developing and Debugging

- Suggest that you first write queries and test them in provided sample JSP. See API samples “QueryTestJSP”

Enter query-service statement:

```
SELECT [SOXBusEntity].[Name], [LossEvent].[Name]
FROM [SOXBusEntity]
OUTER JOIN [LossEvent] ON PARENT([SOXBusEntity])
```

Notes: case-sensitive, uses system names for object types and fields. For field names use [Field Group.Field Name].
Example: SELECT [SOXIssue].[Resource ID], [SOXIssue].[Name] FROM [SOXIssue] WHERE [SOXIssue]

Running Test

Factories

Executing Query

Global Financial Services	_LE_0001
Global Financial Services	_LE_0002
Commercial Banking	LE-06-001
Commercial Banking	LE-06-003
Commercial Banking	LE-06-016
Commercial Banking	LE-06-017
Retail Banking	LE-06-000



Advanced: Finding the queries being run on the database

- For debug only, to find timings for different operations of the query service, and to find the physical SQL being on the database, you may enable additional logging
- In <OP_HOME>/aurora/conf/auroralogging.properties add the following lines

```
log4j.category.com.ibm.openpages.aurora.service.repository.resource=DEBUG#com.openpages.aurora.common.logging.Log4jEventLoggerLevel,
QUERYlog4j.category.com.ibm.openpages.service.query.impl=DEBUG#com.openpages.aurora.common.logging.Log4jEventLoggerLevel, QUERY
```

```
log4j.appender.QUERY=org.apache.log4j.RollingFileAppenderlog4j.appender.QUERY.File=${openpages.home}/aurora/logs/${openpages.instance}-
query.loglog4j.appender.QUERY.MaxFileSize=1024KBlog4j.appender.QUERY.MaxBackupIndex=10log4j.appender.QUERY.Threshold=DEBUG#com.openpages.aurora.common.logging.Log4jEventLoggerLevel
log4j.appender.QUERY.layout=com.openpages.aurora.common.logging.Log4jEventLoggerPatternLayoutlog4j.appender.QUERY.layout.ConversionPattern=%d{DATE}%n%-5p %b on
%h%n%m%n[%t](%F:%L)%n%n
```



Considerations

- Currently does not support all database functions or operators
- No aggregations – Min, Max, Count – No Group By
- No Other Database Functions in the syntax yet (Date/Time functions, String manipulation, or expressions).
- LIKE and = are case sensitive. No case insensitive search. Can use wildcards %, _
- Does not support nested “sub-select” in either SELECT, FROM or WHERE
- All join “path” must be explicitly given. Multiple alternate paths require additional UNION.
- Queries will execute against RT tables in the database. Filtering on a non-indexed column performance will be slow (same as Cognos report, or FLV filter)
- Should you find that limitations in QueryService do not allow you to express certain filters or expressions:
 - You can also use Cognos report method, available in APIs Application Service
 - Perform additional filtering/logic in Java with results.



Review Java API Services and methods.

Application operations



Review API Services and methods.

- **ApplicationService**
 - Misc OpenPages operations, Cognos reports
- **SecurityService**
 - Users, Groups, Role based security
- **WorkflowService**
 - Workflow sub-system interactions: Jobs, Tasks, etc.
- **AuditService**
 - GRC Object Change History



Application Service

- Interface: `com.ibm.openpages.api.service.IApplicationService`
- Miscellaneous methods for working with Resource instances
 - Hierarchical Copy
 - Hierarchical Move
 - Lock
 - Unlock
- Invoking Cognos Reports from OP
- Package for Application Service classes and interfaces
 - **`com.ibm.openpages.api.application`**



Application Service - Copy

- **copyToParent**(Id parentId, java.util.List<Id> childrenToCopy, CopyObjectOptions copyOptions)
 - Performs same operation as UI's 'Copy an existing...' from the parent object
 - Returns java.util.List<IGRCObject> of copies!
 - Copies one or more GRCObjects to a parent GRCObject based on the CopyObjectOptions
 - Copies a "tree" of Objects to the new parent. All primary children are copied by default. Can limit using the Options
 - Returns the copies made for each of the childrenToCopy that are specified i.e. the "roots" of the trees being copied.
 - Honors the autonaming setting for Copy for the root objects if applicable.
-
- The arguments must be IGRCObjects, folders cannot be copied with this call. The children Ids must be of Types that have a parent association enabled with the Parent Object.



Application Service - Copy

- **CopyObjectOptions**
- Supports specifying whether to include children in the copy and an explicit list of child Object Types to be copied.
 - **setChildrenTypesToCopy**(java.util.List<ITypeDefinition> childrenTypesToCopy)
- Supports the setting of a behavior to take when a naming conflict occurs as a result of the copy.
 - **setConflictBehavior**(CopyConflictBehavior conflictBehavior)
- Default Options:
- Static method: **getDefaultOptions()**
 - includeChildren = true
 - ChildrenTypesToCopy = ALL
 - conflictBehavior = CopyConflictBehavior.CREATECOPYOF



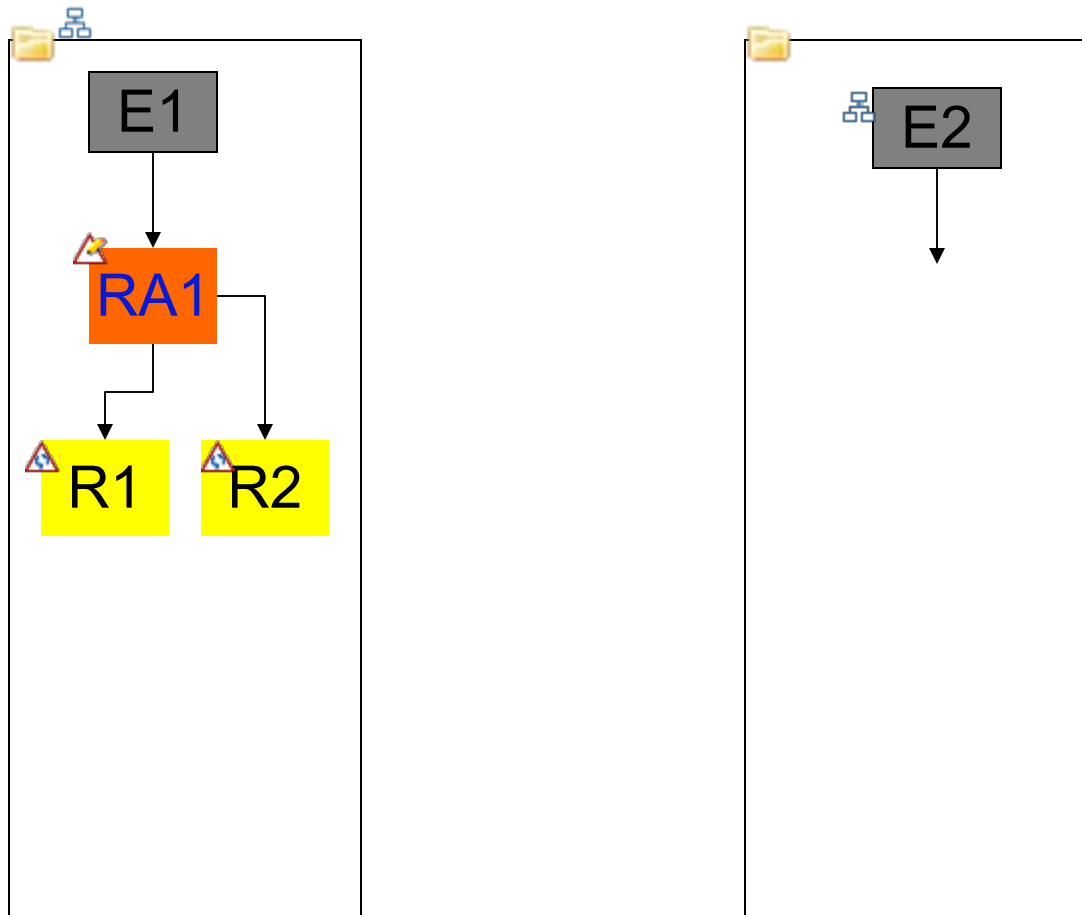
Application Service – Hierarchical Move

- **moveGRCOBJECT**(Id parentId, java.util.List<Id> childrenToMove, MoveObjectOptions moveOptions)
- This performs the same move operation as the user interface when choosing 'Move To' from a Filtered List or Folder View and selecting a new Parent object.
- Moves one or more GRCOBJECTs to a new parent GRCOBJECT.
 - Updates folder location of the moved children to the parent's folder path,
 - disassociates from the previous primary parent, and creates association to the new parent.
 - Any primary descendants of the children to be moved will remain associated to the moved children, their folder locations will also be updated.
 - Secondary association descendants will **not** be moved.
 - The behavior of which descendants will be moved as well is configurable through the MoveObjectOptions parameter.
- Reminder: for purely Folder based moves, use IResource.setParentFolder() + saveResource()



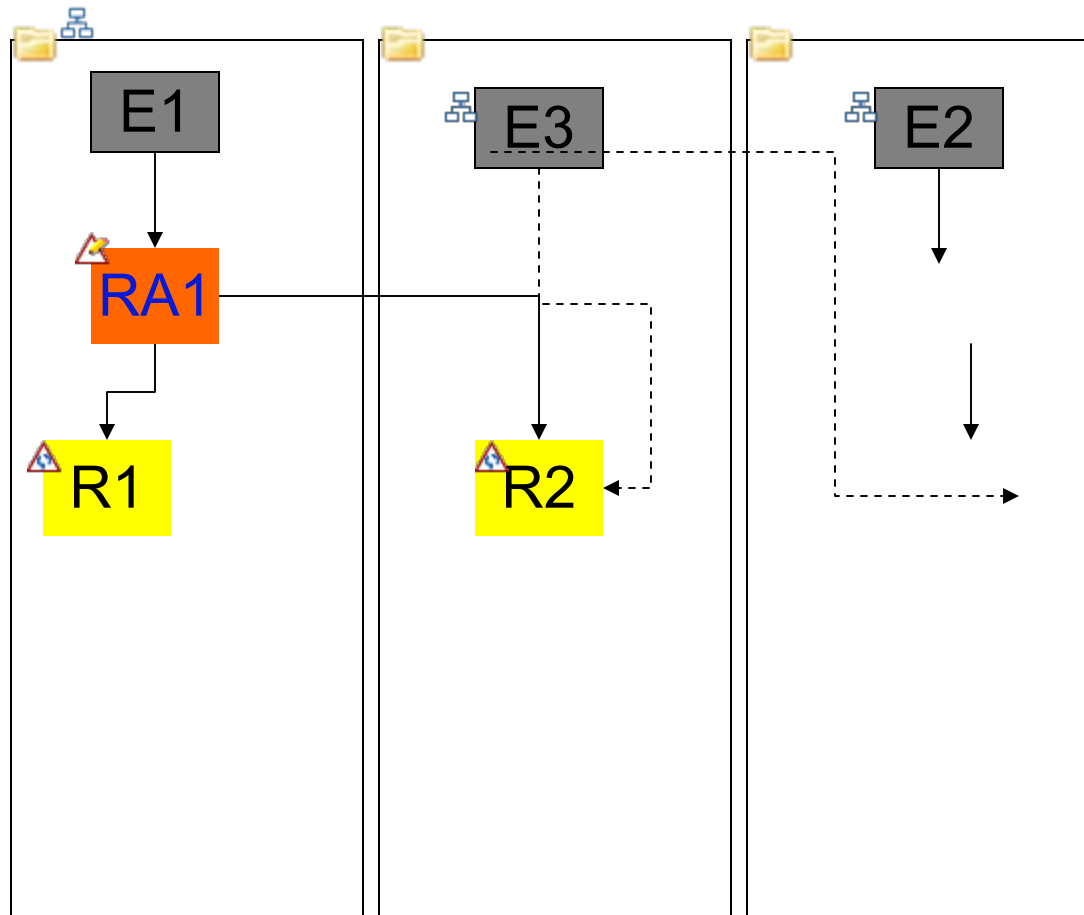
Hierarchical Move - Examples

- Basic Move: Risk Assessment with children to another Entity



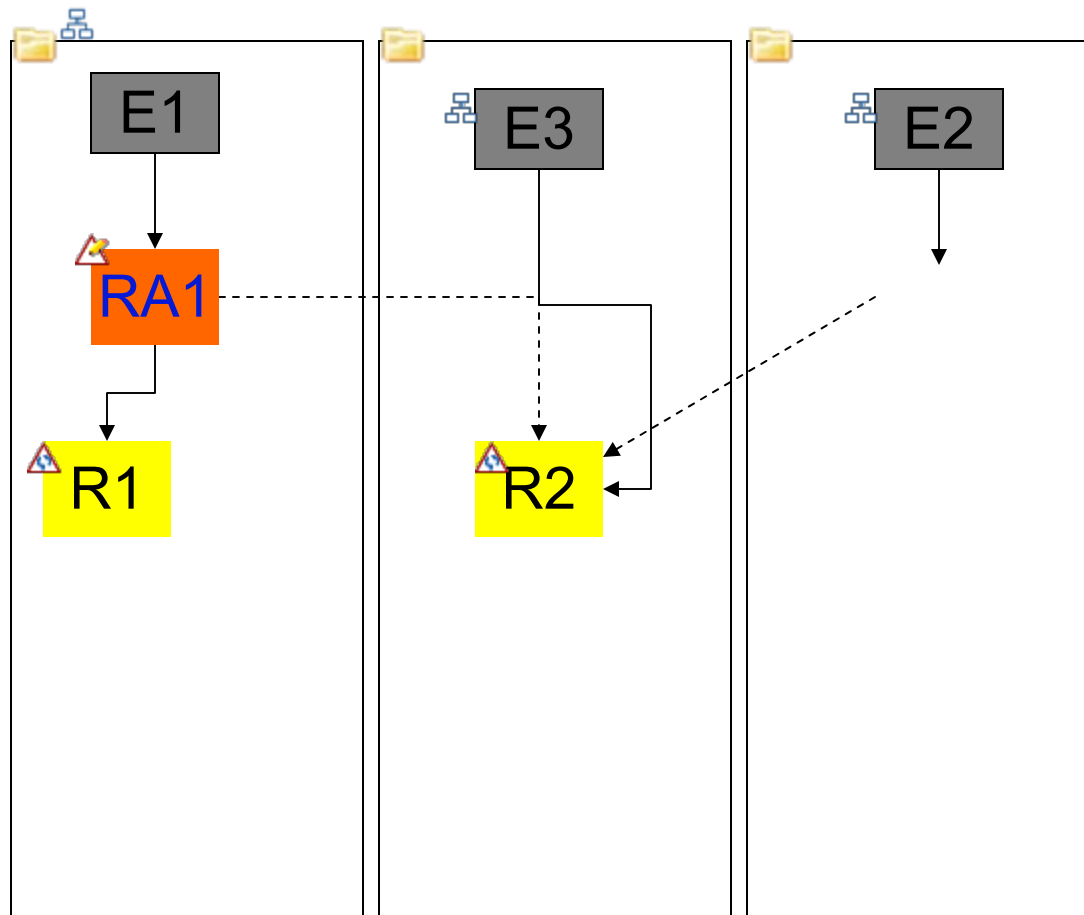
Hierarchical Move - Examples

- Basic Move+: R2 is still primary associated to RA1



Hierarchical Move - Examples

- Basic Move++: R2 is secondary associated to RA1, primary in E3 hierarchy



Application Service – Hierarchical Move

- **MoveObjectOptions**
- Supports specifying an explicit list of child Object Types to be moved. Supports the setting of a behavior to take when a naming conflict occurs as a result of the move.
 - **setChildrenTypesToMove**(java.util.List<ITypeDefinition> childrenTypesToMove)
- Supports the setting of a behavior to take when a naming conflict occurs as a result of the move.
 - **setConflictBehavior**(MoveConflictBehavior conflictBehavior)
- Default Options:
- Static method: **getDefaultOptions()**
 - includeChildren = true
 - childrenTypesToMove = ALL
 - conflictBehavior = MoveConflictBehavior.ERROR



Application Service – Hierarchical Move

▪ Example

```
MoveObjectOptions moveOptions = MoveObjectOptions.getDefaultOptions();  
Id parentId = parent1.getId();  
//Move 4 selected children to this parent  
List<Id> childrenToMove = new ArrayList<Id>(4);  
childrenToMove.add(selectedChild1);  
childrenToMove.add(selectedChild2);  
childrenToMove.add(selectedChild3);  
childrenToMove.add(selectedChild4);  
appService.moveGRCObject(parentId, childrenToMove, moveOptions);  
//same Ids after the Move for all objects, nothing new returned.
```



Application Service Locking

- Can Lock and Unlock GRCObjects via the IApplicationService
- Can test if an GRC Object is locked using IGRCObject.isLocked()

- Locking a GRC Object

lockGRCObject(Id grcObject) - Applies direct (red) lock to the GRC Object. Uses Registry Setting: Lock Child Types to cascade the inherited locks

- There are two different unlock calls, which are distinct in the UI as well:

removeAllLocks(Id grcObject) - Removes all locks, direct and inherited, for a tree of GRC Objects starting from the specified GRC Object. Searches for any locks on the tree of the GRC Objects beneath the root and removes them.

unlockGRCObject(Id grcObject) - Unlock a locked GRC Object. Any inherited locks on children of this GRC Object will be removed automatically. If the GRC Object is locked with inherited locks as well a direct lock, the inherited locks will **not** be removed.



Application Service Cognos Reports

- Application Service capability to programmatically call Cognos report from OpenPages code.
- This is referred to as the Cognos Report or the “Batch Job” method for querying data in OpenPages.
- Can be used as data retrieval mechanism where Openpages query is not sufficient for your use case
- Could also just return the report contents for display or as attachment content
- Encapsulates direct dependencies on aurora and Cognos SDK code
- For tabular reports we have new reader mechanism to iterate over report rows



Application Service Cognos Reports

invokeCognosReport(java.lang.String reportPath, CognosOutputFormat format, IReportParameters parameters)

- Invokes a Cognos Report programatically and have the resulting output file returned in a format that can be read by the returned InputStream.
- reportPath is the path in Cognos for the report to call:
 - Example for report in 'Public Reports/MY_FOLDER/My Report Name' use:
"/content/folder[@name='MY_FOLDER']/report[@name='My Report Name']"
- Format is one of these: CSV, HTML, PDF, XLWA, XML
 - XLWA = Excel 2002 - Microsoft Excel Web Archive (XLWA) format



Application Service Cognos Reports

IReportParameters **getParametersForReport**(java.lang.String reportPath)

- Makes remote call to Cognos for the specified report path and asks for any parameters defined in the report. Returns IReportParameters object that contains all parameters a report expects for inputs. .
- IReportParameters is a structure that contains all parameters and their values. You may set the value for a parameter using the **setParameterValue**(java.lang.String parameterName, java.lang.String... values)
 - A parameter may have one or more values, e.g. a multi-select prompt can have multiple.
 - Refer to JavaDoc for specific formats, especially for date prompts.
- To know what are the possible parameters the report needs use **getAllParameterNames**()
- For info on what data type a prompt expects, use **getParameter**(java.lang.String parameterName)



Application Service – Basic Invoking Report

- Example run a report as PDF with two parameters

```
final CognosOutputFormat format = CognosOutputFormat.PDF

final String reportPath =
    "/content/folder[@name='APITest']/report[@name='Sample-Users-Report2']";

IReportParameters params =
    appService.getParametersForReport(reportPath);

params.setParameterValue("rpid", "-1" );

params.setParameterValue("dateprompt", "2013-12-07T12:20:46.275" );

InputStream reader = appService.invokeCognosReport(reportPath,
    format,params);

//read from reader...
```



Application Service Cognos Row Based Reports

invokeCognosRowReport(java.lang.String reportPath, IReportParameters params)

- Similar to invokeCognosReport
 - Specify reportPath, and IReportParameters for the report
 - Report must tabular i.e. a List report.
- Report implicitly calls Cognos with a format that allows us to consume it more easily and give the caller structured data without doing all the String parsing from raw Cognos output formats
- Returns a **ICognosReportRowReader**
 - The report output can be read row by row, with the values for each column parsed to be consumed as a native Java object type.
- ICognosReportRowReader implements Iterable interface, you use directly in new for() loops
 - Iterate over the rows, each row is a **ICognosReportRow**
 - From the ICognosReportRow getValue by column or by name, or iterate over every column



Application Service Cognos Row Based Reports

- Example run a report and consume the rows

```
//first get report parameters etc..
```

```
//call report and get row reader
```

```
ICognosReportRowReader reader = appService.invokeCognosRowReport(reportPath, reportParams);
```

```
for (ICognosReportRow row : reader) {
```

```
    //iterate over columns in the row
```

```
    for (ICognosReportColumnValue value : row) {
```

```
        System.out.println(value.getDataType());
```

```
        //value is a java.lang.Object but may be cast to something more specific
```

```
        System.out.println(value.getValue());
```

```
    }
```

```
}
```



Application Service Cognos Row Based Reports

- Other approach, specifically getting certain columns from the Row – if you don't want to iterate over the columns
- Retrieve columns from the row individually, by name or index:

//index

```
BigDecimal userid = (BigDecimal) iCognosReportRow.getColumnValue(0).getValue();
```

```
String username = (String) iCognosReportRow.getColumnValue(1).getValue();
```

//by name

```
Short enabled = (Short)
```

```
iCognosReportRow.getColumnValueByName("USER_ENABLED").getValue();
```

```
System.out.println("userid: "+userid+", username: "+username+", isenabled: "+enabled);
```

- Col Name is available in report output MetaData, or by looking at Query subject in Cognos.



Application Service Cognos Row Based Reports

- Other features from ICognosReportRowReader
 - Get Metadata for the columns returned by the report you ran:
 - Column Data Type – these are a subset of XML Schema Data Types that Cognos outputs – *not necessarily just OpenPages Field types!*
 - Column Name – name of the query item in the main query of the report
- Can also get the data type from every ICognosReportColumnValue
 - Refer to the JavaDoc for ICognosReportColumnValue.getValue() method on what Object is returned based on the data type of the column in the report output for all possible data types Cognos may output.
 - E.g. double is java.lang.Double, dateTime is java.util.Calendar
 - Conversions based on javax.xml.bind.DatatypeConverter
 - getValue() will return null if the value is null.



Security Service

- Interface: `com.ibm.openpages.api.service.ISecurityService`
- Access to the Security information in OpenPages
 - Users
 - Groups
 - Roles
- Can create and update users, make role assignments
- Package for related public interfaces
 - **`com.ibm.openpages.api.security`**



Security Service key interfaces

- **ISecurityPrincipal**
 - Similar to the Actor concept in the SDK
 - Base interface for both Groups and Users with common info
 - Not directly created, but some interfaces will return this when it could either be a user or a group such as **IGroup.getAllMembers()**

- Name, App Permissions, is enabled, is deleted

- Operations you can perform
 - Enable
 - Disable



Security Service key interfaces

- IUser – extends ISecurityPrincipal
 - Represents a single user
 - User information, first name, last name,
 - password expiry
 - Role assignments

- Operations you can perform with this interface
 - Change password – For OP Security authentication
 - Lock/Unlock
 - Update fields



Security Service key interfaces

- IGroup – extends ISecurityPrincipal
 - Represents a group and its members
 - Group information
 - Sub groups, user members
 - Parent group
 - Role assignments for the group

- Operations you can perform with this interface
 - Add Member
 - Remove Member
 - Update fields



Security Service key interfaces

- IRoleTemplate
- Interface for a role template for grouping object permissions and application permissions. This is a central piece of the OP security model. Role templates are assigned to users and groups for specific folders.
 - Role Information: Name, ID
 - What Role Type object: either Business Entity or other if extended security model
 - Object Permissions – what access for each object type
- Operations you can perform with this interface
 - Enable / disable
 - Lock / Unlock
- Do not support updates or creates of Role Templates



Security Service

- Use SecurityService to perform create/delete operations on IUser, IGroup Objects
- Use SecurityService to retrieve RoleTemplates/ApplicationPermissions and Assign Role/Roles to IUser.
- Has various other security related helper functions
 - Get Current User for this Session
 - Get Current Password Policy (system wide)
 - Whether you have certain permissions



Security Service – Create a User

- Example to create a new user. Need to set up a UserInfo

```
UserInfo user = new UserInfo(uniqueUserName,  firstName,  lastName,
    middleName, password, Locale.US);

user.setDescription(description);

user.setEmail(emailAddress);

IUser createdUser = null;

List<IGroup> groupList = new ArrayList<IGroup>(1);

IGroup sc_securityService.createUser("OpenPagesApplicationUsers");

// creates a user in the group

createdUser = securityService.createUser( user, gList);
```



Security Service – Find a User's Role Assignments

- Example to get the Role Assignments for a User

```
IUser bill = secService.getUser("bill");

Iterator<IRoleAssignment> assignments = bill.getRoleAssignments();
while(assignments.hasNext()){

    IRoleAssignment assignment = assignments.next();

    System.out.println("Role:
"+assignment.getRoleTemplateName()+"-
"+assignment.getFolder().getPath());
}
```

Role Assignment Get Folder will return an Entity *Folder*

/_op_sox/Project/Default/BusinessEntity/SomeCompany



Security Service – Assign Roles

- Make a new role assignment to a user for a entity security context point

```
//get Entity's Parent Folder as Security Context point
IGRCObject busEntityObject =
    resourceService.getGRCObject(entityPath);

IFolder securityContext = busEntityObject.getParentFolder();

//get the role template to be assigned
IRoleTemplate rt = secService.getRoleTemplate("AuditRole");

IUser testUser = secService.getUser("bill");

//make role assignment
secService.assignRole(testUser, rt, securityContext);
```



Workflow Service

- Interface: `com.ibm.openpages.api.service.IWorkflowService`
- Service interface for managing Business Process Engine components like
 - Job
 - Task
 - Job Type
- Retrieve Jobs, Tasks by Id, or by criteria
- Retrieve Job Type
- Create a Job to submit an `GRCObject` to a workflow
- Package for related public interfaces
 - **`com.ibm.openpages.api.workflow`**



Workflow Service – Job Interface

- IJob represents an instance of a workflow job in OpenPages
- Has access to all the info about the Job
- Association to a GRC Object is through the attachment capability of Jobs
- E.g. to submit a GRC Object with a Job you use **addAttachment(IResource resource)**
- To get to related Tasks assigned to user use
 - **getActiveTasks()**
- To perform operations on the job, there are methods on IJob interface
 - **queue()** – queue the start, this throttles job submission and adds delays
 - **terminate(java.lang.String comment)**
- If changes happen on server while you have a reference you can call **refresh()** to get latest state



Workflow Service – Task Interface

- ITask represents an user assigned work item in a workflow job in OpenPages
- Has access to all the info about the Task
- Get Task Comments **getComments()**
- Get User Assignees **getAssignees()**
- The list of possible choices (arrows in Workflow Studio) for that task: **getChoices()**

- Operations you can perform on the ITask
 - **accept**(java.lang.String comment)
 - **decline**(java.lang.String comment)
 - **reAssign**(java.lang.String comment, java.util.List<IUser> users)
 - **complete**(java.lang.String comment, java.lang.String nextAction)



Workflow Service – Submit a Job

- Example of submitting a Job using WorkflowService API

```
// get the JobType - this should be loaded
IJobType basicJobType = workflowService
    .getJobType("ApprovalJob");

//create the Job
IJob newJob = workflowService.createJob(basicJobType);
//add the attachment to submit to this job
newJob.addNamedAttachment(targetGRCObject, typeDefinitionName);

//queue the job
newJob.queue();
```



Workflow Service – Complete a user's task

```
Iterator<ITask> taskIt = runningJob.getActiveTasks();
ITask task = null;
//find the active task I want to complete
while (taskIt.hasNext())
{
    task = taskIt.next();
    if (expectedTaskName.equals(task.getName()))
    {
        // make sure to retrieve the latest task info.
        task.refresh();
        break;
    }
}
//accept the task on behalf of the assignee if not accepted
if(!TaskState.ACCEPTED.equals(task.getState())){
    task.accept("Accept comment");
}
//complete the task, making a choice on behalf of assignee
task.complete("Complete task comment", "Approve");
```



Audit Service

- Interface: `com.ibm.openpages.api.service.IAuditService`
- Retrieve the Audit Trail or “Change History” for GRC Objects
 - Field Changes
 - Association Changes
- Package for related public interfaces
 - **`com.ibm.openpages.api.audit`**



Audit Service Logs type

- AudiLogs retrieved are of 2 types IFieldAuditLogs and IAssociationAuditLog
 - IFieldAuditLog returns the Logs for field changes on the specified object.
 - IAssociationAuditLog returns the association changes performed on the specified object.

 - Both extend common interface IAuditLog which has basic getters for all audit trail events:
 - **getId()** *
 - **getName()** *
 - **modifiedBy()**
 - **modifiedDate()**
- * Depends on which log type what these getters mean



Audit Service Field Changes

- **getFieldAuditLog**(Id id) - Retrieves a List of FieldAuditLogs for a given GRCOBJECT Id.
- In addition to values found in `IAuditLog` provides additional data about field value changes
 - **getId()** FieldDefinition Id
 - **getName()** Field Name
 - **getOldValue()** – Old Value, see below
 - **newValue()** - New Value, see below
- The value returned depends on the data type of the Field:
 - “Numerics”: Currency Field, Decimal Field, Integer Field - Object returned is of type `java.lang.Double`.
 - Boolean Field - Object returned is of type `java.lang.Boolean`.
 - Date Field - Object returned is of type `java.util.Date`.
 - Enum Type - Object returned is of type `java.lang.String`.
 - Multi Enum Type - Object returned is of type `List`.
 - All other Types - Object returned is of type `java.lang.String`.



Audit Service Field Changes

- **Currency information is a slightly more complex**
- Due to the way that the data is persisted in the OP Audit Trail, we can only expose the currency sub-properties that have changed.
- Example of a Currency log entries

Audit Log 1

Name: Risk Appetite_BA

Id: 1226

Old:

New: 1000.0

ModBy: OpenPagesAdministrator

ModDate: 2013-06-12 16:13:04.0

Audit Log 2

Name: Risk Appetite_ER

Id: 1228

Old:

New: 1.0

ModBy: OpenPagesAdministrator

ModDate: 2013-06-12 16:13:04.0

Etc...



Audit Service Association Changes

- **getAssociationAuditLog**(Id id) - Retrieves a List of IAssociationAuditLogs for a given GRCOBJECT Id.
- Based on child associations.
- In addition to values found in IAuditLog provides additional data about field value changes
 - **getId()** Child GRCOBJECT Id that whose association to the current object was changed
 - **getName()** Name of the Child GRC Object
 - **getObjectType()** – Type of Child GRC Object
 - **getStatus()** – What was the change of association
 - ADDED, DELETED, ADDED_PRIMARY, REMOVED_PRIMARY



Audit Service - AuditLogFilter

- **Retrieve only certain logs from Audit Trail either Fields, Association or both**
- **Filter by Date range – or – Filter by Reporting Period**
- **Filter by ObjectType for Association changes**
- **Filter by FieldDefinition for Field changes**

```
ITypeDefinition issueType=ms.getType("SOXIssue");  
AuditLogFilter filter = new AuditLogFilter();  
filter.getAssociationFilter().getTypeFilters().add(issueType);  
filter.setStartDate(somedate);  
filter.setEndDate(today);  
List<IAuditLog> auditLogs = auditService.getAssociationAuditLog(id, filter);
```



Audit Service variants

- **Retrieve logs with filters**
 - **getAssociationAuditLog**(Id id, AuditLogFilter filter)
 - **getAuditLog**(Id id, AuditLogFilter filter) - this is both Field and Association logs together
 - **getFieldAuditLog**(Id id, AuditLogFilter filter)
- **Retrieve logs with no filters (all logs)**
 - **getAssociationAuditLog**(Id id)
 - **getAuditLog**(Id id)
 - **getFieldAuditLog**(Id id)



Solution Changes.

Workflow Actions, Triggers



Solutions Changes

- Workflow Actions and Triggers have new abstract base classes to extend to leverage the new API within those contexts.
- Same API code should work in both contexts (along with JSPs Workflow Actions pretty much the same as with OPSDK)
- Triggers now use different Event Framework which wraps the SDK methods being called and their raw parameters.



Workflow Actions

- Custom Java Actions that plug into IBPM Workflow Engine to perform custom logic at specific nodes within a Workflow Job Type
- Workflow Author will use them to do more functionality than the out of box actions supply.
- All existing Workflow Actions will continue to work unchanged.
- To write actions with the new API, there are some differences than the legacy conventions for writing actions.



Workflow Actions

1. New classes must extend `com.ibm.openpages.api.workflow.action.CustomJavaActionBase`

```
public class TestQueryAction extends CustomJavaActionBase {
```

- CustomJavaActionBase extends the SDK WorkflowAction base class. It adds new methods:
- `getContext()` to get the API Context within a Java Action method for the session used in workflow actions.



Workflow Actions

2. For every method that you want to be called directly as an action you would add the method as you do today, as a public method, with any arguments as well as the Workflow SecurityEnactmentContext

- **Example**

public String getUsername(String userId, ServerEnactmentContext sec) **throws** Exception {



Workflow Actions

3. Put the custom code within a try / catch / finally block

```
Context context = null;
boolean errorOccurred = false;
String userName = "";
try {
    // get the API Context
    context = getContext();
    //do stuff
} catch (Exception e) {
    // track if exception occurred
    errorOccurred = true;
    // handleException for workflow
    handleException(e, sec);
    throw e;
} finally {
    // always handle clean up of context in finally {}
    handleContextCleanUp(context, errorOccurred);
}
```



Workflow Actions

4. Always put new method **handleContextCleanUp**(Context context, boolean errorOccurred)
In the finally block
 - Returns or Removes the OpenpagesSession from the WorkflowAction's session pool. If errorOccured is true then the session will be removed. This method should be used in the finally block of your workflow action method.
 - Must call this method if using getContext().
 - Usage:

```
//... finally{ handleContextCleanUp(context, errorOccurred); }
```



Trigger Framework

- Review
- Feature backwards compatibility, no change to existing trigger framework rules or actions, or XML
- Requires Adapters where Trigger Actions and Rules are written using the New API
 - More explicit event framework: Create Event, Update Event, Associate Event with appropriate information.
- In General, Events triggered by calls in the New API will fire triggers even if they were written in old SDK.



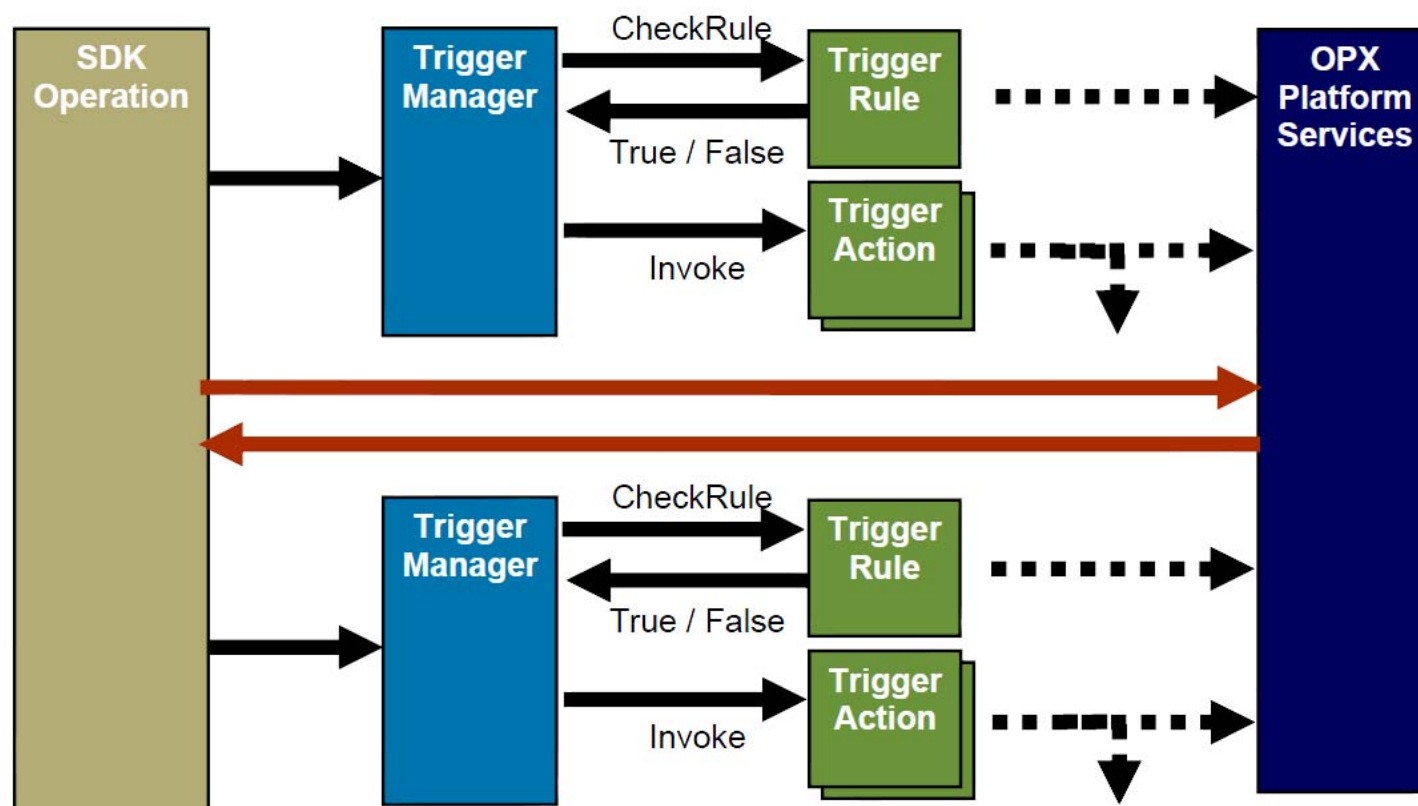
Trigger Framework - Recap

- The current form of a trigger is a piece of code that can be injected before or after the execution of an operation is performed on the OpenPages platform. This piece of code can perform anything that can be written in Java.
- Are only available for SDK operations*
- Are executed within the existing transaction boundary of the original operation.



Trigger Framework Recap

Flow Diagram



Trigger Framework – New API approach

- Each operation is represented by an event. See **com.ibm.openpages.api.trigger.events**
- Events extend from AbstractEvent
- From this Event you can get
 - **getContext()** - the API context object
 - **getPosition()** - PRE or POST
 - **getTriggerEventType()** – Java Enum representing the event type
- There is an Event Type for all current supported trigger operations
- The actual Event object will contain more info about the event that is relevant such as Ids or Objects



Trigger Framework – New API approach

- Each operation is represented by an event. See **com.ibm.openpages.api.trigger.events**
- Events extend from AbstractEvent
- From this Event you can get
 - **getContext()** - the API context object
 - **getPosition()** - PRE or POST
 - **getTriggerEventType()** – Java Enum representing the event type
- There is an Event Type for all current supported trigger operations
- The actual Event object will contain more info about the event that is relevant such as Ids or Objects
 - CreateResourceEvent
 - **IResource getResource()** – returns IResource being created



Trigger Framework – New API approach

- Other Event Types may represent different information so have different getters
- **AssociateResourceEvent**
 - **List<Id> getChildren()**
 - **List<Id> getParents()**
 - **boolean isSingleParent()** – if true then one parent to many children, else many parents to one child.



TriggerFramework - Rules

- New Rules will handle Events. Extend **com.ibm.openpages.api.trigger.ext.DefaultRule**
 - The DefaultRule has an isApplicable() method for every event type which always return false.
 - Your new Rule can override any of the Events that you wish to capture with this Rule class and implement logic in the isApplicable() method based on the Event details
- See samples/Triggers – ContentTypeMatchRule

@Override

```
public boolean isApplicable(CreateResourceEvent event) {  
    IResource resource = event.getResource();  
    if(resource.isFolder()) {  
        return false;  
    } else {  
        return evaluate((IGRCObject)resource);  
    }  
}
```



TriggerFramework – Event Handlers

- Instead of Actions, we have EventHandler classes. One or more Event Handlers can be configured per Rule. **com.ibm.openpages.api.trigger.ext.DefaultEventHandler**
 - The DefaultEventHandler has an `handleEvent()` method for every event type which always return false.
 - Your new handler Override the specific `handleEvent()` methods only for the Events that your trigger is concerned with.
- See samples/Triggers – SetCurrentDateAction

```
@Override  
public boolean handleEvent(UpdateResourceEvent event) {  
    IGRCOBJECT object = (IGRCOBJECT)event.getResource();  
    setCurrentDate(object);  
    return true;  
}
```



TriggerFramework – New Configuration

- To define a Rule with Event Handlers in the new framework in the same existing `_trigger_config.xml` file add `grcTrigger` element with one `<rule>` and multiple `<eventHandler>`. `<attribute>` tags are determined by the rule or event handler config requirements.

```
<grcTrigger name="Risk Assessment create trigger" event="create.object"
  position="PRE">
  <rule class="com.ibm.openpages.api.trigger.ext.ContentTypeMatchRule" >
    <attribute name="contentTypeName" value="RiskAssessment" />
  </rule>
  <eventHandler class="com.ibm.openpages.api.trigger.ext.DateValidationAction">
    <attribute name="start.date.field" value="OPSS-RA:Start Date" />
    <attribute name="end.date.field" value="OPSS-RA:End Date" />
  </eventHandler>
  <eventHandler class="com.ibm.openpages.api.trigger.ext.SetCurrentDateAction">
    <attribute name="current.date.field" value="OPSS-RA:Next Assessment
Date" />
  </eventHandler>
</grcTrigger>
```



TriggerFramework – New Error messages

- From AbstractEventHandler (parent of DefaultEventHandler)

throwException(java.lang.String message, java.util.List<java.lang.Object> parameters, java.lang.Throwable cause, Context apiContext)

Allows the trigger to create a formatted exception object that is thrown stopping the execution of the current operation.

- Example using String message instead of Application Text:

```
throwException("Start date before end date",Collections.emptyList(),null,event.getContext());
```

- If an Application Text key were provided for the first argument, will return the localized app text label, and format with any parameters from the second argument.
- Thrown from a PRE Event Handler will halt the operation and display the OP-0072 error with your message



TriggerFramework – Notes on Setting Values

- In a create or update event handler, you are modifying the values of the object being saved.
- Do not need to call `IResourceService.saveResource()`;
- For other events where you want to update an object that is part of the main operation, you would need to call `saveResource()`.
- If you want to update a parent or other unrelated resource from within the event handler then you will need to call `saveResource()`
- Note that as with current trigger framework, if there are triggers defined on the object you are calling `saveResource()` on then they will also be invoked



TriggerFramework – Read Trigger Considerations

- Read Triggers were introduced to provide some customization of data that was viewable within certain product screens in OpenPages
- Example in Filtered List View you can modify the Filter and add additional conditions before the data is returned.
- This was tied directly to the SDK findResources API calls. The args that the Trigger and Action worked on were HierFindResourceOptions objects.
- This type of trigger event is mapped to our SearchEvent type. API cannot directly interact with or convert HierFindResourceOptions to something that is easily workable using only New API.
- Note: The use cases for Read Triggers should strongly be looked at for opportunity to implement using new Security Rules feature in v7.



Example External Form

- Allow users to enter in financial losses in an anonymous way without logging in to OpenPages
- They need not be created as users in OpenPages security
- Users will navigate to a web page on company intranet or existing web portal infrastructure and fill out the form with details of the Loss.
- Upon submission the Loss may be created in OpenPages using new REST API.
- Possible Features
 - Validation of input data based on Metadata from OpenPages
 - Submit the created Loss to a workflow
 - Autonaming, default field values, associate to parents, attachments



REST API



REST Definition

- **Representational state transfer (REST)** is a style of software architecture for distributed hypermedia systems such as the World Wide Web. REST-style architectures consist of clients on one side and a server on the other.
- Clients initiate requests to the server; the server processes the request and returns an appropriate response. Requests and responses are built around the transfer of representations of resources. A resource can be essentially any coherent and meaningful concept that can be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.



REST Introduction

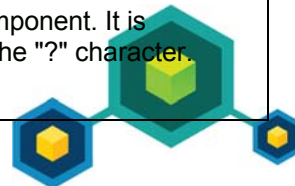
- HTTP/HTTPS protocol. Uses the protocols support for following Request methods to express actions
 - GET – Retrieve a resource
 - POST – Create a resource
 - PUT – Updates a resource
 - DELETE – Delete a resource
- All Server Resources represented by URIs
 - <https://server.com/path/to/my/resource>
- Examples
 - “GET <http://server.com/path/to/my/resource/abc>” will request the server to send a response containing ‘abc’
 - “PUT <http://server.com/path/to/my/resource>” with data in my request representing the new Resource will request the server to create it.



URI Syntax

- Most URIs adhere to the following general format:
- `<scheme>://<host>:<port>/<path>;<params>?<query>`

scheme	Which protocol to use when accessing a server to get a resource (http, https, ftp, etc.).
host	The hostname or dotted IP address of the server hosting the resource.
port	The port number on which the server hosting the resource is listening. Many schemes have default port numbers (the default port number for HTTP is 80).
path	The local name for the resource on the server, separated from the previous URL components by a slash (/). The syntax of the path component is server- and scheme-specific. The text portion between consecutive / are called path segments.
params	Used by some schemes to specify input parameters. Params are name/value pairs. A URL can contain multiple params fields, separated from themselves and the rest of the path by semicolons (;).
query	Used by some schemes to pass parameters to active applications (such as databases, bulletin boards, search engines, and other Internet gateways). There is no common format for the contents of the query component. It is separated from the rest of the URL by the "?" character.

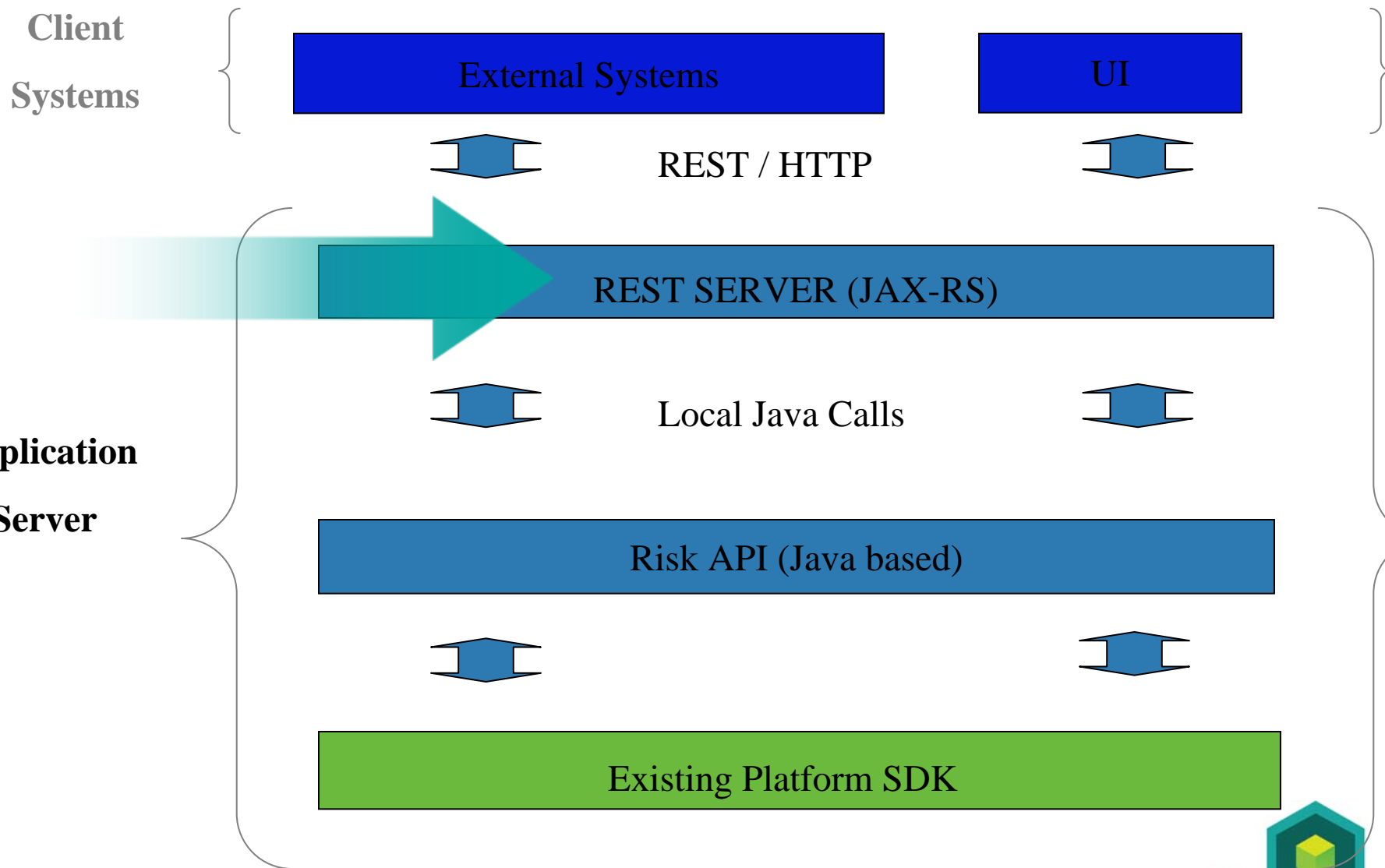


REST Background

- Firefox add-on RESTClient <https://addons.mozilla.org/en-us/firefox/addon/restclient/>
- JSON standard background: <http://www.json.org/>



Technology Stack – In the Labs



RESTful Webservices – GRC API

- Root URL /grc/api
- Expose core services to interact with data in OpenPages
 - /contents – data (includes audit trail)
 - /folders – folder data
 - /types – metadata
 - /query – query service
 - /workflow – workflow jobs and tasks
 - /configuration – OpenPages configuration data
 - /security – users, groups and role based security
- Data format in the form of JSON
- Implemented using JAX-RS. Java Standard for REST(JSR-311)



REST Example

- Get Object Type and Field Definitions for Object Type 'LossEvent'
 - Note: requires authentication to access server resources

Method	GET	URL	<input type="text" value="http://srdev7.rtp.raleigh.ibm.com:10108/grc/api/types/LossEvent"/>	☆ ▼	<input type="button" value="SEND"/>
--------	-----	-----	--	-----	-------------------------------------

- Response code 200 (OK). Response body server is JSON for this MetaData request

[.] Response

Response Headers Response Body (Raw) Response Body (Highlight) Response Body (Preview)

```
1. {
2.   "name": "LossEvent",
3.   "localizedLabel": "Loss Event",
4.   "description": "Unified Object Type",
5.   "jspPath": "/propertyForm/renderProperties.jsp",
6.   "id": "24",
7.   "rootFolderPath": "/_op_sox/Project/Default/ICDocumentation/Loss Events",
8.   "rootFolderId": "4268",
9.   "fieldDefinitions":
10.  {
11.    "fieldDefinition":
12.    [
13.      {
14.        "id": "28",
15.        "name": "Resource ID",
16.        "localizedLabel": "Resource ID",
17.        "dataType": "ID_TYPE",
18.        "required": true,
19.        "description": "Resource ID",
20.        "computed": false,
```



Reference Documentation

- GRC_REST_API.pdf
 - Reference guide, documents all supported paths, examples of JSON formats

Paths and parameters

The OpenPages GRC REST API supports the following paths and parameters.

API Path Segment	HTTP Method			
	GET	PUT	POST	DELETE
/grc/api/types	X			
/grc/api/types/{typeId}	X			
/grc/api/types/{typeId}/associations	X			
/grc/api/types/{typeId}/associations/parents	X			
/grc/api/types/{typeId}/associations/children	X			
/grc/api/types/{typeId}/associations/{associationId}	X			
/grc/api/contents			X	
/grc/api/contents/{contentId}	X	X		X
/grc/api/contents/{contentId}/document/{fileName}	X	X		
/grc/api/contents/{contentId}/template	X			

- Samples

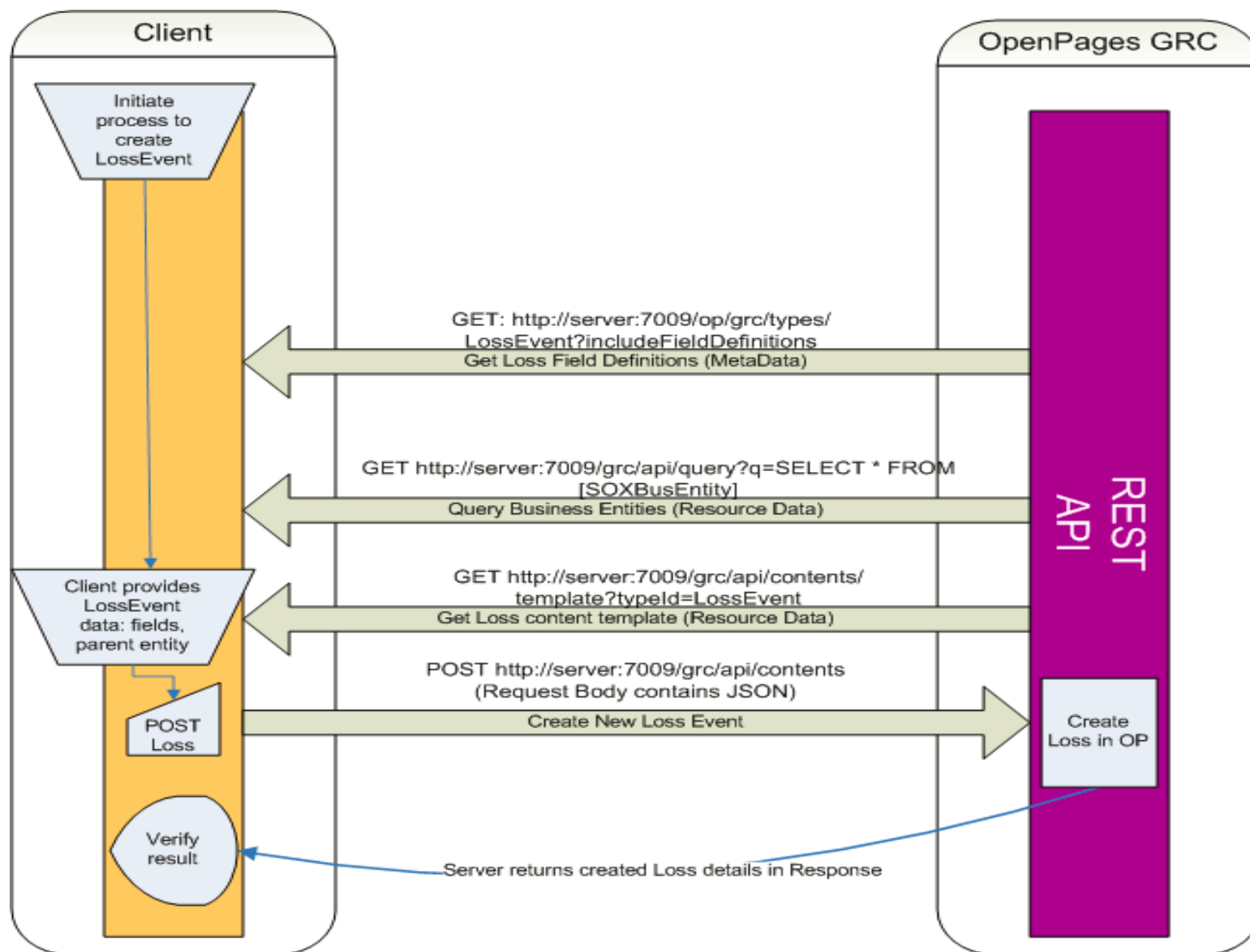


Basic authentication

- Default authentication done via Basic authentication through Application Server (Websphere, Weblogic)
- Basic authorization schema is defined by RFC 2617, and is implemented by the client sending an HTTP request header. Basic authorization contains the text “Basic” followed by the username and password separated by colon ‘:’ and encoded in Base64.
 - Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
- REST API adheres to stateless principles of REST. The client sends credentials with each request.
- Browsers will automatically prompt for credentials. Providing basic authorization credentials is usually handled by most client libraries or utilities.
- Best Practice: customers should configure SSL and use HTTPS for secure transmission
- **Note: Please pay attention to Post Installation Tasks in 7.0 Installation Guides. Required steps for REST!**



Client Server interactions



Implementing client side code

- Choose your own adventure
- REST APIs can be consumed by clients written in any language that supports HTTP requests. Java, .NET, Javascript/Dojo, VBA, etc...
- The choice may depend on the skill sets available or the technology stack of the client system, which may be another system or product that could be .NET, Java based.
- Example: **Java**
 - For REST client, the Apache Wink library is an open source library used in OpenPages for Rest Server, it also has REST Client libraries
 - For parsing JSON, there are multiple libraries, IBM Juno is used in OpenPages, Apache Wink also has JSON parser libraries
 - See Samples: “AnonLossEventFormREST”



Example Client code - Java

```
this.hostURLBase = "http://" + host + ":" + port;
ClientConfig config = new ClientConfig();
//for Basic authentication
BasicAuthSecurityHandler basicAuthHandler = new BasicAuthSecurityHandler();
basicAuthHandler.setUserName(user);
basicAuthHandler.setPassword(password);
config.handlers(basicAuthHandler);
// create the Wink rest client instance
restClient = new RestClient(config);
//get the full type since we need field defs
Resource typeResource =
restClient.resource(hostURLBase+TYPES_URL+"/"+objectType+"?includeFieldDefinitions=true");

ClientResponse typeResponse = typeResource.accept(MediaType.APPLICATION_JSON).get();

String body = typeResponse.getEntity(String.class);
//parse json body
TypeDefinitionType lossEventTypeFull = parser.parse(body, TypeDefinitionType.class);
System.out.println("Found " + lossEventTypeFull.getName() + " with Id: " + lossEventTypeFull.getId());
```



Demo External Form

- Allow users to enter in financial losses in an anonymous way without logging in to OpenPages
- They need not be created as users in OpenPages security
- Users will navigate to a web page on company intranet or existing web portal infrastructure and fill out the form with details of the Loss.
- Upon submission the Loss may be created in OpenPages using new REST API.
- Possible Features
 - Validation of input data based on Metadata from OpenPages
 - Submit the created Loss to a workflow
 - Autonaming, default field values, associate to parents, attachments



Demo Excel Integration

- User Provisioning through Excel macro
- Written in MS Excel's VBA code. It uses REST APIs to get user details, create users and assign roles
- This is an example of a different use case for APIs outside of data integration, as well as illustrating non-Java clients.

IBM® OpenPages® GRC Platform			
User Provisioning			
User	Name	MyTestUser13	Get User
	Id	1216	Create User
	First	A	
	Last	B	
	Email	test@ibm.com	
Roles	Id	Name	Entity Path - Security context
	1	Manager	/Test 01



Use Case: External System Integration

- System to System Integration for Openpages data
- Data and message transfer to multiple systems
 - Importing data from external data sources
 - Publishing OpenPages objects, data to other systems
- Examples
 - E.g. Tivoli Directory Integrator, IT Security systems for ITG – e.g. Q1 QRadar, Modeling tools, Business Hierarchy changes



Migrating from OpenAccess

- OpenAccess is a legacy webservices API provided by OpenPages.
- There will be a deprecation notice for OpenAccess in v7.0
- OpenAccess was WSDL/SOAP-XML based webservices. It was only supported on WebLogic application servers.
- For existing customer implementations using OpenAccess webservices
 - Any client side code will need to be ported to REST API
 - For each OpenAccess service there is an analogous set of REST API paths



Data - /contents

- **/grc/api/contents/**: A POST to this URL containing the entry representing the object to be created will create the GRC Object. The entry can be constructed from the template returned from `/grc/api/contents/template` for the type of object being created.
- **Example:** Creating Action Item POST this body to `/grc/api/contents` with header
- Content-Type: `application/json`
- Body:

```
{  
  "fields": {},  
  "typeDefinitionId": "45",  
  "primaryParentId": "3954",  
  "name": "testActionItem01",  
  "description": "Action Item description"  
}
```



Data - /contents – working with a Resource

- **/grc/api/contents/{id}**
- The GRC Objects can be referenced by their Id, or path. The path may be either relative or full path. The path must be URL encoded as it may contain '/' reserved character.
- Example: For referencing an instance of an Issue using different options for the "id"
 - Resource Id: 3954
 - Relative Path: Issue/Test 01/Issue 1
 - Full Path: /_op_sox/Project/Default/Issue/Test 01/Issue 1.txt
- A GET to this URL will have a response which contains json data in the body:

```
{  
  "name": "Issue 1",  
  "id": "3954",  
  "path": "/_op_sox/Project/Default/Issue/Test 01/Issue 1.txt",  
  "description": "This is an example of free text.",  
  "parentFolderId": "1969",  
  "fields":  
  {  
    ...
```



Data - /contents

- **/grc/api/contents/{id}**: A GET operation on such a path would return a representation of the OP resource. Given the restrictions on the characters part of a URI, the path must be URL encoded.
 - An application data resource with URI template /grc/api/contents/{id} can be modified by performing a PUT operation with the updated value for the resource.
 - A DELETE operation would delete the Resource.

- **/grc/api/contents/{id}/document/{document name}**: A GET operation will retrieve the file attachment contents for this document
 - A PUT operation to this path will update the document.



Data - /contents

- **/grc/api/contents/{ID}/associations:** A GET operation Retrieve all associations for the given resources. A POST will create new associations. .
- **/grc/api/contents/{ID}/associations/parents:** A GET operation Retrieve only parent associations for the given resources. A POST with a Feed for new associations will create new parent associations.
- **/grc/api/contents/{ID}/associations/parents:** A GET operation Retrieve only parent associations for the given resources. A POST with a Feed for new associations will create new parent associations.
- **/grc/api/contents/{id}/associations?children={id},{id}...;parents={id},{id}...:** A DELETE operation will Remove parent and/or child associations for the given resource. The list of parents or children Ids can be any length.



Data - /contents

- **/grc/api/contents/template?typeld={ID}**: GET operation on /grc/api/contents/template would return a "templated" Entry for this type of object specified by the typeld query parameter. It would be used as a template to create the new object. Once initialized, the new resource would be POSTed to /grc/api/contents.
- **/grc/api/contents/**: POST to this URL containing the Entry representing the object to be created as taken from the template returned from /grc/api/contents/template for the type of object being created.
- **/grc/api/contents/{id}/associations?children={id},{id}...;parents={id},{id}...**: A DELETE operation will Remove parent and/or child associations for the given resource. The list of parents or children Ids can be any length.



Data - /folders

- **/grc/api/folders:** A GET Operation returns the root folder “/”. The folder information is in the op:folder extension. To view the containees perform a GET operation on the “down” link e.g. /grc/api/folders/1/containees
- **/grc/api/folders/{id}:** A GET operation returns an Entry for the specified Folder by it's Id
- **/grc/api/folders/{id}/containees:** A GET operation returns a Feed for all children of the folder.
 - Note the entries in some cases will be of Folders, but also other Resources or GRC Object entries.



Query Service - /query

- **/grc/api/query?{query specification}**: a GET operation using this URI is meant to be used to perform a query using the query service syntax call on the OP model and page through the result set.
 - Result is a Feed containing an <op:row> Entry for every row returned by the query.
 - For results greater than default page size, use href URL rel="next" link to go to next page.



Eclipse Project templates

Reference for developing with API on Eclipse



Developing with Eclipse

- Development using Eclipse IDE
 - Recommend Eclipse Indigo (3.7.x) or later
- JDK 1.6
- Sample projects provide instruction
- Depending on project type, have different requirements
 - Jars will be different for Triggers vs Workflow Action development
- Can leverage existing 'openpages-ext' projects



Basic existing 'openpages-ext' project

- Take an existing 6.x project for creating openpages-ext.jar
- Add additional jars to project's lib/ directory:
 - com.ibm.openpages.api.jar
 - com.ibm.openpages.api.encryption.jar
- Using Eclipse Build Path editor for the project add the two jars to build path.
- The project should now compile Java source files using the API
- The project's build.xml should already pick up the new jars from the lib directory
- Note: For Weblogic you must modify the **build.env** properties file to include com.ibm.openpages.api.jar on classpath

```
# OpenPages openpages-ext.jar build environment...  
# Enter JARs that must be added to the classpath of openpages-ext.jar.  
# These JARs will be loaded by both OP and IBPM servers  
classpath=../../applications/op-apps/com.ibm.openpages.api.jar
```



openpages-ext.jar manifest

- As stated on previous slide. You must modify the **build.env** properties file to include `com.ibm.openpages.api.jar` on classpath
 - This creates an entry in `openpages-ext.jar`'s MANIFEST.MF file.
- Having this jar on the `openpages-ext.jar` is required for API to function.
- If you are using other tools or methods to create the `openpages-ext.jar`. You must also include the this entry in the META-INF/MANIFEST.MF:
 - Class-Path: `com.ibm.openpages.api.jar`

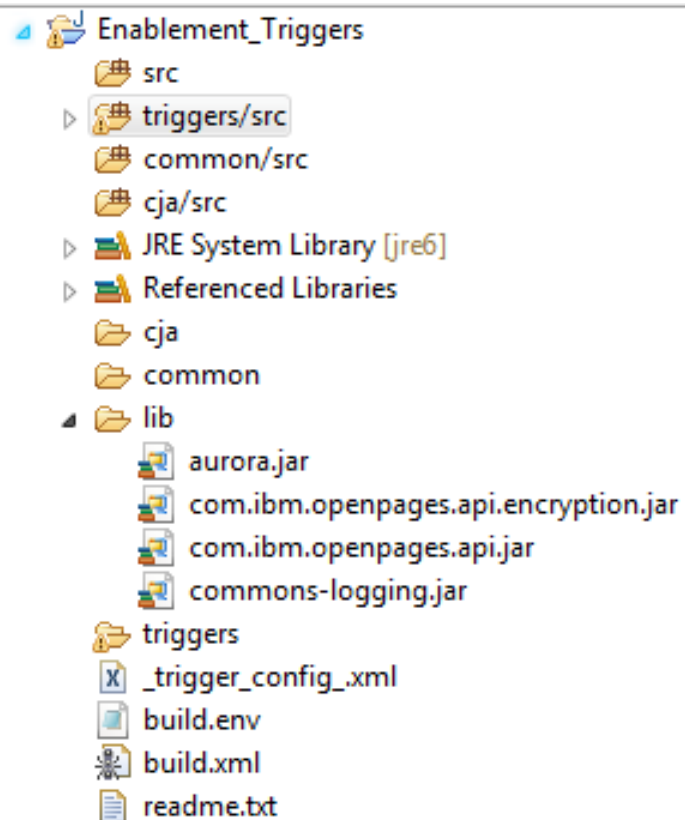


New Trigger Project

- If you do not have existing OPSDK in your Java source code, you may create a lighter weight project that only has minimal required API jars
- How to create as Project In Eclipse
 - From File menu select New > Java Project
 - Make sure to use a Java 1.6 JRE
 - Copy the 'samples/Triggers' folder contents to your project
 - In Java Build Path > Source, use Add Folder and add common/src, cja/src and trigger/src
- Add the following jars to the project's lib folder
 - com.ibm.openpages.api.jar and com.ibm.openpages.api.encryption.jar
 - from GRC_API/lib
 - aurora.jar
 - from <OpenPages_Home>/applications/op-apps
 - commons-logging.jar
 - from <OpenPages_Home>/applications/op-apps
- See samples/Triggers and contained readme.txt



Sample Trigger Project














Workflow Action Projects

- In Addition to the jars mentioned in New Trigger project, when creating a project for developing WorkflowActions you must add the following product jars to the lib dir and project classpath:
 - iFlow.jar
 - from <OpenPages_Home>/applications/op-apps
 - op-integration.jar
 - from IBPM server install location e.g.
C:\Fujitsu\InterstageBPM\BPMDomain\servers\<hostname>-InterstageBPMCS1\tmp_WL_user\op-ibpm-integration\
 - See samples/WorkflowActions and contained readme.txt



Sample Workflow Project

- ▲  Enablement_WorkflowActions
 - ▷  src
 - ▷  JRE System Library [jre6]
 - ▷  Referenced Libraries
 - ▲  lib
 -  aurora.jar
 -  com.ibm.openpages.api.encryption.jar
 -  com.ibm.openpages.api.jar
 -  commons-logging.jar
 -  iFlow.jar
 -  op-integration.jar



JSP Development

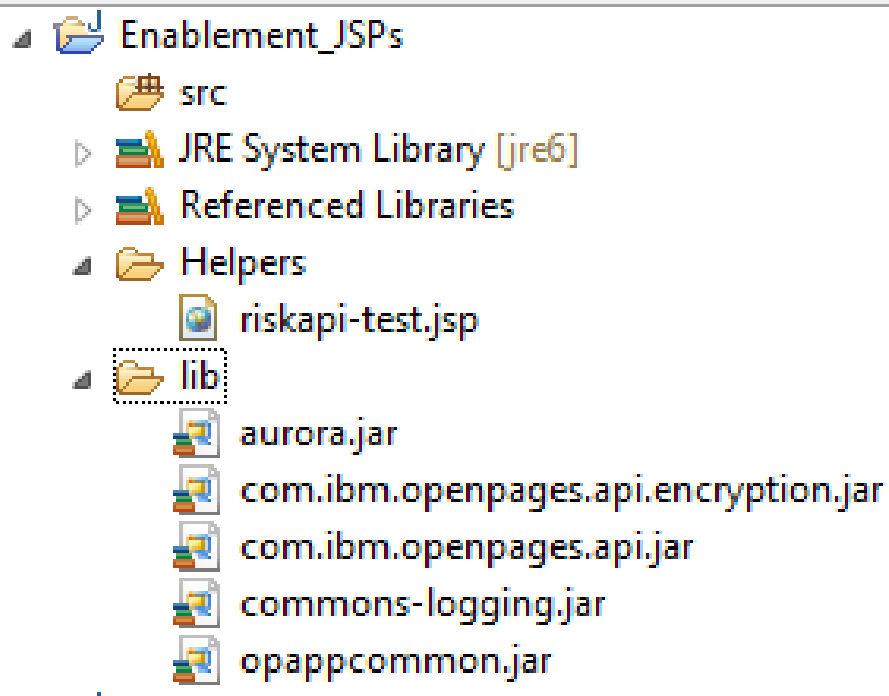
- Basically same as the Trigger development project or openpages-ext.jar.
- Either “publishweb” or “sosa” JSPs
- Create new Java project
- Add directory for JSPs in your project e.g. Helpers
- In order use the current OpenpagesSession of a user in the JSP (ie helper) then you will need to add one jar:
 - opappcommon.jar
 - from <OpenPages_Home>/applications/op-apps
 - This is so you can use HttpContext, an op apps class:

```
//init steps
if (!HttpContext.IsValidSession(request))
{
    HttpContext.createValidSession(request);
}
OpenpagesSession opSession = HttpContext.getOpenpagesSession(request);

//put the OpenpagesSession into the API's Context
context = new Context();
context.put(Context.SERVICE_SESSION,opSession);
```



Sample JSP Project



Using Remote Debug Feature

- This is NOT API specific.
- WebLogic Server set up.
 - In Windows Registry on server, modify the server JVM arguments:
 - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\OpenPagesServer1\Parameters > CmdLine
 - Add: -Xdebug -Xrunjdwp:transport=dt_socket,address=3099,server=y,suspend=n
 - Need to restart services.
- In Eclipse
 - Open menu: Run > Debug Configurations
 - For Remote Java Application, add New configuration for this server
 - On Connect tab, In Connection properties give the server hostname and the socket.address port # provided on server arguments above.
 - On Source tab, click Add > Java Project > select any Java Projects you may wish to debug.
- To use, put a break point in your code. Connect to the server, using Debug Configurations > your configuration > Debug
- Perform actions on the server to reach your code.
- You can now step through your custom code.



Sample Debug Remote Java Configuration

- Read More <http://www.ibm.com/developerworks/library/os-eclipse-javadebug/>

Name: blaskey-OpenPagesServer1

Connect Source Common

Project: aurora Browse...

Connection Type: Standard (Socket Attach)

Connection Properties:

Host: blaskey

Port: 3099

☐ Allow termination of remote VM



