Aras Innovator 12

Programmer's Guide

Document #: 12.0.02019054101

Last Modified: *05/12/2020*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810

Phone: 978-806-9400
Fax: 978-794-9826

E-mail: Support@aras.com

Website: https://www.aras.com

Table of Contents

Send Us Your Comments

Aras Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for future revisions.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where and what level of detail?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, indicate the document title, and the chapter, section, and page number (if available).

You can send comments to us in the following ways:

Email:
Support@aras.com
Subject: Aras Innovator Documentation

Or,

Postal service:
Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Attention: Aras Innovator Documentation

Or,

FAX:
978-794-9826
Attn: Aras Innovator Documentation

If you would like a reply, provide your name, email address, address, and telephone number.

If you have usage issues with the software, visit https://www.aras.com/support/

Document Conventions

The following table highlights the document conventions used in the document:

Table 1: Document Conventions

Convention	Description
Bold	This shows the names of menu items, dialog boxes, dialog box elements, and commands. Example: Click **OK**.
`Code`	Code examples appear in `courier` font. It may represent text you type or data you read.
`Yellow highlight`	Code highlighted in yellow draws attention to the code that is being indicated in the content.
`Yellow highlight with red text`	Red text highlighted in yellow indicates the code parameter that needs to be changed or replaced.
Italics	Reference to other documents.
Note:	Notes contain additional useful information.
Warning	Warnings contain important information. Pay special attention to information highlighted this way.
Successive menu choices	Successive menu choices may appear with a greater than sign (-->) between the items that you will select consecutively. Example: Navigate to **File** --> **Save** --> **OK**.

1 Introduction

The purpose of this document is to provide a Guide to Programming Aras Innovator. It covers key aspects of programming Aras Innovator for implementing your own business logic within the Aras Innovator Enterprise Application Framework.

This document is intended to be used as a Desktop Reference and User Guide covering the following topics:

- The AML (Aras Markup Language), which is the language that drives the Aras Innovator server.
- The IOM (Innovator Object Model), which is the Object API for the AML.
- How Methods work and the Methodology for implementing your own business logic.
- A Cookbook of recipes for performing common tasks.
- The .NET controls the API that the Aras Innovator Client is built on.

1.1 The Item

Everything in Aras Innovator is an Item, which is an instance of an ItemType, which itself is an Item; illustrating that Aras Innovator is a self-describing system. Don't get hung up on the self-describing nature of Aras Innovator and focus on the simplicity that everything eventually is an Item.

Items may have relationships to other Items illustrating that Items have structure. Relationships are defined by RelationshipType, which is an Item that has three properties to define the RelationshipType rule for the:

- source (parent) Item.
- related (child) Item.
- relationship Item.

When you create the RelationshipType you also create an 'is_relationship' ItemType which has the same name as the RelationshipType. Its id is the value of the relationship_id Property on the RelationshipType. This can get a bit confusing but simply put there is a RelationshipType/ItemType pairing to define the RelationshipType rule and an ItemType to store the relationship Items.

Relationship Items have a related_id Property of type Item, which is the related (child) Item for the relationship. The related_id Property is a link that points to an Item. The relationship Item also has a source_id Property of type Item and is the source (parent) Item for the relationship.

So, in Aras Innovator everything is an Item, and Items may have relationships, which are Items that have source and related Item Properties forming an Item configuration.

For example, an ItemType Item has Property relationships and this Item configuration maps directly to the relational database for persistent storage of the Item instances. Every ItemType has a matching relational TABLE where the Property names are the COLUMN names.

1.2 The Aras Markup Language (AML)

The Aras Markup Language (AML) is an XML dialect that follows the simple `/Item/Relationships/Item/Relationships` repeating pattern to describe Item configurations. Clients submit AML documents to the Aras Innovator server and receive an AML document back.

An AML document contains data (Items), structure (Relationships, which are hierarchical Items), and logic (an action to perform some business logic on the Item). Each Item in the AML document has an action attribute, which is the name of an Aras Innovator Method that performs business logic on the Item. The Aras Innovator server interprets AML documents similar to scripting languages. AML documents are often referred to as AML scripts.

This is an example of a BOM in AML language:

```
<Item type="Part" action="add">
  <item_number>999-888</item_number>
  <description>Some Assy</description>
  <Relationships>
    <Item type="Part BOM" action="add">
      <quantity>10</quantity>
      <related_id>
        <Item type="Part" action="add">
          <item_number>123-456</item_number>
          <description>1/4w 10% 10K Resistor</description>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

1.3 Methods and the IOM

The IOM (Innovator Object Model or Item Object Model) is an Object Model on top of the AML. It provides the ability to build and submit AML documents to the Aras Innovator Server using a simple Object API.

There is the 'Method' ItemType, which is used to implement user defined business logic. Methods are written in JavaScript, C#, or VB.Net and use the IOM API to implement the business logic.

The following is a Method in JavaScript using the IOM that is the same as the AML BOM example in the previous section:

```
var innovator = new Innovator();
var partItem = innovator.newItem("Part","add");
partItem.setProperty("item_number", "999-888");
partItem.setProperty("description", "Some Assy");

var bomItem = innovator.newItem("Part BOM","add");
bomItem.setProperty("quantity", "10");

var relatedItem = new Item("Part","add");
relatedItem.setProperty("item_number", "123-456");
relatedItem.setProperty("description", "1/4w 10% 10K Resistor");

bomItem.setRelatedItem(relatedItem);
partItem.addRelationship(bomItem) ;

var resultItem = partItem.apply();
```

2 AML

The AML is the XML dialect and language that drives the Aras Innovator server. Clients submit AML documents to the Aras Innovator server via HTTP. The server parses the AML applying the business logic defined as the action attribute for the Items in the AML document, and an AML document is returned. The AML dialect is very simple. The following sections describe the tags that define the AML language:

2.1 <Item> Tag

The `<Item>` tag defines an Item instance. XML is case-sensitive (notice the capitalized Item.) There are three principle attributes for the Item tag to define the Item instance:

- `id` – the unique ID for the Item.
- `type` – the ItemType name for the Item.
- `action` – the name of the Method that is applied to the Item.

There are other attributes but these are the most significant. They define the Item and the action to apply on it. The following is an AML query example requesting a Part Item by ID:

```
<Item type="Part" id="ACBDEF0123456789…" action="get"/>
```

Refer to section 4.2 Built in Action Methods for the list of Aras Innovator pre-built actions.

2.2 <Relationships> Tag

Items can have relationships to other Items. The `<Relationships>` tag is a container tag that holds the set of relationship Items. Again notice the capitalized Relationships. There are no attributes for the tag because it is a container. With relationships it is possible to describe an Item configuration to any level deep as necessary.

The following is an AML query example requesting a Part Item and its BOM relationships:

```
<Item type="Part" id="ACBDEF0123456789…" action="get">
  <Relationships>
    <Item type="BOM" action="get"/>
  </Relationships>
</Item>
```

2.3 <property> Tags

Properties for the Item are the nested tags directly below the `<Item>` tag. The Property name is the tag name. For example, a 'Part' ItemType may have the properties: item_number, description, and cost, which are also the tag names in the AML. Property names are lowercase so the property tag names are also lowercase.

The following AML illustrates a simple Item configuration for describing a Part to Part BOM relationship:

```
<Item type="Part" action="add">
  <item_number>999-888</item_number>
  <description>Some Assy</description>
  <Relationships>
    <Item type="Part BOM" action="add">
      <quantity>10</quantity>
      <related_id>
        <Item type="Part" action="add">
          <item_number>123-456</item_number>
          <description>1/4w 10% 10K Resistor</description>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

Property values always use locale-neutral formats. Decimal and float values use the period symbol (.) as the decimal separator and no digit separator (i.e. commas separating thousands). The dash symbol (-) is used to denote a negative value. Date/Time values should be in 'YYYY-MM-DD[Thh:mm:ss]' format and in the local (or corporate) time zone. Language-specific values use the xml:lang attribute to specify the language code. For example:

```
<Item type="Part">
  <item_number>292-102</item_number>
  <i18n:name xml:lang="de"
xmlns:i18n="http://www.aras.com/I18N">Rad</i18n:name>
  <i18n:name xml:lang="en"
xmlns:i18n="http://www.aras.com/I18N">Wheel</i18n:name>
  <cost>232.13</cost>
  <created_on>2008-08-24T08:12:02</created_on>
</Item>
```

2.4 Attributes

Properties are used to define the data for an Item. Attributes are meta-data for the Item or Property. They are used to control the server logic and Methods. Think of attributes like command line switches, or as arguments to a function.

In addition to the type, id, and action attributes mentioned above there are several additional attributes used to control the server. The following is the attribute reference for the <Item> tag:

2.4.1 Item Attributes

Attribute	Type	Usage
type	String	The ItemType name for which the Item is an instance.
id	String	The unique ID value for the Item instance.
where	String	Used instead of the id attribute to specify the WHERE clause for the search criteria. Include the table name with the column name using

Attribute	Type	Usage
		the dot notation: where='user.first_name like 'Tom%''
action	String	The name of the Method (or Built in Action Method) to apply to the Item.
doGetItem	Boolean	If 0 then do not perform a final get action on the Item after the server performed that action as defined by the action attribute. Default is 1.
Used with action="get"		
select	String	A comma delimited list of property names (column names) to return which is the SELECT clause in the SQL statement.
orderBy	String	A comma delimited list of property names (column names) to order the results and is the ORDER BY clause in the SQL statement.
page	Integer	The page number for the results set.
pagesize	Integer	The page size for the results set.
maxRecords	Integer	This defines the absolute maximum Items to be searched in the database.
levels	Integer	The Item configuration depth to be returned. This should be used with caution because of the performance hit due to its lack of granularity in the data fetched. Use the nested Relationships style of defining your queries to do the same thing but with far greater performance.
serverEvents	Boolean	If 0 then disable the server events improving performance. Default is 1.
isCriteria	Boolean	If 0 then include the nested structure for the Item configuration in the response but don't use it as search criteria. Default is 1, which uses the nested structure in the request as search criteria.
related_expand	Boolean	If 0 then do not expand the related_id Property for the relationship Items to include the related Item. Another word returns its ID.
language	String	A comma-delimited list of language codes, or "*" to return all languages. Multilingual property values is returned (if present) for all specified languages.
Used with action="update" | "edit"		
version	Boolean	If 0 then don't version an Item on update. Default is 1, which is version the Item (if it's a versionable Item) on update.
serverEvents	Boolean	If 0 then disable the server events improving performance. Default is 1. Only 'Update' events are disabled, 'Lock' events can be executed if using 'Edit'.

2.4.2 Property Attributes

<table>
<tr><th>Attribute</th><th>Usage</th></tr>
<tr><td>type</td><td>The ItemType name for the item instance.</td></tr>
<tr><td>keyed_name</td><td>This is the keyed_name Property for the Item referenced by the Item type Property.</td></tr>
<tr><td>condition</td><td>This is the condition value for the AML query. The condition value is any valid SELECT condition supported by the database. The following is the list of possible condition values:
<table>
<tr><th>Condition</th><th>Comments</th></tr>
<tr><td>eq
ne
ge
le
gt
lt</td><td>The SQL conditional symbols:
=, <>, <=, >=, >, and <
are expressed as two letter mnemonic words:
eq, ne, le, ge, gt, and lt
Example:
<name condition="gt">100</name></td></tr>
<tr><td>like
not like</td><td>The value for the Property tag would include wild card symbols, which are % for any characters and _ for a single character.
Example:<name condition="like">Tom%</name></td></tr>
<tr><td>between
not between</td><td>This is a range condition. You would include the AND keyword in the Property tag value.
Example:
<cost condition="between">10.00 and 50.00</cost></td></tr>
<tr><td>in
not in</td><td>This is a set based condition where the value for the Property tag is a comma delimited list of values for the set.
Example:
<name condition="in">'Tom', 'Peter', 'Joe'</name></td></tr>
<tr><td>is
is null
is not null</td><td>Possible values for the Property tag could be null or not null.
Example:
<cost condition="is null"/>
<cost condition="is not null"/></td></tr>
</table>
</td></tr>
</table>

Attribute	Usage
Xml:lang	Language code for multilingual properties. You must use this in conjunction with the internationalization namespace on the property tag. For example: `<i18n:name xml:lang="de" xmlns:i18n="http://www.aras.com/I18N">Rad</i18n:name>`

3 IOM Reference

IOM Reference provides a general description of the IOM (Innovator Object Model or Item Object Model) API. A more detailed API reference may be obtained from one of the following:

- **On-line**: Go to https://www.aras.com/support/documentation/

  Under the section **Other Documents**, click On-**Line API Guide.html**.
- **From Aras Innovator Client UI**: Login as Innovator Administrator. Under the Help menu, select API Reference.
- **In Aras Innovator CD Image**: Go to the documentation folder and open `Aras Innovator - API Guide.html`.

The IOM is an Object Model for the AML, but it is not purely Object Oriented. Using Object Oriented terms, an ItemType is like a 'Class' and the Item is like an 'Object'. Although the Item is an Object with methods, there is only one Item Class for all ItemTypes. In a pure Object Oriented representation, there would be a Class for each ItemType because each ItemType has its own set of Properties to describe the different Items.

An Item Object is intended to be abstract and pliable. Depending on its internal structure, the Item Object usually represents one of five following supported types of IOM Items:

- **Single** Aras Innovator Item of an arbitrary ItemType.
- **Set** of Aras Innovator Items, as in the case for results of the action 'get' when more than one Item returns.
- **Error**, as in the case when an action request results in an error from the Aras Innovator Server.
- **Result** to represent an arbitrary text wrapped by <Result> XML tags.
- **Logical** to represent a set of properties wrapped in a logical statement by one of logical XML tags: <or>, <and>, or <not>; usually used to specify the search criteria for the action 'get'.

The IOM is intended to be a generic and compact API for modeling the Item structure of the AML as abstract Objects. The majority of the methods for the IOM deal with memory management of the AML document for the Item Object. The AML is a script sent as a message to the Aras Innovator Server. The IOM is an Object API to build the AML messages, submit them to the Aras Innovator Server and parse an AML document that is returned by the Server.

There are two major types of methods in the Item Class:

- methods that only work with item's AML in memory
- methods that communicate with the Server, i.e. send request(s) to and get response(s) from the server.

All get\set type of methods as well as isXXX(...) (e.g. isError(), isCollection(), etc.) and add\remove methods (e.g. addRelationship(...), removeProperty(...), etc.) belong to the former group. Methods like fetchXXX(...) (e.g. fetchLockStatus()), apply(...), email(...), promote(...), lock\unlockItem(...), etc. belong to the latter group. In case a method sends a request to the server it must be explicitly mentioned in the API reference method comments.

Note:	The term 'method' has two meanings in this guide; the typographical convention used throughout this Guide is as follows: Lowercase 'method' refers to methods on an IOM Item Object. Capitalized 'Method' refers to Method Items stored in the Aras Innovator database. Sample code is shown in `Courier 10pt font.` Optional arguments are surrounded by [] characters.

3.1 IOMCredentials Class

The IOMCredentials Class defines the login credentials for connecting to the Aras Innovator Server. The Item Class has a credentials public property, which is an IOMCredentials Object.

Typically, you do not need to know about or deal with the credentials Object because the Methods run in a logged in session. You can, however, set the credentials for the Item Object using these methods, if you want to submit the apply requests to a different Aras Innovator server. The IOMCredentials Class methods are mostly getters and setters for URLs (Innovator and Vault servers), DB name, and login name and password.

3.2 Innovator Class

In Innovator Class, methods like `applyAML(…)`, `applyMethod(…)`, and `applySQL(…)`, being type Item, send the apply request to the Aras Innovator Server. In response, the Server creates XML that apply_ methods used to construct an Item Object to return; methods like `getItemById(…)` and `getItemByKeyedName(…)` search the database the logged in session is connected to, and methods like `newItem(…)`, `newError(…)` and `newResult(…)` construct a new instance of the Item Object.

Other members of this class perform miscellaneous non-Item related operations. Use them if you need to get a new GUID (methods `getNewID()`), generate a next sequence value (method `getNextSequence(…)`), or calculate the MD5 hash value for a given string (`method ScalcMD5(…)` .)

3.3 Item Class

An Item Object can represent an Item, a set of Items, or an Error.

Item public constructor has one required argument itemtype-name and one optional argument action. The new Item is only populated with the Properties and default values from the ItemType when the optional action argument is 'add'.

The Item Class public field `dom` represents a DOM Object that holds the data for the Item in the AML format.

3.3.1 Base Methods

The Item Class base method `apply(…)` submits the AML apply request to the Aras Innovator Server using the context Item DOM as the AML source and returns a new Item built on the XML returned by the Server. In contrast, the base method `loadAML(…)` does not return a new Item but rather rebuilds `this.dom` using AML taken as an argument.

You can call `clone(…)` method to get a new identical instance of the context Item and you can call `setNewID()` method to replace the context Item id by newly generated GUID.

3.3.2 Boolean Methods

Use Call method `isCollection()` to find out whether or not the Item represents a set of Items, e. g. whether or not its dom property holds more than one Item node. Use Call `isError()` method to find out whether or not the Item represents an Error. See the IOM API on-line reference for more boolean members of the Item Class.

3.3.3 Attribute Methods

Methods such as `getAction()`, `getID()`, and `getType()` return a value of Item node action, id, and type attribute respectively, while method `getAttribute(…)` takes an attribute-name as an argument and, therefore, can be used to get any attribute by name. Thus, `getAction()` is a short cut to `getAttribute("action")`, `getID()` is a short cut to `getAttribute("id")`, and so on.

Each get method in this group has a corresponding set method: `setAction(…)`, `setID(…)`, `setType(…)`, and `setAttribute(…)`. Notice that method `setAttribute(…)` not only sets new value for an existing attribute but can also add a new attribute to the Item node and then sets its value.

3.3.4 Property Methods

Property methods comprise a set of accessors to Item properties and Item property's attributes: get_, set_ (which acts also as add_), and remove_. In addition, if a property has/needs a nested Item there are methods to get/insert these nested Items.

Accessors to Properties are `getProperty(…)`, `setProperty(…)`, and `removeProperty(…)`.These methods take then property name as an argument.

Accessors to Property's attribute are `getPropertyAttribute(…)`, `setPropertyAttribute(…)`, and `removePropertyAttribute(…)`. These methods take property and attribute names as arguments.

Method `setProperty(…)/setPropertyAttribute(…)` not only sets a new value for an existing Item property/ property's attributes but creates new property/ property's attributes and then sets its value if the property/ property's attributes with a given name does not yet exist.

The `setProperty` method requires property values to be in a locale-neutral format. Decimal and float values should use the period symbol (.) as the decimal separator and no digit separator (i.e. commas separating thousands). The dash symbol (-) should be used to denote a negative value. Date/Time values should be in 'YYYY-MM-DD[Thh:mm:ss]' format and in the local (or corporate) time zone. Language-specific values should be set using the language code as the third argument.

Accessors to Property's nested Item are: `getPropertyItem(…)`, `setPropertyItem(…)`, and `createPropertyItem(…)`. All these methods take property name as an argument, and method `setPropertyItem(…)` needs, in addition, an Item Object as the second argument to create a DOM for the nested Item.

3.3.5 Relationship Methods

Relationship methods are made up of a set of Item's relationship accessors: `getRelationships(…)`, `addRelationship(…)`, `createRelationship(…)`, and `removeRelationship(…)`. The difference between `createRelationship(…)` and `addRelationship(…)`, two methods that both are adding an Item node to the Relationship parent node, is subtle. Let's consider the following server-side C# method to illustrate how `createRelationship(…)` works:

```
Item myInnovator = this.getInnovator();
Item myItem = myInnovator.newItem("myType","myAction");
myItem.createRelationship("User","add");
```

This code results in the following myItem.dom XML:

```
<Item isNew="1" isTemp="1" type="myType" action="myAction">
  <Relationships>
    <Item isNew="1" isTemp="1" type="User" action="add" />
  </Relationships>
</Item>
```

Similar code that exercises addRelationship(…) methods is:

```
Item myInnovator = this.getInnovator();
Item myItem = myInnovator.newItem("myType","myAction");
Item relItem = myInnovator.newItem("User","add");
myItem.addRelationship(relItem)
```

and this code results in following myItem XML:

```
<Item isNew="1" isTemp="1" type="myType" action="myAction">
  <Relationships>
    <Item isNew="1" isTemp="1" type="User" action="add"
id="7EA3F18935CC493A900DAB63E839FDA2">
      <classification>/*</classification>
      <default_vault>67BBB9204FE84A8981ED8313049BA06C</default_vault>
    </Item>
  </Relationships>
</Item>
```

As you can see `addRelationship(…)` produces a more detailed Item node under the Relationship parent node.

There are three methods `getRelatedItem(…)`, `setRelatedItem(…)`, and `createRelatedItem(…)` that are valid for relationship Items only and are used as a short cut to the `Item.get/setPropertyItem()` methods.

3.3.6 Item Collection Methods

This group is comprised of methods to work with collections. Collection in this context is a set of Items as in the case for results from a query. Method `appendItem(…)` appends its Item Object argument to an existing collection or converts a single Item instance to a set of Items. This mechanism is illustrated in the following C# sample:

```
Innovator myInnovator  = this.getInnovator();
Item myItem  = myInnovator.newItem("myType","myAction");
```

At this point, myItem presents a single Item instance, myItem.dom XML looks like this:

```
<Item isNew="1" isTemp="1" type="myType" action="myAction" />
and myItem.isCollection() returns false.  But three extra line of code:
Item addedItem  = myInnovator.newItem("added","myAction");
// set ID to be able to remove addedItem later
addedItem.setID(myInnovator.getNewID());
myItem.appendItem(addedItem);
```

It converts myItem to a collection, e.g. `myItem.isCollection()` returns `true`, and `myItem.dom` XML becomes as follows:

```
    <AML>
      <Item isNew="1" isTemp="1" type="myType" action="myAction" />
      <Item isNew="1" isTemp="1" type="added" action="myAction"
id="B12F9384B1DE4C4F8158C36D18269BE9" />
    </AML>
```

If the Item object id is not NULL it can be removed from the collection:

```
myItem.removeItem(addedItem);
```

Use the `getItemCount()` method to determine the size of a collection, the `getItemByIndex(…)` method to get an instance of the Item Object based on its position inside of the collection, and method `getItemsByXPath(…)` to find an Item by its XPath.

3.3.7 Logical Methods

The methods `newOR()`, `newAND()`, and `newNOT()` inserts logical node with tag <or>, <and> and <not> respectively under the parent Item node and returns an Item Object that represents a newly inserted logical node. For example, the following code:

```
Innovator myInnovator  = this.getInnovator();
Item myItem  = myInnovator.newItem("myType","myAction");
Item logicalOR = myItem.newOR();
logicalOR.setProperty("foo", "bar");
```

produces the following `myItem.dom` XML:

```
<Item isNew="1" isTemp="1" type="myType" action="myAction">
  <or>
    <foo>bar</foo>
  </or>
</Item>
```

The method `removeLogical(…)` removes a logical node specified by method's argument.

3.3.8 Creating New Item Method

You can create a new item using method `newItem(…)`. See code samples in sections 3.3.6-3.3.8 for the method usage illustration.

3.3.9 Error Methods

Error methods are comprised of a set of accessors to Error specific properties such as "faultcode" (methods `get/setErrorCode(…)`,) "faultstring" (methods `get/setErrorString(…)` ,) "faultactor" (`get/setErrorSource(…)` ,) and "detail" (methods `get/setErrorDetail(…)` .)

Thus, the code sample below:

```
Innovator myInnovator  = this.getInnovator();
Item error = myInnovator.newError("default detail");
error.setErrorCode("any number");
error.setErrorString("Hello, World");
error.setErrorSource("myMethod");
error.setErrorDetail("new detail");
```

produces the following Error Item DOM:

```
<Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Body>
    <Fault>
      <faultcode>any number</faultcode>
      <faultstring>Hello, World</faultstring>
      <faultactor>myMethod</faultactor>
      <detail>new detail</detail>
    </Fault>
  </Body>
</Envelope>
```

3.3.10 Extended Item Class methods

This set of methods implements specific functionality on the Item, which extends the base Item Class. For reference purposes all the Extended Item Class methods are organized in the following four categories:

- Getting Innovator reference method:

  `getInnovator()` – see examples of usage in section 3.2.

- Lock methods:

  `lockItem()`

  `unlock`Item()

- Life Cycle method:

  `promote(…)`

- Workflow methods:

  `instantiateWorkflow(…)`

The following methods are obsolete and will be removed from future releases:

`startWorkflow(…)` –use `Item.apply("startWorkflow")` instead.

`cancelWorkflow(…)` - use `Item.apply("CancelWorkflow")` instead.

`closeWorkflow(…)` - use `Item.apply("closeWorkflow")` instead.

The detailed characteristic and usage illustration of methods above is left outside the scope of the document. See on-line API reference for more details.

4 Methods

Business logic in Aras Innovator is implemented using Method Items and is written in JavaScript, C#, or VB.Net often using the IOM to interact with Aras Innovator Items.

There are three ways to implement Methods in Aras Innovator on the server side:

- Item Action Methods which extend the Item Class and perform logic on Item instances.
- Generic Methods, which implement arbitrary logic.
- Server Events which implement logic on the context Item before and\or after the server operates on the Item.

Similarly there are three ways to implement Methods in Aras Innovator on the client side:

- Item Methods which extend the Item Class and perform logic on Item instances.
- Generic Methods, which implement arbitrary logic.
- Form, Field, and Grid Events which implement logic on client side UI events.
- Client Events that can be attached to an Item Type; triggered when a user's interaction with the Aras Innovator UI generates a new Item.

The Method Item has a comment Property that you can use to annotate the Method and can be seen when you search and review the Methods as mentioned above.

4.1 Item Actions Extend the Item Class

One purpose for Methods is to extend the Item Class. Methods extend the Item Class when they are bound as the related Item for 'Item Action' relationships on the ItemType. In the AML the Method name is the action attribute name for the Item tag.

```
<Item type="My ItemType" action="My Method" id="…"/>
```

The Method could be called using the IOM like this (all three examples below are equivalent and are written in C#):

```
1) Item myItem = this.newItem("My ItemType", "My Method");
   myItem.setID(this.getID());
   Item results = myItem.apply();
2) Item myItem = this.newItem("My ItemType");
   myItem.setID(this.getID());
   Item results = myItem.apply("My Method");
3) Item myItem = this.newItem();
   myItem.setID(this.getID());
   myItem.setType("My ItemType");
   myItem.setAction("My Method");
   Item results = myItem.apply();
```

4.1.1 Context Item

As mentioned before Item Action Methods are executed on an instance of Item which is called context item (read section 4.2 Built in Action Methods for more information on how context item is obtained for Item Action Methods). The context item must be referenced inside Item Action Methods as the `this` keyword in JavaScript, and C#, and the `Me` keyword Object in VB.Net. The context item is an instance of IOM Item class; correspondingly any methods of IOM Item class (see section 3.3 for more details) could be called on the context item, e.g this.getProperty("foo") (C#) or Me.getProperty("foo") (VB.Net).

Note: In order to be able to execute Method's code Aras Innovator plugs it into a particular template that provides required code attributes (method and class boundaries, import statements, etc.). Each supported language (JavaScript, C#, VB.NET, etc.) has several available templates in Aras Innovator. Everything written in the section is applied to default templates (there is one default template per supported language). Methods can explicitly redefine the template that is used during the method compilation. Usage of alternative templates and the methodology of writing valid Methods for them is left outside the scope of the document.

4.1.2 Methods are Item Factories

Methods follow the Factory design pattern in that they return an Item or Error Object. The 'Item Action' Method must return an Item, which often is the result of an Item.apply() method call; typically the last step in the business logic for the Method. There are several ways to create an Item; the following IOM methods return an Item Object: Item.apply(), Item.newItem(), Item.clone(), Innovator.newItem(), Innovator.newResult(), and Innovator.newError().

C#

```
Item qryItem = this.newItem(this.getType(), "get");
qryItem.setID(this.getID());
qryItem.setLevels(1);
return qryItem.apply();
```

If the Method needs to return an Error then use the Innovator.newError(text) method.

C#

```
Innovator innovator = this.getInnovator();
return innovator.newError("This method has <b>failed</b>.");
```

4.1.3 Handling the Wrong ItemType

Sometimes it is desirable to share the same Method for many ItemTypes. However, there are cases in which the Method is intended to be used only by an Item of a specific ItemType.

The way you can prevent the use of a Method with a context item of a wrong type is to throw an exception when the wrong ItemType is used (this is what the core Item Class methods do when the ItemType is not of the specific desired value). Here is a sample of what should be done in a Method that needs to operate on a specific type of Item:

C#

```
Innovator innovator = this.getInnovator();
if (this.getType() != "My ItemType")
{
  return innovator.newError("Item must be of type 'My ItemType'");
}
return null;
```

4.1.4 Methodology

One of the principle concepts in programming Aras Innovator is to write Methods called on Items. The 'Item Action' relationships on ItemTypes simulate Object Oriented programming, where the ItemType is the Class and 'Item Action' relationships to Methods are the Class methods. Literally the Method code is compiled dynamically to extend the Item Class with this method and calls it (see also notes to 0). Similar to class instance in OO programming the context item is an object on which Methods are performed. At the same time there are some peculiarities in how to use a context item in Aras Innovator's Item Action Methods. One important detail about Item Action Methods is that they must always return an Item which is considered to be the result of work done by the Item Action Method. An Item Action Method might work with context item but the context item is used here more as an input value (it still must be referenced as this or Me from inside Methods). It's highly recommended that Item Action Methods do not change the context item but rather create a new item that is returned from the method. You can use the combination of the `Item.getProperty(…)` method with the `Item.setProperty(…)` method to populate the new temporary Item or use the `Item.clone(…)` method to construct the new Item from the context item. If the developer of Method code chooses to modify the context item and not create a new item, the context item must be returned from the method.

4.2 Built in Action Methods

The Method name is passed as the action attribute for the <Item> tag in AML.

```
<Item type="My ItemType" action="My Method" id="…"/>
```

In addition to Method names as the action attribute value there is also a set of 'Built in Action Methods'. These are basically the same as Methods but you cannot find them in the database when searching the Method Items. Nevertheless, they are called the same way as ordinary Methods via the action attribute.

The following table is a reference for Built in Action Methods.

Built in Action Method	Comments
add	Add the Item as an instance of an ItemType.
update	Updates the Item. ○ The Item must be locked. ○ If the Item is versionable and is being updated for the first time since being locked, the update versions the Item applying the update to the new version, unless the version='0' attribute is specified, which disables the versioning.
purge	Deletes the version of the Item.
delete	Deletes all versions of the Item. The purge and delete are the same for non-versionable Items.
get	Gets the Item(s) and its configuration based on the AML Item configuration used to query the database.
getItemConfig	Returns the Item configuration as described by the standard AML query. The AML in and out are no different from the standard action='get'. The GetItemConfig is optimized by limiting the logic done between the SQL call and the AML result. The performance improvement is gained by limiting the features typically available in Innovator GetItem (no server events or access checking on the sub level Items).
GetItemRepeatConfig	Performs a recursive query on a related item. It will continue getting all of the related IDs of the parent item recursively. The number of recursive calls is dependent on the value of the repeatTimes variable.
edit	Locks, updates, and unlocks the Item.
create	Acts as a 'get' if the Item exists, otherwise acts as an 'add'.
merge	Acts as an 'edit' if the Item exists, otherwise acts as an 'add'.
lock	Locks the Item and is the same as the `Item.lockItem()` method.
unlock	Locks the Item and is the same as the `Item.unlockItem()` method.
version	Creates a new generation of an Item, clearing the locked_by_id of the originating Item and setting the locked_by_id in the new generation. It then applies an update to the newly created generation. The server events triggered in the following sequence: onBeforeVersion, onAfterVersion, onBeforeUpdate, onAfterUpdate. If the item is not versionable, an exception is thrown.

4.3 Generic Methods

Generic Methods are used to perform arbitrary business logic. They can be used to perform any logic you require and its input Item is up to you. They are called in the IOM with the `Innovator.applyMethod(…)` method.

4.3.1 Context Item

The context Item is the `this` keyword Object in JavaScript and C#, and is the `Me` Object in VB.Net. The XML data for the context Item is the XML submitted as the payload for the request and it may not be valid AML, just well formatted XML. It does not matter it is the input for the Generic Method and can be whatever you want it to be.

4.3.2 Methods are Item Factories

The Generic Method must return an Item or an Error similar to 'Item Action' Methods. Often the result of the Generic Method is some simple text or an HTML fragment. The text can be returned using the `Innovator.newResult(text)` method. If the Method needs to return an Error use `Innovator.newError(text)` method.

C#

```
Innovator innovator = this.getInnovator();
return innovator.newResult("This method was <b>successful</b>.");
OR
Innovator innovator = this.getInnovator();
return innovator.newError("This method has <b>failed</b>.");
```

C# Example

```
Innovator innovator = this.getInnovator();
Item item = this.newItem("User", "get");
Item results = item.apply();

int count = results.getItemCount();
if (count<1) return innovator.newError("No users found.");

StringBuilder content = new StringBuilder();
content.Append("<table>");

for (int i=0; i<count; ++i)
{
  Item user = results.getItemByIndex(i);
  content.Append("<tr><td>Login Name:</td><td>");
  content.Append(user.getProperty("login_name"));
  content.Append("</td></tr>");
}
content.Append("</table>");
return innovator.newResult(content.ToString());
```

4.3.3 Methodology

Typically all you need are simple name/value pairs as input for your Method and those are like Property tags for the Item. The body for the Generic Method is nested inside an `<Item>` tag so you can pass a name/value pair as arguments to the Generic Methods like ordinary Property tags.

The Item passed as the context Item can represent any Item you want including fictitious Items. You have the added advantage of continuing to use the IOM API to operate on the context Item Object.

4.4 Server Events

The purpose of Server Event Methods is to perform some custom actions either before (`OnBeforeXXX`) or after (`OnAfterXXX`) a particular server action (like add, delete, etc.) or fully replace the action processing on server (`OnXXX`). Detailed information about Aras Innovator server events can be found in section 4.4.3.

4.4.1 Context Item

In the case of the Server Event Method, the context item is a direct analogy of a class instance (i.e. object) in OO programming in the sense that the Method operates on its context item (as it was mentioned above the context item could be referenced as the `this` keyword in JavaScript and C#, and the `Me` keyword Object in VB.Net from inside the Method). In other words the purpose of a Server Event can be defined as 'changing the context item', so the modified context item is the result of work done by the Server Event Method. Of course, the Server Event Method doesn't necessarily have to alter its context item but rather perform some other actions (e.g. log some info; send e-mail; etc.); this is usually typical for Methods performed on `OnAfterXXX` event.

4.4.2 Methodology

A Server Event Method might return an Item only if it wants to return an error. Otherwise Server Event Method may not have a return statement.

C#

```
Innovator innovator = this.getInnovator();
return innovator.newError("This method failed.");
```

In the case Method is called as the `OnBeforeXXX` event and it returns an error, the context Item is replaced with an Error Item and is simply passed on through to the client. No further server action is taken. In the case of an `OnAfterXXX` event the server rolls back the transaction and passes the Error back on through to the client.

It's important to understand that Server Event Methods that are called on `OnBeforeXXX` events operate on the request AML sent from the client. Server Event Methods that are called on `OnAfterXXX` events operate on the response AML that server is about to send back to client and Server Event Methods that fully replace server actions (`OnXXX`) get client request AML as context item and must replace it with response AML that is passed on through to the client. In other words, it's important to remember that Server Event Methods called on `OnBeforeXXX` events are invoked before the server parses the request and after the Method is done the context item must have a valid request AML format (it could be modified by the method but it still should have a valid format so that server would be able to parse it). From the other side, Server Event Methods called on `OnAfterXXX` events are invoked after the server processed the request, and after the Method is done the context item must have a valid response AML format (it could be modified, e.g. Method could populate it with federated data, but it should be valid response AML, so that client would be able to parse it.)

4.4.3 Available Server Events

The following Server Events are currently available in Aras Innovator. Each of the following events is followed by a short description and an example of common use.

- `OnBeforeAdd`
  - Runs before an item is added to the database (through the add, create or merge actions.)
  - `OnBeforeAdd` methods are often used for validation purposes (e.g. to make sure the property values do not violate a business rule). The same method that is called OnBeforeAdd is often also called OnBeforeUpdate, to perform the same validation.
- `OnAfterAdd`
  - Runs after an item is added to the database (through the add, create or merge actions), but before it is returned to the client.
  - `OnAfterAdd` methods are used to synchronize with other items or with external systems (e.g. add a new part number to ERP).
- `OnAdd`
  - Runs in place of the built-in add action (via add, create or merge). Neither `OnBeforeAdd` nor `OnAfterAdd` events are called when using `OnAdd`.
  - An `OnAdd` method completely replaces the built-in add action, and is typically used for federated items. The method is expected to create the appropriate records in the database and form a proper AML response. Failure to do either is likely to result in an error.
- `OnBeforeUpdate`
  - Runs before an item is updated in the database (through the update, edit or merge actions.)
  - `OnBeforeUpdate` methods are often used for validation purposes (often along with OnBeforeAdd). The request may either be rejected completely (by returning an error) or modified to conform to the proper rules.
- `OnAfterUpdate`
  - Runs after an item is updated in the database (through the update, edit or merge actions), but before it is returned to the client.
  - `OnAfterUpdate` methods can be used to synchronize with other items or with external systems (e.g. updating a part description in ERP).

- `OnUpdate`
  - Runs in place of the built-in update action (via update, edit or merge). Neither `OnBeforeUpdate` nor `OnAfterUpdate` events are called when using `OnUpdate`.
  - An `OnUpdate` method completely replaces the built-in update action, and would typically be used for federated items. The method is expected to modify the appropriate records in the database and form a proper AML response.
- `OnBeforeDelete`
  - Runs before an item is deleted (through the delete or purge actions.)
  - `OnBeforeDelete` methods are typically used to cancel a delete operation based on a business rule.
- `OnAfterDelete`
  - Runs after an item is deleted (through the delete or purge actions.)
  - `OnAfterDelete` methods would be used to synchronize with other items or with external systems (e.g. remove a record from ERP.)
- `OnDelete`
  - Runs in place of the built-in delete action. Neither `OnBeforeDelete` nor `OnAfterDelete` events are called when using `OnDelete`.
  - An `OnDelete` method completely replaces the built-in delete action, and is typically used for federated items. The method is expected to remove the appropriate records in the database and form a proper response.
- `OnBeforeGet`
  - Runs before a search.
  - `OnBeforeGet` methods are typically used to add additional criteria to a search, based on business rules. For example, the method might find the default location of the user and add that location as criteria for the query.
- `OnAfterGet`
  - Runs after a search is executed, but before the results are returned.
  - `OnAfterGet` methods are commonly used to populate federated properties. This might involve performing calculations on other properties or extracting data from an external system. Once the value of the federated property is set, it appears to the client like any other property.
- `OnGet`
  - Runs in place of the built-in get action. Neither `OnBeforeGet` nor `OnAfterGet` events are called when using `OnGet`.
  - An `OnGet` method completely replaces the built-in get action, and is typically used for federated items. The method is expected to retrieve the appropriate records in the database and form a proper AML response.
- `OnBeforeCopy`
  - Runs before an item is copied (via the copy action.)
  - An `OnBeforeCopy` method might be used to cancel a copy operation that violates a business rule.

- `OnAfterCopy`
  - Runs before an item is copied (via the copy action.)
  - `OnAfterCopy` methods can be used to set properties of the new item created by the copy.
- `OnBeforeLock`
  - Runs before an item is locked (via the lock or edit actions)
  - An `OnBeforeLock` method may be used to prevent an item from being locked based on business rules.
- `OnAfterLock`
  - Runs after an item is locked (via the lock or edit actions.)
  - `OnAfterLock` methods may be used to synchronize locks with other items.
- `OnBeforeUnlock`
  - Runs before an item is unlocked.
  - An `OnBeforeUnlock` method may be used to prevent an item from being unlocked based on business rules.
- `OnAfterUnlock`
  - Runs after an item is unlocked.
  - `OnAfterUnlock` methods may be used to synchronize locks with other items.
- `OnBeforeVersion`
  - Runs before an item is versioned (through the version, update, edit or merge actions.)
  - `OnBeforeVersion` methods are typically used to cancel a version operation based on a business rule.
- `OnAfterVersion`
  - Runs after an item is versioned (through the version, update, edit or merge actions.)
  - An `OnAfterVersion` method might be used to set properties of the new item version.
- `OnBeforeMethod`
  - Runs before Server Action Method.
- `OnAfterMethod`
  - Runs after Server Action Method.
- `OnGetKeyedName`
  - Runs when the system generates the keyed_name for an item.
  - Used to override the standard logic for generating keyed names.

4.4.4 Polymorphic ItemTypes Server Event Inheritance

The ability to define a server event against a Polymorphic ItemType was introduced with the Aras Innovator 12.0 release. The benefits of defining the server event against the Polymorphic ItemType is that the event will be processed for all poly-sources.

For example, if you want a single server event to be triggered against all `CAD`, `Document`, and `Part` ItemTypes, you could simply add an event handler to a server event on the `Change Controlled Item` ItemType. Once defined against the Polymorphic ItemType, all poly sources will inherit the event handler which will be executed every time the event occurs on any of poly sources.

You can view all inherited methods against an ItemType by selecting the `Inherited Server Events` tab for the ItemType you are working with.

4.4.5 Required Server Events

The ability to mark a server event as "required" was introduced with the release of Aras Innovator 12.0. Server Events that are marked as required cannot be ignored by using the `serverEvents=0` attribute as part of AML request.

To set a server event as required, so that it is not ignored, you need to set the `is_required` Boolean property to 1 for the ServerEvent.

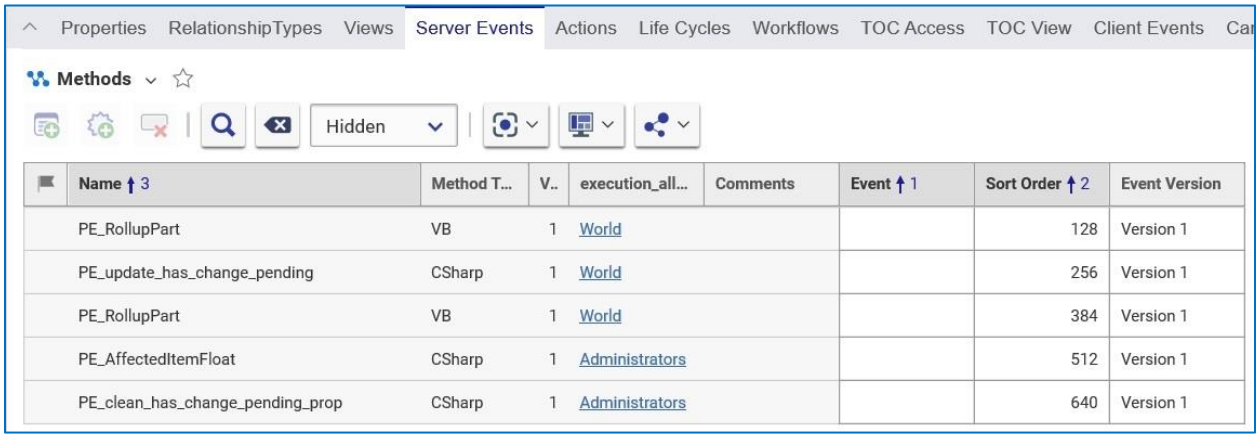


Figure 1.

4.4.6 Server Event Version

A new version for server events was introduced with the release of Aras Innovator 12. The new version of server events allows for improved performance when there is a group of items passed to be acted upon.

By default all server events follow the standard behavior that has been seen in previous releases of Aras Innovator (Version 1). The new version (Version 2) is described for each operation in the following sections.

4.4.6.1 onAfterUpdate, onAfterAdd, and onAfterVersion Version 2

The onAfterUpdate, onAfterAdd, and onAfterVersion Version 2 event is designed for use when *where* or *idList* attribute is specified on the *<Item …>* node of the AML request. The server event is executed only once for the entire group, as opposed to once per item as the Version 1 event does. The context item contains the results of all the items applied as part of the update statement.

4.4.6.2 onBeforeCopy/onAfterCopy Version 2

When a source item is versioned or is cloned it forces cloning of all existing relationships that belong to the Item. The onBeforeCopy and onAfterCopy Version 2 allows for onBeforeCopy and onAfterCopy server events to be triggered on the relationship items when a version/copy of a parent item exits. These events are not triggered on a Version 1 server event. These events should be placed on the is_relationship=1 ItemType.

4.5 Client Events

There are several Events available on the client side, including:

- Form Events
- Field Events
- Grid Events
- Item Type Events
- Item Actions

4.5.1 Context Item

The `this` keyword context Object is an Item Object for Item Actions. However, the context Object is not the Item Object for Form, Field, and Grid Events. This context Object is the browser document (DOM) Object for the Form and Grid Events and is the Field Object for Field Events.

The context Item Object for Form, Grid, and Field Events is the `document.thisItem` Object, which is an Item Object and should be used with the IOM API. For relationship grid events use `parent.thisItem`, which is a pointer to the `document.thisItem` Object.

4.5.2 Form Events

The Form Events are the HTML page events; for example, `onLoad`, `onUnload`, `onResize`, `onMouseDown`, `onMouseUp`, and others (refer to the 'Form Events' List in Aras Innovator for the complete list of available events.)

You bind your Method to the Form Event using the Form Tool. Select the Form Event tab and add the event relationships as shown below:

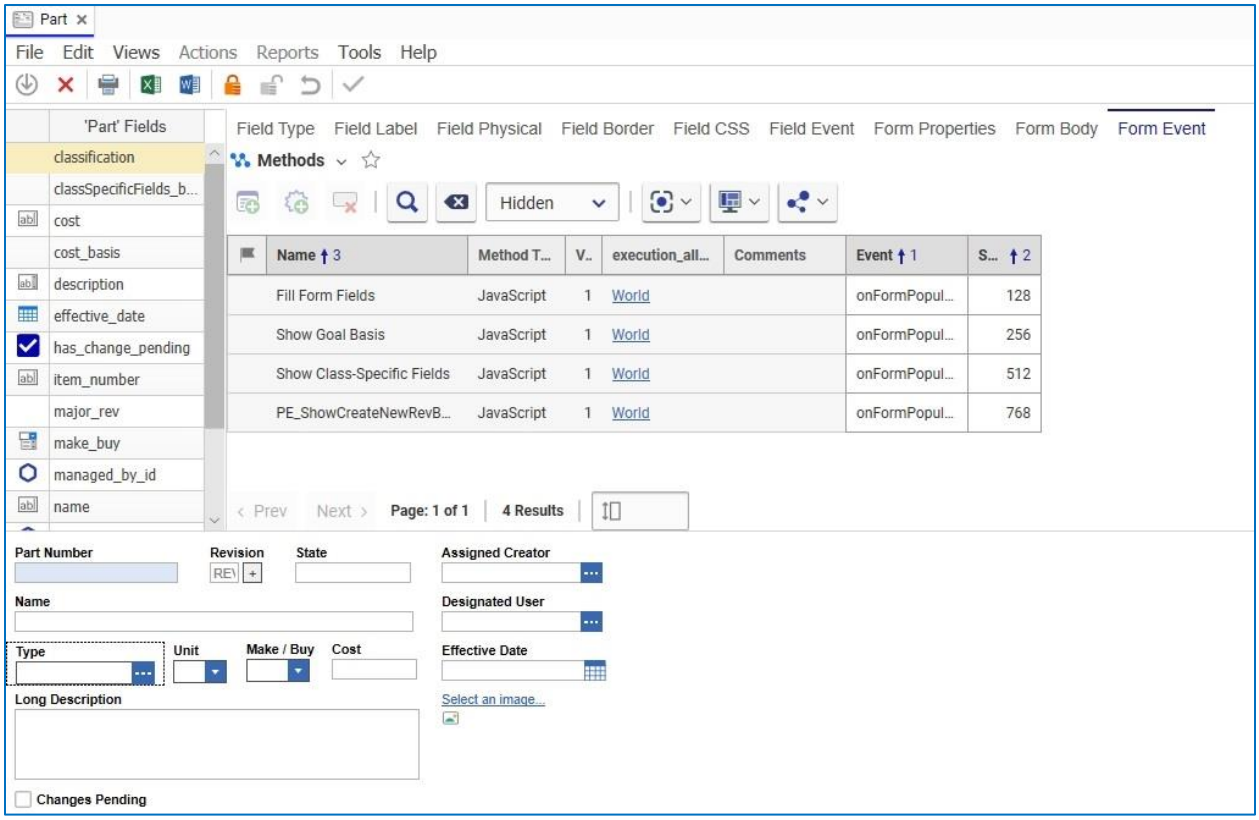


Figure 2.

4.5.3 Field Events

The Field Events are the HTML field events; for example, onSelect, onClick, onChange, onBlur, onFocus, and others (refer to the 'Field Events' List in Aras Innovator for the complete list of available events.) Use the following procdure:

1. Bind your Method to the Field Event using the Form Tool.
2. Select the Field by clicking on it in the canvas area in form Tool or from the Fields grid in the upper left hand corner of the Form Tool.
3. Select the Field Event tab and add the event relationships as shown below:

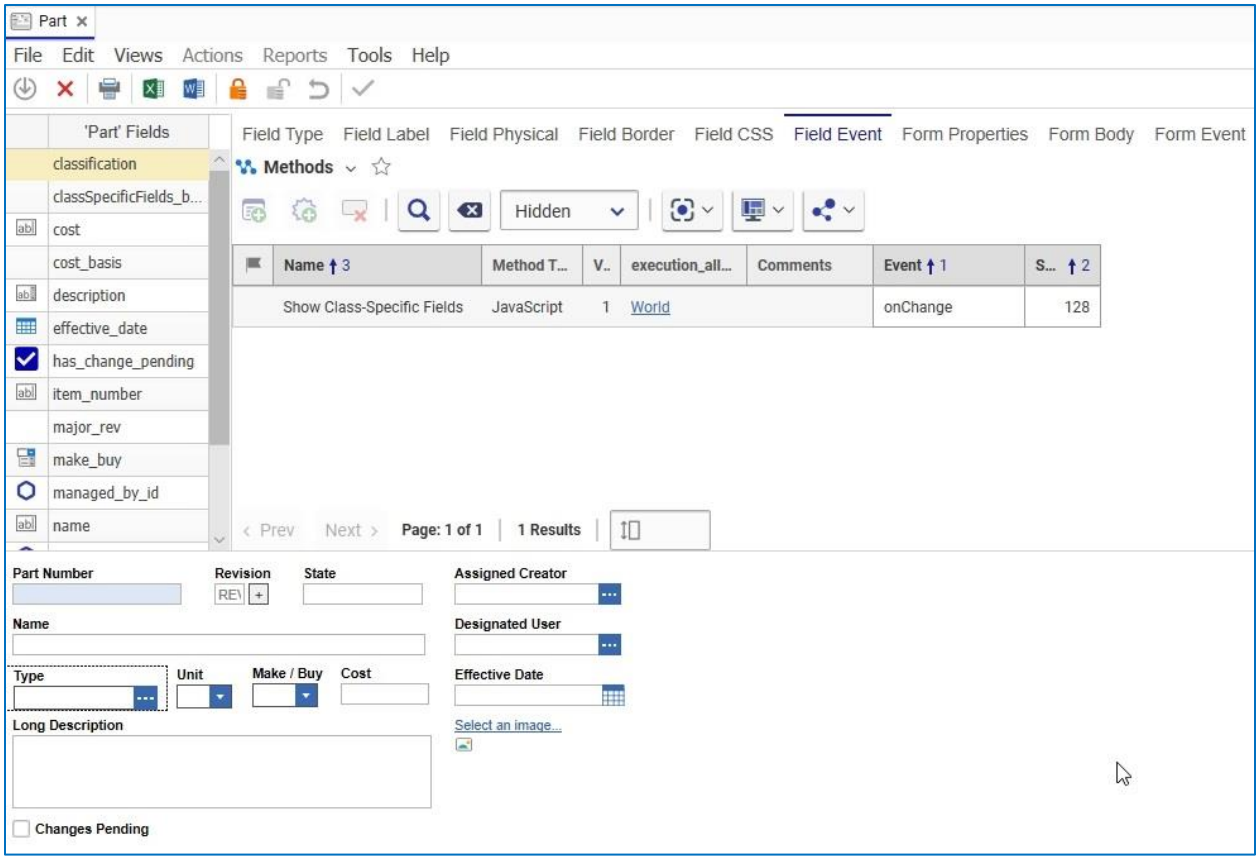


Figure 3.

4.5.4 Grid Events

Grid Events are the events for the grid control, which is used in the Relationships tab area for tear off Item windows. The grid events occur on the row (section 4.5.4.1) and on the cell (section 4.5.4.2.)

Like Server Events you bind a Method as the callback for the event as the Grid Event relationship on the RelationshipType Item, and as the Grid Event relationship on the Property Item.

The Method gets at least three arguments: relationshipID, relatedID, gridApplet. The relationshipID is the ID for the relationship Item for the selected row. The relatedID is the ID for the related Item for the selected row. And the gridApplet is a handle to the grid control object. The relatedID maybe empty if there is no related Item for the relationship row. The relationshipID is also the ID for the grid control row. Edit the RelationshipType Item and add Grid Events relationships as shown below:

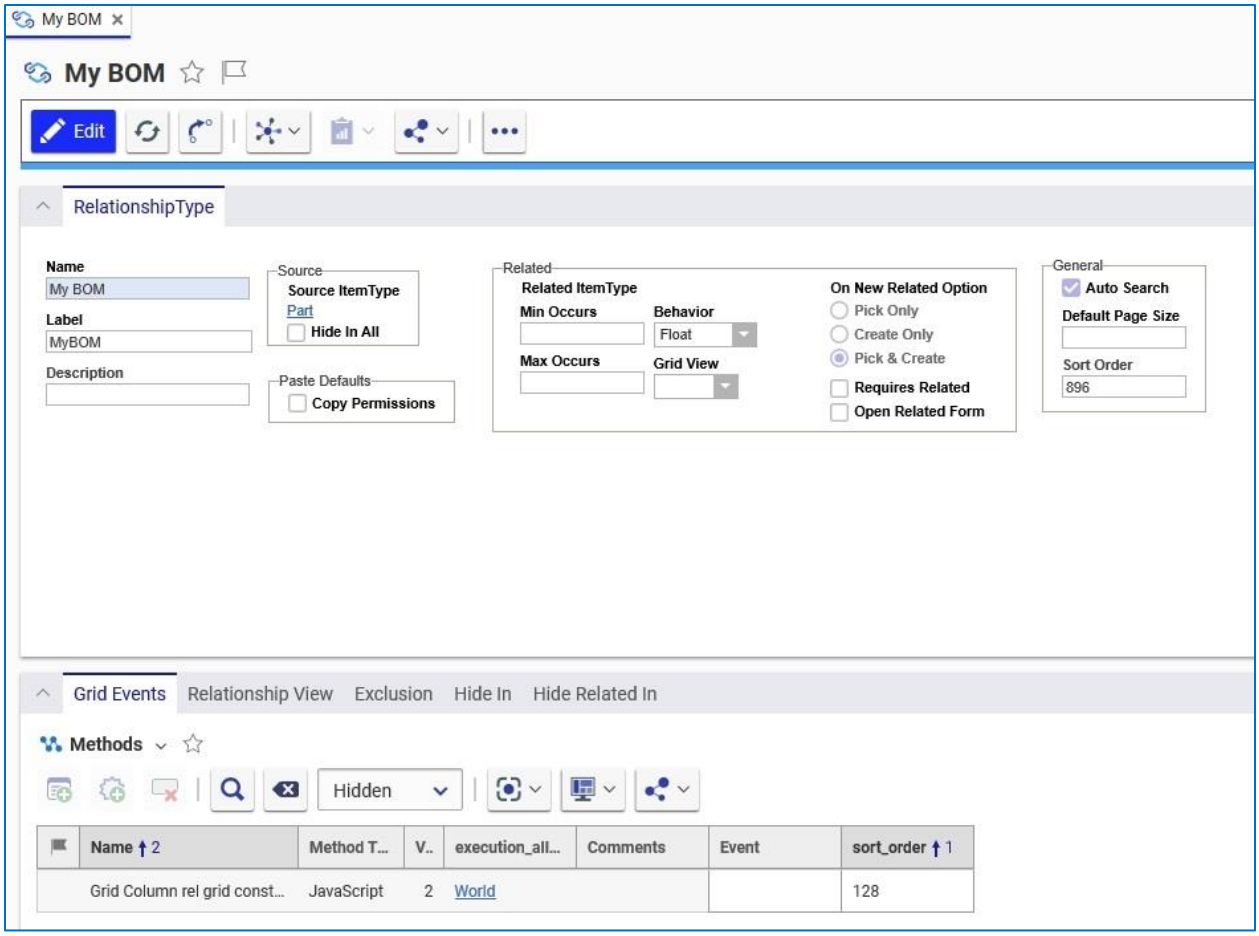


Figure 4.

4.5.4.1 Row Events

The row events include:

Event	Comment
onSelectRow	Fires when the row is selected.
onInsertRow	Fires when the row is inserted.
onDeleteRow	Fires when the row is deleted.

The Method for the event is called with three arguments:

Argument	Type	Comment
relationshipID	String	The ID for the relationship Item. This is also the selected row ID for the grid control.

Argument	Type	Comment
relatedID	String	The ID for the related Item. The relatedID maybe empty if there is no related Item for the relationship row.
gridApplet	GridControl	The handle to the grid control.

4.5.4.2 Cell Events

Edit the Property Item and add Event relationships as shown here:

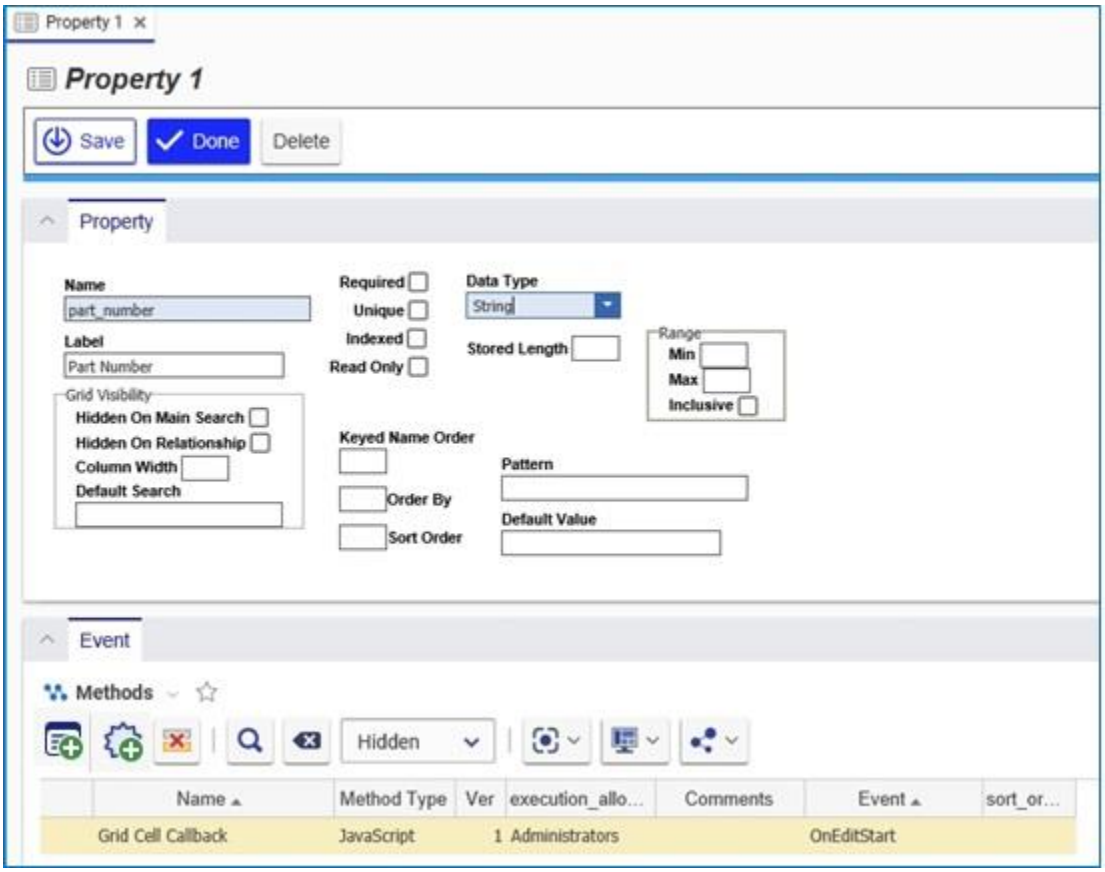


Figure 5.

The cell events include:

Event	Comment
onEditStart	Fires when the cell gets focus.
onEditFinish	Fires when the cell loses focus.
onChangeCell	Fires when the cell value changes.

Event	Comment
Default Search	
onSearchDialog	

The Method for the event is called with five arguments:

Argument	Type	Comment
relationshipID	String	The ID for the relationship Item. This is also the selected row ID for the grid control.
relatedID	String	The ID for the related Item. The relatedID maybe empty if there is no related Item for the relationship row.
gridApplet	GridControl	The handle to the grid control.
propertyName	String	The name of the Property for the cell column selected.
colNumber	Integer	The column position number in the grid

4.5.5 Item Type Events

Client Events that can be attached to an Item Type are triggered when your UI actions generate a new Item. These events are triggered from the client interface regardless of UI context or where in the GUI the new Item creation was initialized. These events would be triggered universally from the Main Menu, the TOC RMB menu, the Main Grid RMB menu, Relationship Grid, etc.

For Item Type new item creation, 3 events have been implemented. One event is triggered before a new Item is created; another event is triggered after an Item has been created; the third event replaces the standard client 'new Item' logic.

- `onBeforeNew`: Method runs prior to a new Item creation. It has the ability to cancel subsequent client operations (i.e., form opening). It has the ability to cancel creation of new Items.

  This event is often used to validate current conditions and determine if it is ok to create a new Item.

- `onAfterNew`: Method runs after a new Item is created. Subsequent standard client logic is executed following method completion (i.e., form opening). Method is passed a new Item.

  This event is often used to populate a new item with data and open custom dialogs.

- `onNew`: The method replaces the standard 'new Item' client behavior.

  This event is used in special situations where a solution must maintain full control over the new Item creation process.

Note: It is possible that both `onBeforeNew` and `onAfterNew` events are assigned to the same ItemType and therefore executed sequentially.

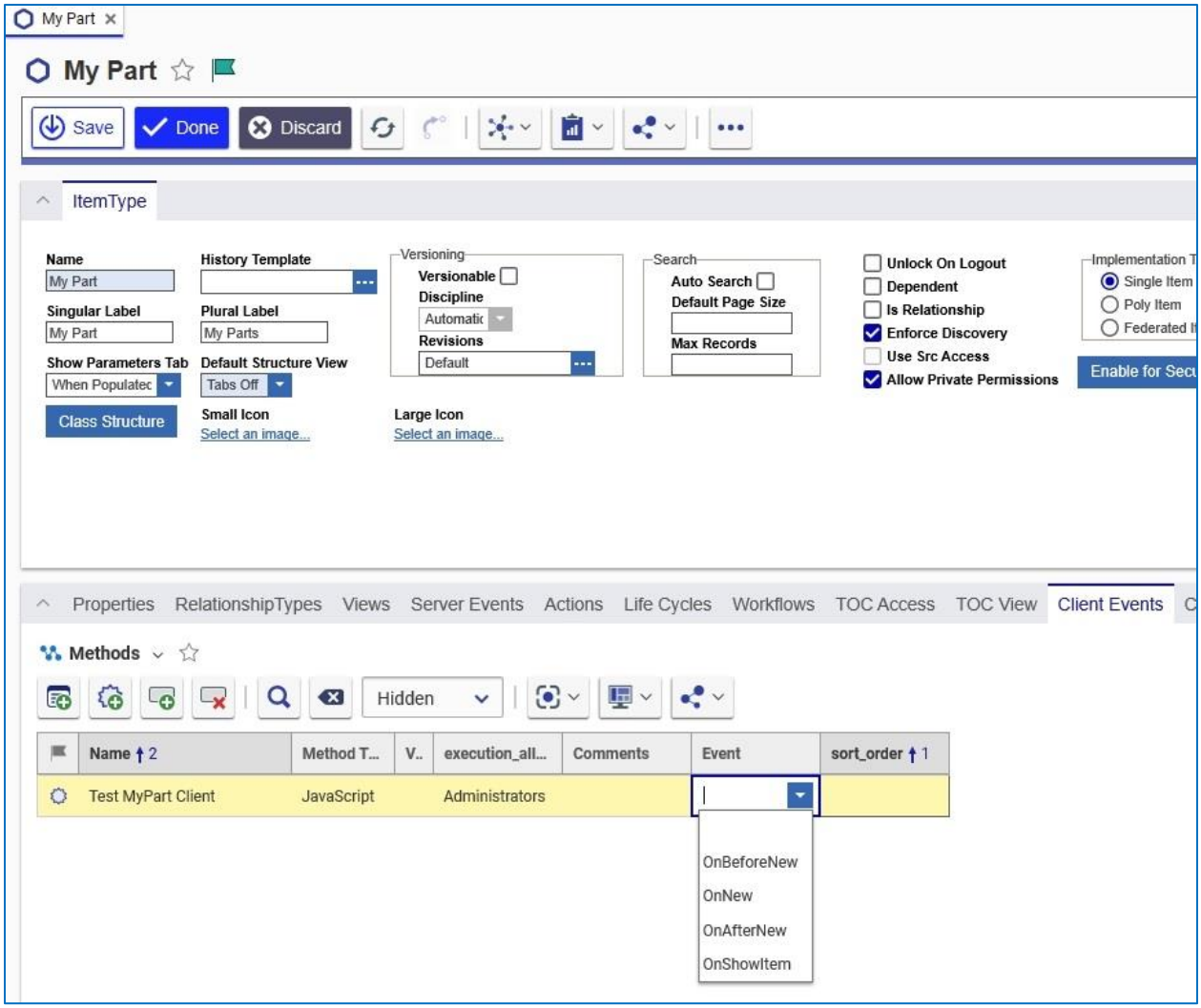


Figure 6.

4.5.6 Item Actions and Server Event

The Methods related to the ItemType via the 'Item Action' relationship are called via the action attribute and the `Item.apply()` method. Review them by searching the 'Item Actions' Tab on the ItemType.

The Methods related to the ItemType via the 'Server Event' relationship are called by the server as pre and post event callbacks for the primitive server actions: add, update, delete, and get. Review them by searching the 'Server Events' Tab on the ItemType.

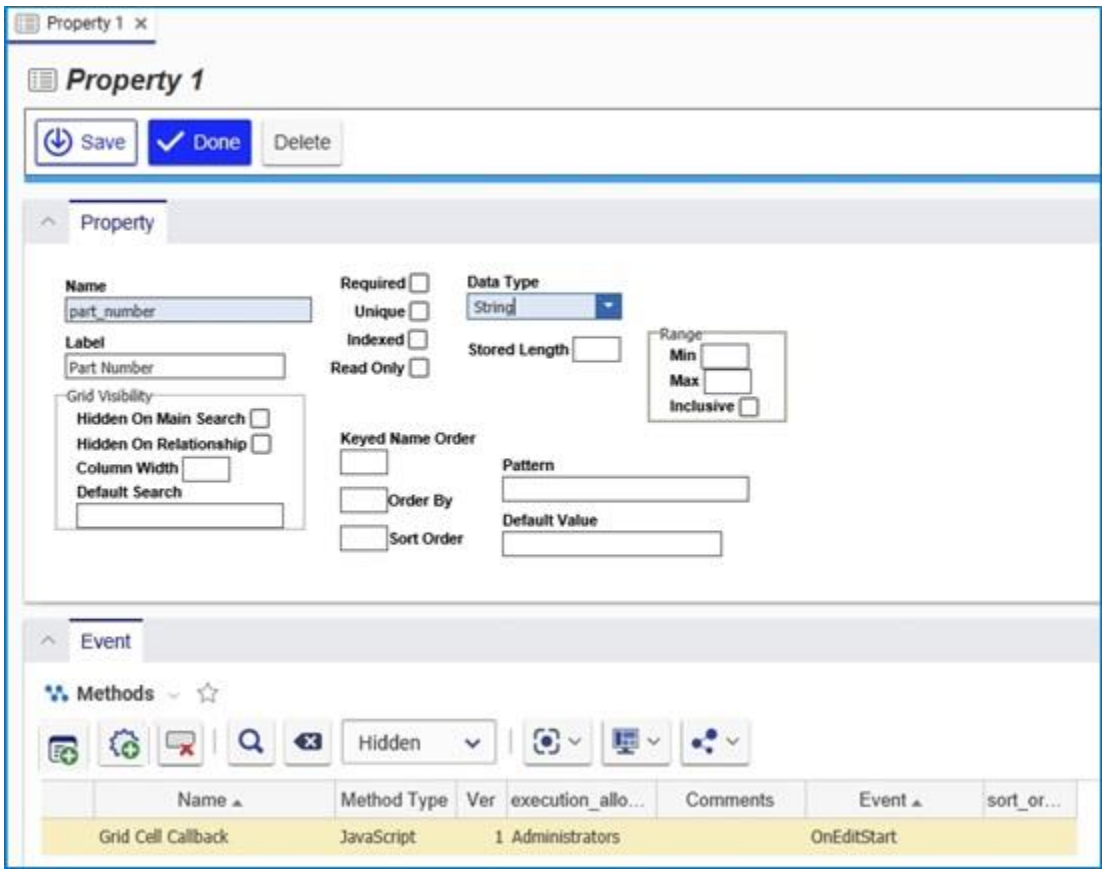
Review the Generic Methods by searching the Method Items.

5 CUI Overview

The Aras CUI was introduced to include additional types of controls that are dynamic and depend on the context in which they appear. A number of Aras solutions use this functionality, such as QMS, MPP, and Technical Publications. The steps below allow you to customize the command bar to customize CUI functionality. The 'Client Presentation' item, which is the root of all CUI items can be found in the Administration > Configuration > Client Presentation section of the Table of Contents.

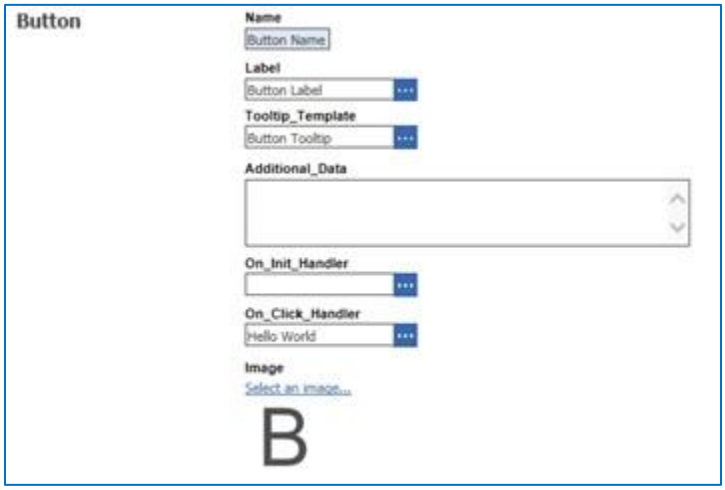
5.1 Adding a button on a global scope

- Go to the Table of Contents: **Administration > Configuration > Client Presentation**.
- Select Global Presentation Configuration
- In the Command Bar Section tab, right-click on item with Location: Main Window Header > View "Command Bar Section"



- Lock the Command Bar Section Item for edit.
- In the Command Bar Item tab, select Create Related from the dropdown.
- Click New relationship.
- Select Button in the new dialog.

- Click OK. A new line will appear in the Command Bar Item tab.
- Set Action to Add, and Identity to World.
- Right-click and select "View Command Bar Item"
- Set the values for the fields in the form.



- Save, Unlock, and Close all items.
- Log out and back in to see the changes
- Clicking on the button will run the On_Click_Handler.

5.2 Removing a button from ItemType scope

- Go to the Table of Contents: Administration > ItemTypes.
- Run a search for the ItemType.
- Open the ItemType for editing.
- In the Client Style tab, right-click on the first entry > View "Presentation Configuration".
- In the Presentation Configuration, create a New Related Command Bar Section Item with Classification: Data Model.
- In the new row, set the Identity to World.
- Right-click on the new row > View "Command Bar Section"
- Set the property values. In the below example, we are removing the Print button from the Tearoff Window.
  - Name: Remove Print
  - Location: Tearoff Window Toolbar
- In the Command Bar Item tab, select Pick Related from the dropdown.
- Select the Item you wish to remove. For example, run a search for '*twt_normal_print', and return the selected Item.
- Set the Action of the new row to Remove.

- Save, Unlock, and Close all items.
- Log out and back in to see the changes.

5.3 Defining an Identity on the button

- Go to the Table of Contents: Administration > Configuration > Client Presentation.
- Select Global Presentation Configuration.
- In the Command Bar Section tab, right click on an Item with the location: Tearoff Window Toolbar > View "Command Bar Section".
- Lock the Command Bar Section for editing.
- In the Command Bar Section tab, set your chosen Identity on the button of your choice.
- Save, Unlock, and Close all items.
- Log out and back in to see the resulting changes.

6 Aras Innovator Methodology

This section is a summary of the Aras Innovator Methodology. Following this methodology helps you build better quality Aras Innovator business logic more quickly; plus it is easier to understand and maintain the code.

- The AML is the language that drives the Aras Innovator Server.
- AML documents contain Items, Structure, and Logic, so they are scripts.
- The Aras Innovator Server is a message based system in that it accepts AML scripts as messages and returns AML messages. AML document, AML script, AML message all mean the same thing.
- The IOM is the Object API used to build and apply AML messages.
- Methods implement business logic using the IOM API.
- Methods extend the Item Class when used as "Item Action" relationships on the ItemType, which simulates Object Oriented programming, where the ItemType is the Class and "Item Action" relationships to Methods are the Class methods.
- Methods are also generic arbitrary business logic that can be called like a sub routines from other Methods using the IOM `Innovator.applyMethod(…)` method.
- Methods follow the Item Factories design pattern; they should return a new Item and not side effect the context Item.
- Server Events are the exception because the purpose is to intercept and operate on the AML before the server parses it, and before the AML is returned to the client after the server parses it. So you modify the context Item and return nothing.
- Implement changes/edits to the context Item in the OnBefore Event by altering the AML before the server parses it. Refrain from attempting to update the context Item after the server has already operated on it in the OnAfter Event. Use the OnAfter Events to update/include federated data in the response AML.
- Use the `select`, `page`, and `pagesize` attributes for the AML queries to optimize the performance for the request.
- Use the generic IOM methods to construct the AML queries rather than convenience methods like `getItemBy_`, `getRelationships()`, or use the `levels` attribute because the convenience methods typically return far more data than required imposing a performance hit.
- The context Item is the keyword `this` Object for JavaScript and C#, and the `Me` Object for VB.Net.
- The context Item for Generic Methods is any XML you want but it is highly recommended that you continue to use AML to represent your data. This provides the benefit of using the IOM to manage the input for the Generic Methods.
- Attributes are used to pass control switches to the Method. You can invent your own because they are a simple way to pass meta-data to the Method.

7 Cookbook

This section is a Cookbook of recipes to help you solve common tasks while developing Methods. The examples are shown in JavaScript, C# and VB.Net when possible.

7.1 Create an Aras Innovator Object

You need an Aras Innovator Object to return a `newResult()` or `newError()` Item.

Technique

There are basically two ways to create a new *Innovator* object; by getting the *Innovator* from the *Item* object or (only in JavaScript) calling the Class constructor.

JavaScript

```
var myInnovator = new Innovator();
var myInnovator = this.getInnovator();
```

C#

```
Innovator myInnovator = this.getInnovator();
```

7.2 Create an Item Object

You need an Item Object to submit a query or to add an Item.

Technique

There are basically two ways to create a new Item Object; by calling the factory methods on the *Item* object or *Innovator* object or (only in JavaScript) calling the Class constructor.

JavaScript

```
var myItem = new Item();
var myItem = this.newItem(myType,myAction);
var myInnovator = this.getInnovator();
var myItem   = myInnovator.newItem(myType,myAction);
var myResult = myInnovator.newResult(resultText);
var myError  = myInnovator.newError(errorMessage);
```

C#

```
Item myItem = this.newItem(myType,myAction);

Innovator myInnovator = this.getInnovator();
Item myItem   = myInnovator.newItem(myType,myAction);
Item myResult = myInnovator.newResult(resultText);
Item myError  = myInnovator.newError(errorMessage);
```

7.3 Query for an Item

You want to query for an Item that you know by id and type.

Technique

There are a few ways to get an Item when you know its id and type, the simplest being the `Innovator.getItemById()` method. However, if you need to be granular about your request then building the query using the IOM is required. This provides the ability to include controls to limit the results and define the structure to be returned for the Items found.

JavaScript

```
var qryItem = this.newItem(myType,"get");
qryItem.setID(myId);
var results = qryItem.apply();

var myInnovator = this.getInnovator();
var results = myInnovator.getItemById(myType, myId);
```

C#

```
Item qryItem = this.newItem(myType,"get");
qryItem.setID(myId);
Item results = qryItem.apply();

Innovator myInnovator = this.getInnovator();
Item results = myInnovator.getItemById(myType, myId);
```

7.4 Query and iterate over a set of Items

You want to query for the Items that match some criteria and generate an HTML Table as the result.

Technique

There is no difference in setting up a query for a single Item or for many. Only the criteria define the set size returned. In this recipe you create an Item and populate the query criteria, apply it, and iterating over the Items returned producing a HTML `<TABLE>` fragment.

JavaScript

```
var qryItem = this.newItem("Part","get");
qryItem.setAttribute("select","item_number,description,cost");
qryItem.setProperty("cost", "100");
qryItem.setPropertyCondition("cost", "gt");
var results = qryItem.apply();
var count = results.getItemCount();
var content = "<table>";
for (var i=0; i<count; ++i)
{
  var item = results.getItemByIndex(i);
  content += "" +
    "<tr>" +
      "<td>" + item.getProperty("item_number") + "</td>" +
      "<td>" + item.getProperty("description") + "</td>" +
      "<td>" + item.getProperty("cost") + "</td>" +
    "</tr>";
}
content += "</table>";
return content;
```

C#

```
Item qryItem = this.newItem("Part","get");
qryItem.setAttribute("select","item_number,description,cost");
qryItem.setProperty("cost", "100");
qryItem.setPropertyCondition("cost", "gt");
Item results = qryItem.apply();
int count = results.getItemCount();
int i;
string content = "<table>";
for (i=0; i<count; ++i)
{
  Item item = results.getItemByIndex(i);
  content += "" +
    "<tr>" +
      "<td>" + item.getProperty("item_number") + "</td>" +
      "<td>" + item.getProperty("description") + "</td>" +
      "<td>" + item.getProperty("cost") + "</td>" +
    "</tr>";
}
content += "</table>";
Innovator innovator = this.getInnovator();
```

```
return innovator.newResult(content);
```

7.5 Query for an Item and return its configuration

You want to query for an Item and return its configuration in the results.

Technique

To query for an Item and retrieve its structure you build the query as the structure you want returned. Use the IOM methods to add the relationships you want and build the structure in the Item. The server returns the structure that follows the request structure.

This recipe illustrates several related concepts together, which are how to get a set of Items from an Item and how to iterate over the set, plus how to get the related Item from the relationship Item.

JavaScript

```
var innovator = this.getInnovator();

// Set up the query Item.
var qryItem = this.newItem("Part","get");
qryItem.setAttribute("select","item_number,description,cost");
qryItem.setID(myId);

// Add the BOM structure.
var bomItem = this.newItem("Part BOM","get");
bomItem.setAttribute("select","quantity,related_id(item_number,description,co
st)");
qryItem.addRelationship(bomItem);

// Perform the query.
var results = qryItem.apply();

// Test for an error.
if (results.isError()) {
  top.aras.AlertError("Item not found: " + results.getErrorDetail());
  return;
}

// Get a handle to the BOM Items.
var bomItems = results.getRelationships();
var count = bomItems.getItemCount();

// Create the results content.
var content = "<table border='1'>" +
  "<tr>" +
    "<td>Part Number</td>" +
    "<td>Description</td>" +
    "<td>Cost</td>" +
    "<td>Quantity</td>" +
  "</tr>";

// Iterate over the BOM Items.
for (var i=0; i<count; ++i)
{
// Get a handle to the relationship Item by index.
```

```
  var bom = bomItems.getItemByIndex(i);
// Get a handle to the related Item for this relationship Item.
  var bomPart = bom.getRelatedItem();

  content += "<tr>" +
      "<td>" + bomPart.getProperty("item_number") + "</td>" +
      "<td>" + bomPart.getProperty("description") + "</td>" +
      "<td>" + bomPart.getProperty("cost") + "</td>" +
      "<td>" + bom.getProperty("quantity") + "</td>" +
    "</tr>";
}
return content + "</table>";
```

C#

```
Innovator innovator = this.getInnovator();

// Set up the query Item.
Item qryItem = this.newItem("Part","get");
qryItem.setAttribute("select","item_number,description,cost");
qryItem.setID(myId);

// Add the BOM structure.
Item bomItem = this.newItem("Part BOM","get");
bomItem.setAttribute("select","quantity,
related_id(item_number,description,cost)");
qryItem.addRelationship(bomItem);

// Perform the query.
Item results = qryItem.apply();

// Test for an error.
if (results.isError()) {
  return innovator.newError("Item not found: " + results.getErrorDetail());
}

// Get a handle to the BOM Items.
Item bomItems = results.getRelationships();
int count = bomItems.getItemCount();
int i;

// Create the results content.
string content = "<table border='1'>" +
  "<tr>" +
    "<td>Part Number</td>" +
    "<td>Description</td>" +
    "<td>Cost</td>" +
    "<td>Quantity</td>" +
  "</tr>";

// Iterate over the BOM Items.
for (i=0; i<count; ++i)
{
// Get a handle to the relationship Item by index.
  Item bom = bomItems.getItemByIndex(i);
// Get a handle to the related Item for this relationship Item.
```

```
  Item bomPart = bom.getRelatedItem();

  content += "" +
    "<tr>" +
      "<td>" + bomPart.getProperty("item_number") + "</td>" +
      "<td>" + bomPart.getProperty("description") + "</td>" +
      "<td>" + bomPart.getProperty("cost") + "</td>" +
      "<td>" + bom.getProperty("quantity") + "</td>" +
    "</tr>";
}
content += "</table>";

return innovator.newResult(content);
```

7.6 Query using AML to construct the query

You want to perform a query using the AML to construct the query criteria.

Technique

Create an Item Object but use the `Item.loadAML()` method to populate the Item.

JavaScript

```
var innovator = new Innovator();
var qryItem = innovator.newItem();
qryItem.loadAML(
  "<Item type='Part' action='get' select='item_number,description,cost'>" +
    "<item_number condition='like'>1%</item_number>" +
    "<Relationships>" +
      "<Item type='Part BOM' action='get' select='quantity'>" +
        "<quantity condition='gt'>1</quantity>" +
      "</Item>" +
    "</Relationships>" +
  "</Item>"
);

var resultItem = qryItem.apply();
if (resultItem.isError()) {
  top.aras.AlertError("Item not found: " + resultItem.getErrorDetail());
  return;
}

var count = resultItem.getItemCount();
for (i=0; i<count; ++i) {
  var item = resultItem.getItemByIndex(i);
}
```

7.7 Query for the Item next promotion states

You want to get the next promotion states for an Item and use the states as the choices for a dropdown control.

Technique

Use the item action `getItemNextStates` to get the next state values. This recipe assumes there is a select input field on the form for us to populate with state values.

HTML

```
<select id="mySelect"></select>
```

JavaScript

```
var results = document.thisItem.apply("getItemNextStates");
var nextStates = results.getItemsByXPath('//Item[@type="Life Cycle State"]');
var count = nextStates.getItemCount();

var oSelect = document.getElementById('mySelect');
for (var i=0; i<count; ++i) {
  var item = nextStates.getItemByIndex(i);
  var opt  = document.createElement('option');
  opt.text = item.getProperty('name');
  oSelect.add(opt);
}
```

7.8 Query using relationships as search criteria

You want to search for an Item using the relationship and related Items as search criteria. In this recipe, we get the User Item that matches the Identity name as an Alias relationship to the User Item.

Technique

The following are some key points to understand when constructing an AML query:

1. Use the get action on the relationship Items to include it as search criteria.
   a. Without the get action the relationship Item is ignored as search criteria.
   b. The relationship Items are also returned. Currently there is no way to use relationships as search criteria and not return them in the results, as you can with the related Item described below.
2. Include the related_id property name in the select attribute for the relationship Item if you want to return the related Item nested inside the related_id property in the results.

   `<Item type="Part BOM" select="quantity,related_id"/>`
   Use () to include the select attribute value for the related Item inside the select attribute for the relationship Item.
   `<Item type="Part BOM" select="quantity,related_id(item_number,description)"/>`
3. The select attribute for the nested Item tag for the related_id property has higher precedence over the select value inside the () for the relationship's select attribute.
4. The get action is not required for the nested Item tag for the related_id property to include it as search criteria.

These two AML scripts are equivalent queries for selecting the name property for the related Item:

```
<Item type="User" action="get" select="first_name,last_name,email">
  <Relationships>
    <Item type="alias" action="get" select="related_id(name)"/>
  </Relationships>
</Item>
<Item type="User" action="get" select="first_name,last_name,email">
  <Relationships>
    <Item type="alias" action="get" select="related_id">
      <related_id>
        <Item type="Identity" action="get" select="name"/>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

Clearly the first example is simpler and requires less coding (referring to the IOM logic that would construct the AML) and is the recommended style when all you require is specifying a select for the related Item for the query.

But the second style opens the opportunity to now include additional search criteria for the related Item.

AML

```
<Item type="User" action="get" select="first_name,last_name,email">
  <Relationships>
    <Item type="Alias" action="get" select="related_id">
<!--
This get will limit root Items to only those that match the relationship
criteria.
The get action is required otherwise the criteria are ignored.
To include the nested Item tag for the related_id include the property name
in the select attribute for the relationship Item.
Can include the select attribute value for the related Item inside ()
i.e. related_id(name)
-->
      <related_id>
        <Item type="Identity" action="get" select="keyed_name">
<!--
This get has no effect and the search will work with or without it.
It is recommended that you include it because the AML parser may be stricter
in the future.
The select attribute over rules the parent relationships select.
-->
          <name>Larry Bird</name>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

JavaScript

```
var innovator = new Innovator();
var qry = innovator.newItem("User","get");
```

```
qry.setAttribute("select","first_name,last_name,email");

var alias = new Item("Alias","get");
alias.setAttribute("select","related_id");

var identity = new Item("Identity","get");
identity.setAttribute("select","name");
identity.setProperty("name", "Larry Bird");

alias.setRelatedItem(identity);
qry.addRelationship(alias) ;

var results = qry.apply();
if (results.isError()) {
  top.aras.AlertError(results.getErrorDetail());
  return;
}
```

7.9 Recursive Query on a Related Item

You want to perform a recursive query on a related item.

Technique

Use the GetItemRepeatConfig action to perform the query. For more information, refer to the table in section 4.2 Built in Action Methods.

AML

```
<Item type="Part" action="GetItemRepeatConfig"
select="item_number,name,major_rev,has_change_pending,state" id="{@id}">
<Relationships>
   <Item type="Part BOM" select="sort_order,quantity,_kit_flag,related_id"
repeatProp="related_id" repeatTimes="6"/>
</Relationships>
</Item>
```

7.10 Add an Item configuration in one transaction

You want to add an Item configuration like a BOM as one transaction.

Technique

Adding an Item configuration is done by building the Item structure using the IOM methods.

JavaScript

```
var innovator = new Innovator();
var partItem = innovator.newItem("Part","add");
partItem.setProperty("item_number", "123-456");
partItem.setProperty("description", "Blah blah");

var bomItem = new Item("Part BOM","add");
bomItem.setProperty("quantity", "10");

var relatedItem = new Item("Part","get");
```

```
relatedItem.setProperty("item_number", "555-555");

bomItem.setRelatedItem(relatedItem);
partItem.addRelationship(bomItem) ;

var resultItem = partItem.apply();
if (resultItem.isError()) {
  top.aras.AlertError(resultItem.getErrorDetail());
  return;
}
```

Technique

The following is the same thing but uses the *Item*.loadAML() method to populate the Item Object with AML text.

JavaScript

```
var innovator = new Innovator();
var partItem = innovator.newItem();
partItem.loadAML(
  "<Item type='Part' action='add' >" +
    "<item_number>123-456</item_number>" +
    "<description>Blah blah</description>" +
    "<Relationships>" +
      "<Item type='Part BOM' action='add'>" +
        "<quantity>10</quantity>" +
        "<related_id>" +
          "<Item type='Part' action='get'>" +
            "<item_number>555-555</item_number>" +
          "</Item>" +
        "</related_id>" +
      "</Item>" +
    "</Relationships>" +
  "</Item>"
);

var resultItem = partItem.apply();
if (resultItem.isError()) {
  top.aras.AlertError (resultItem.getErrorDetail());
  return;
}
```

7.11 Add a Named Permission

You want to add a new named Permission Item.

Technique

Use the Item Class Extended Method set to add a new Named Permission Item.

JavaScript

```
var innovator = new Innovator();
var permItem = innovator.newItem("Permission","add");
permItem.setProperty("name", "AK Part Permissions");
```

```
setIdentityAccess(permItem, "All Employees", "get", true);
setIdentityAccess(permItem, "CM", "get", true);
setIdentityAccess(permItem, "CM", "update", true);
setIdentityAccess(permItem, "CM", "delete", true);
resultItem = permItem.apply();
if (resultItem.isError()) {
  top.aras.AlertError(resultItem.getErrorDetail());
  return;
}

function setIdentityAccess(item, identityName, permType, accessState)
{
  var identity = item.newItem();
  identity.setType("Identity");
  identity.setAction("get");
  identity.setProperty("name", identityName);
  var access = item.newItem("Access","add");
  access.setProperty("can_"+permType, (accessState ? "1" : "0"));
  access.setRelatedItem(identity);
  item.addRelationship(access);
}
```

7.12 Set a Private Permission for an Item

You want to set a new private Permission for an Item.

Technique

Use the Item Class Extended Method set to set a new private Permission for an Item.

JavaScript

```
// Set up the Part query
var innovator = new Innovator;
var qryItem = innovator.newItem("Part", "get");
qryItem.setAttribute("select", "id,permission_id");
qryItem.setAttribute("expand", "1");
qryItem.setAttribute("levels", "1");
qryItem.setPropertyCondition("item_number", "like");
qryItem.setProperty("item_number", "123%");

// Run the query and check for errors
var resultItem = qryItem.apply();
if (resultItem.isError()) {
 top.aras.AlertError(resultItem.getErrorDetail());
 return;
}

// Iterate over the Items returned and add the private permissions for each.
var count = resultItem.getItemCount();
for (i=0; i<count; ++i) {
 var item = resultItem.getItemByIndex(i);
 var permItem = item.getPropertyItem("permission_id");

 // Remove existing permissions first
```

```
 var accesses = permItem.getRelationships("Access");
 for (i=0; i<accesses.getItemCount(); i++) {
   var access = accesses.getItemByIndex(i);
   access.setAction("delete");
 }

 permItem.setProperty("name", permItem.getID());
 setIdentityAccess(permItem, "Component Engineering", "get", true);
 setIdentityAccess(permItem, "CM", "get", true);
 setIdentityAccess(permItem, "CM", "update", true);

 // Grant access to the current user's alias identity
 var myAlias = innovator.newItem("Alias","get");
 myAlias.setProperty("source_id", inn.getUserID());
 myAlias = myAlias.apply();

 var aliasId = myAlias.getItemByIndex(0).getProperty("related_id");
 var aliasName =
innovator.getItemById("Identity",aliasId).getProperty("name");
 setIdentityAccess(permItem, aliasName, "get", true);

 item.setAction("edit");
 resultItem = item.apply();
 if (resultItem.isError()) {
top.aras.AlertError(resultItem.getErrorDetail()); }
}

function setIdentityAccess(item, identityName, permType, accessState)
{
  var identity = item.newItem();
  identity.setType("Identity");
  identity.setAction("get");
  identity.setProperty("name", identityName);
  var access = item.newItem("Access","add");
  access.setProperty("can_"+permType, (accessState ? "1" : "0"));
  access.setRelatedItem(identity);
  item.addRelationship(access);
}
```

7.13 Apply a Generic Method

You want to write Generic Methods that can be used as subroutines for other Methods.

Technique

Use the `Innovator.applyMethod()` method to apply Generic Methods. The following examples assume a server-side method named "Reverse String" exists, and that it returns a result item containing the reversed contents of the <string> tag.

JavaScript

```
var inn = this.getInnovator();
var results = inn.applyMethod("Reverse String", "<string>abc</string>");
return results.getResult(); // returns "cba"
```

C#

```
Innovator inn = this.getInnovator();
Item results = inn.applyMethod("Reverse String", "<string>abc</string>");
// Return a result item with "cba" as the contents of the Result tag
return inn.newResult(results.getResult());
```

7.14 Need to Save text to a File

You want to save text to a file.

Technique

Use the File and StreamWriter namespaces to write to a text file.

C#

```
Innovator myInnovator = this.getInnovator();
string path = CCO.Server.MapPath("temp/yoyo.txt");
try
{
  if (File.Exists(path)) File.Delete(path);
  StreamWriter sw = File.CreateText(path);
  sw.Write(this.dom.InnerXml);
  sw.Close();
}
catch (Exception e)
{
  return myInnovator.newError(e.Message);
}
return myInnovator.newResult("ok");
```

7.15 Need to Send Email from a Method

You want to send an Email message from either a server or client side Method.

Technique

Use *Aras.IOM.Item.email(mail_item, idnt_item)* method to send email to a particular Innovator's identity.

Note: Aras Innovator's identity could be a group of people in which case the email is sent to all of them.

C#

```
...
// It's assumed here that required identity (item of type 'Identity')
// has already obtained (see other examples on how to perform 'get'
// requests to Innovator server using IOM). Same about item of type
// 'User' that represents the person who sends the email ('fromUser').
Item idnt = ...
Item fromUser = ...

// It's assumed in the sample that this represents an item of
// type Part. ${Item/item_number} in the email message is a parameter
// that represents the XPath to the property item_number of
```

```
// item of type Part; this parameter will be substituted on
// this.item_number before the email is sent. Note that both
// subject and body of the email item could be parameterized.
// This mechanism allows to created parameterized email templates
// (items of type "Email Message") that could be saved
// in Innovator and used for sending emails with concrete content
// when required.
string subject = "Part promotion notification";
string body = @"The part ${Item/item_number} has been promoted";

// In this particular example instead of getting a ready template
// email from the server a new item of type "EMail Message" is created
Item email_msg = this.newItem("EMail Message");

email_msg.setProperty("subject", subject);
email_msg.setProperty("body_plain", body);
email_msg.setPropertyItem("from_user", fromUser);

// Finally send the email
if( this.email( email_msg, idnt ) == false )
{
  // Error handling
  ...
}
```

7.16 Need a callback for a Relationships Grid Row Event

You want to call a client side Method when the Relationships Grid row is selected to deselect the row if it is not a new relationship row.

Technique

Add the Grid Event relationship to a Method as the callback for the OnSelectRow event.

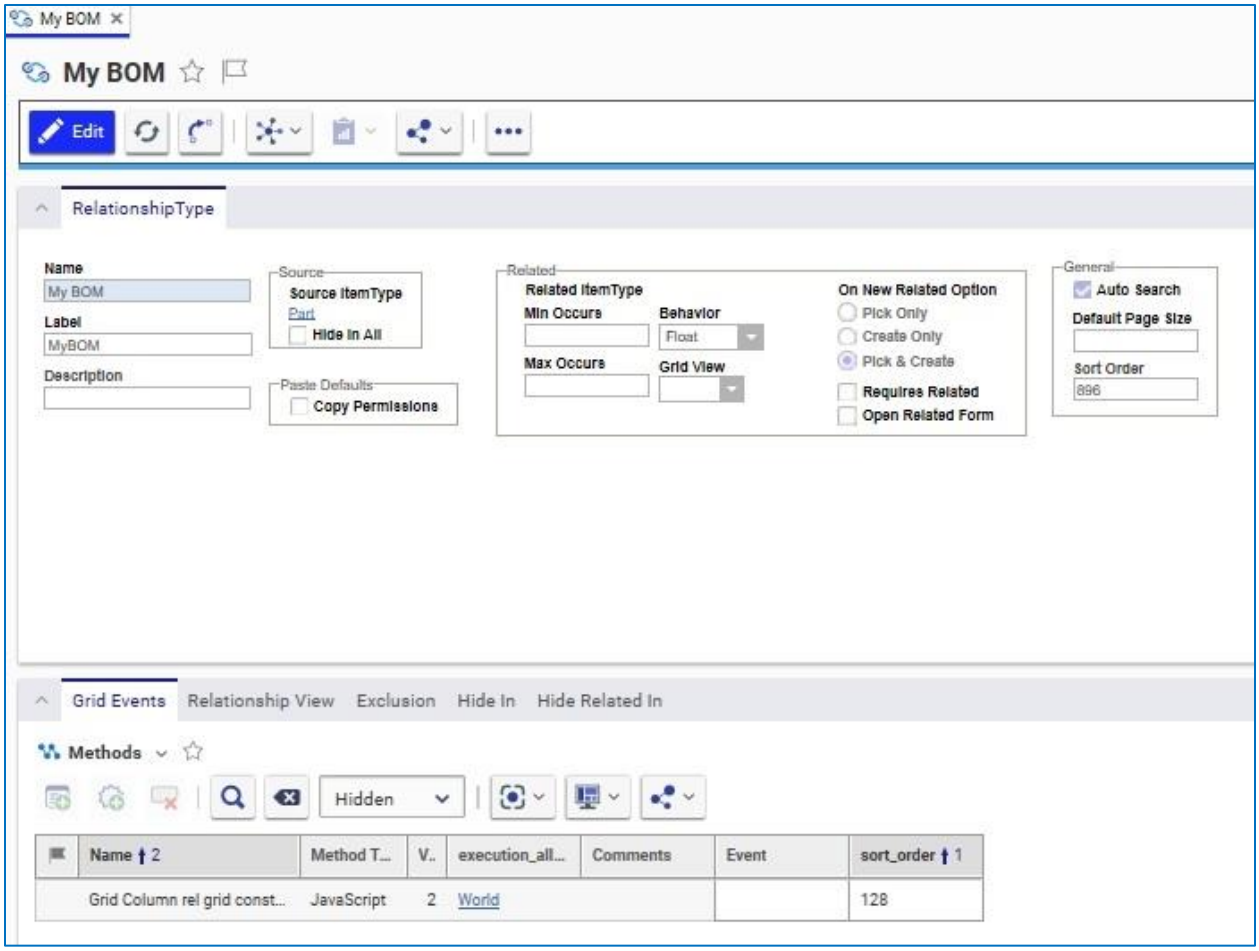


Figure 7.

The Method gets three arguments: relationshipID, relatedID and gridApplet. The relationshipID is the ID for the relationship Item for the selected row. The relatedID is the ID for the related Item for the selected row. The gridApplet is a handle to the grid control object.

The relationshipID is also the ID for the grid control row. This recipe calls the gridApplet Deselect() method if the relationship Item for the selected row has not been modified.

JavaScript

```
// The row ID is the same as the relationship ID
var rowId = gridApplet.getSelectedId();

// Find the relationship item and exit if it's not found
var thisItem = parent.thisItem;
var xpath = "Relationships/Item[@id='" + rowId + "']";
var relItem = thisItem.getItemsByXPath(xpath);
if (relItem.getItemCount() == 1) {
 relItem = relItem.getItemByIndex(0);
} else {
 return;
}
```

```
// Check the isDirty attribute to see if the relationship has been modified
var isDirty = (relItem.getAttribute("isDirty") == "1");
if (!isDirty) { gridApplet.Deselect(); }
```

7.17 Need a callback for a Relationships Grid Cell Event

You want to call a client side Method when the Relationships Grid cell is selected to blur the cell and prevent editing the cell value.

Technique

Add the Event relationship to a Property as the callback for the OnEditCell event.

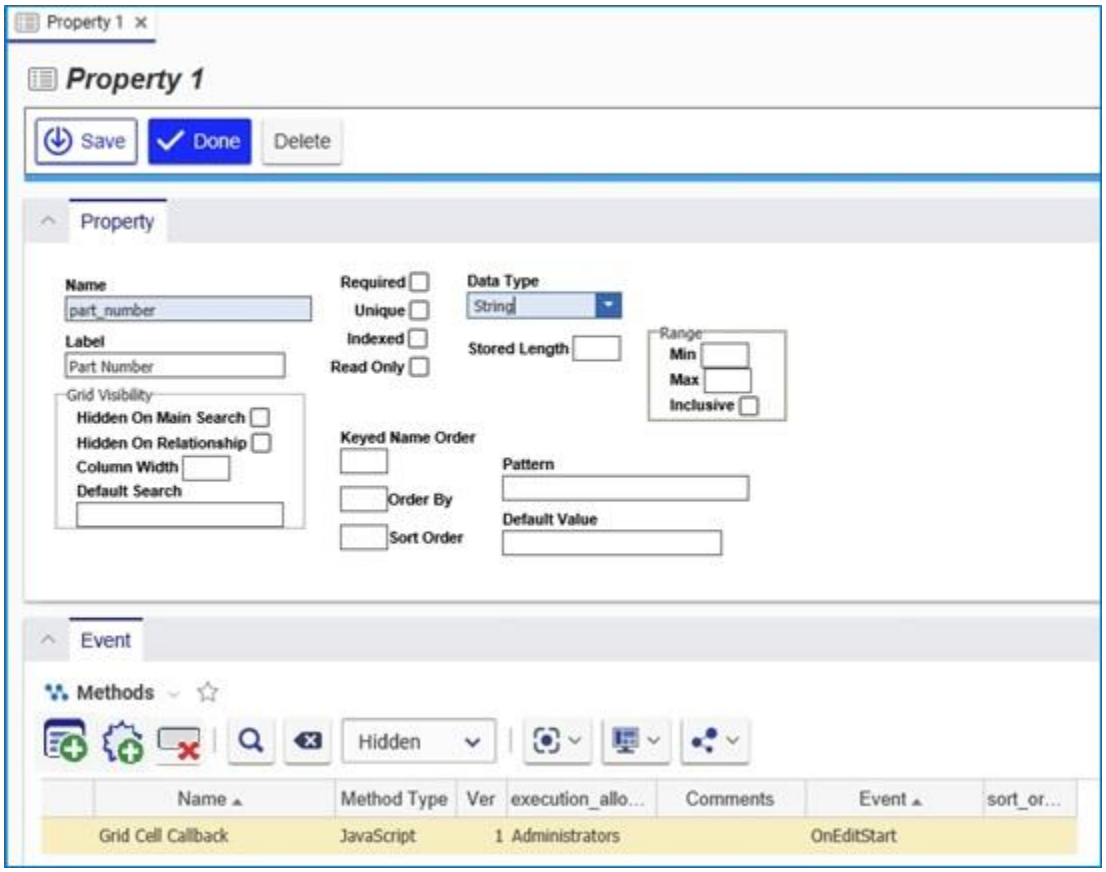


Figure 8.

The Method gets five arguments: relationshipID, relatedID, propertyName, colNumber, gridApplet

The propertyName is the name of the Property for the cell column selected, and colNumber is the column position number in the grid.

Simple return false and this blurs the grid cell.

JavaScript

```
// Get the current value of the cell
var cellValue = gridApplet.GetCellValue(relationshipID,colNumber);
```

```
// If the cell already has a value, disallow editing
if (cellValue !== "") {
 return false;
}
```

7.18 Show relationships in a Grid control on the Form

You want to show the relationships for the context Item in a Grid control on the Item Form.

Technique

Add an HTML Field (positioned at Point (300,10) in the code sample below) and insert an HTML code than defines the tag to hold the dynamically populated grid and the JavaScript to get the populating grid relationships.

HTML Field code

```
<div id="gridTD" style="width: 400px; height: 500px;"></div>
<script type="text/javascript">
  var myCount = 0;
  var gridControl;

clientControlsFactory.createControl("Aras.Client.Controls.Public.GridContaine
r", {id: "grid", connectId: "gridTD", canEdit_Experimental: function () {
return false; }}, function(control){
        gridControl = control;
        clientControlsFactory.on(gridControl, {
          "gridClick": function (rowID, column) {
            alert("rowId:" + rowID + ", col:" + column);
          }
        });
        fillGrid();
  });
  function fillGrid() {
    var item = document.thisItem;
    // Get the relationships
    var qry = item.newItem("Part BOM", "get");
    qry.setAttribute("select", "quantity,related_id(item_number,name,cost)");
    qry.setProperty("source_id", item.getID());
    var results = qry.apply();
    if (results.getItemCount() < 0) {
      top.aras.AlertError(results.getErrorDetail());
      return;
    }
    // Populate the grid with the results.
    populateGrid(item, results);
  }
  function populateGrid(item, results) {
    var propNameArr = new Array("item_number", "name", "cost");
    var gridXml =
    "<table editable='false' draw_grid='true'>" +
    "<columns>" +
    "<column width='30%' order='0' align='left' />" +
    "<column width='40%' order='1' align='left' />" +
    "<column width='15%' order='2' align='right' />" +
    "<column width='15%' order='3' align='right' />" +
```

```
    "</columns>" +
    "<thead>" +
    "<th>Part Number</th>" +
    "<th>Name</th>" +
    "<th>Cost</th>" +
    "<th>Quantity</th>" +
    "</thead>" +
    "</table>";
    var inn = item.getInnovator();
    var gridDom = inn.newXMLDocument();
    gridDom.loadXML(gridXml);
    var tableNd = gridDom.selectSingleNode("/table");
    var c = results.getItemCount();
    for (var i=0; i<c; ++i) {
      var bom = results.getItemByIndex(i);
      var part = bom.getRelatedItem();
      var trNd = gridDom.createElement("tr");
      trNd.setAttribute("id", bom.getID());
      var tdNd;
      for (var j=0; j<propNameArr.length; j++) {
        tdNd = gridDom.createElement("td");
        tdNd.text = part.getProperty(propNameArr[j]);+
        trNd.appendChild(tdNd);
      }
      tdNd = gridDom.createElement("td");
      tdNd.text = bom.getProperty("quantity");
      trNd.appendChild(tdNd);
      tableNd.appendChild(trNd);
    }
    gridControl.InitXML(gridDom.xml);
  } </script>
```

7.19 Want the Identities for the User

You want to get the Identity ID's for the User.

Technique

Use the classic Aras Innovator API top.aras.getIdentityList() method, which returns a comma delimited string of ID's.

Note: The classic API will eventually be eliminated and this method will become available on the IOM Innovator object as `Innovator.getIdentityList()`.

JavaScript

```
// This will get an array of identity IDs from the client cache
var myIdentityIDs = top.aras.getIdentityList().split(',');
```

7.20 Want a field to be either a sequence or user entered value

You want to enable a field on the form to be a sequence value or allow the user to enter a value.

Technique

Use the classic Aras Innovator API top.aras.getNetSequence() method, which returns the next sequence value from the server.

Note: The classic API will eventually be eliminated and this method will become available on the IOM Innovator object as `Innovator.getNetSequence()`.

Using the Form Tool we add an HTML Field to the Form, which we use to insert the HTML and JavaScript code for to provide the button to get the next sequence value and update the Field value and client cache.

HTML Field code

```
<a href="javascript:getNextNumber();">
  <img src="../images/Sequence.svg" border="0" style="max-width: 20px; max-
height: 20px;">
</a>
<script>
 function getNextNumber() {
   // Get the next sequence value.
  var seq = top.aras.getNextSequence("","Default Part");

  // This will update the client cache with the new value.
  handleItemChange("item_number", seq);
 }
</script>
```

7.21 Want to Vault a File

You want to save a CAD Document with an attached Native File.

Technique

You will need to use the `setFileProperty` call which handles creating a file and sets the associated value on the specified property.

Client-side methods use `selectFile()` that gets a File object based on user selection:

Javascript

```
var vlt = top.aras.vault;
vlt.selectFile().then(function (fileObject)
{
   var d = aras.IomInnovator.newItem("CAD", "add");
   d.setProperty("item_number", "007");
   d.setFileProperty("native_file", fileObject);
   d.apply();
});
```

Server-side methods require a string for the physical path to the file. The file must be on the server machine.

C#

```
Item d = this.newItem("CAD", "add");
d.setProperty("item_number", "007");
d.setFileProperty("native_file", @"C:\myFile.txt");
return d.apply();
```

7.22 Want to get an existing Vaulted File and save it with a new Document

You want to get an existing File from the Vault and attach it to a new Document

Technique

This is similar to the last recipe, but uses the `getItemByKeyedName()` method to get an existing File Item and `copyAsNew()` to create it as a new File Item.

JavaScript

```
// Create the Document item
var docItem = this.newItem("Document","add");
docItem.setProperty("item_number","456");

// Get the File item
var innovator = this.getInnovator();
var fileItem = innovator.getItemByKeyedName("File","My Document.doc");
if (fileItem.isError()) {
 top.aras.AlertError(fileItem.getErrorDetail());
 return;
}
// Duplicate File Item as files should be 1 to 1
var newFile = fileItem.apply("copyAsNew");
// Create the relationship between the Document and File
var relItem = this.newItem("Document File","add");
docItem.addRelationship(relItem);
relItem.setRelatedItem(newFile);

var results = docItem.apply();
if (results.isError()) {
 top.aras.AlertError(results.getErrorDetail());
} else {
 // Show the new Document
 top.aras.uiShowItemEx(results.getItemByIndex(0).node, 'tab view', true);
}
```

7.23 Need to reject an Item Promote

You want to reject an Item Promote if a value of Property is in valid.

Technique

Use the Pre Server Method on the Life Cycle Transition to call a server side Method to validate the Item before it is promoted and if invalid rejects the Promote by returning an Error Item.

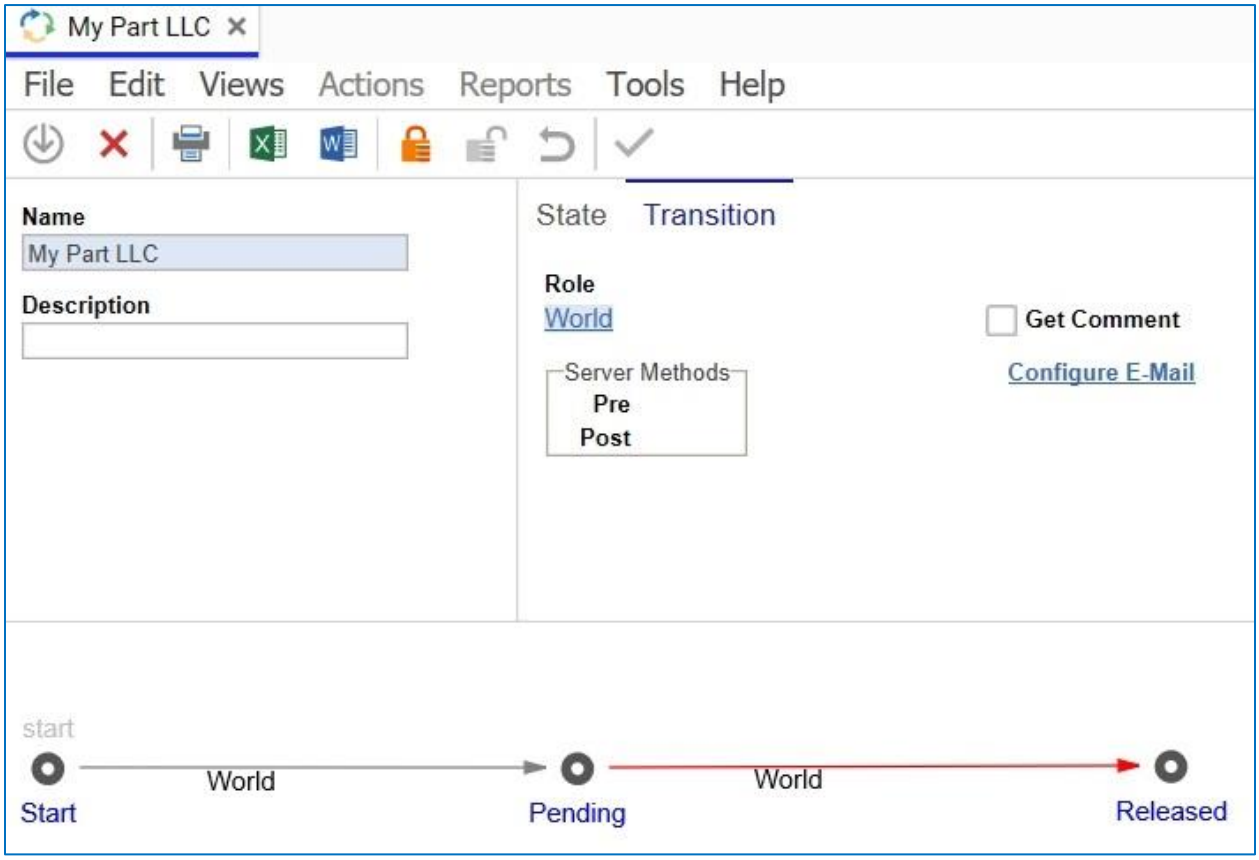


Figure 9.

C#

```
Innovator innovator = this.getInnovator();
if (Convert.ToDecimal(this.getProperty("cost")) > 500) {
 Item error = innovator.newError("Error promoting: Item costs more than
$500.00");
 return error;
}
return this;
```

7.24 How to handle multilingual properties

You want to programmatically get/set multilingual string properties

Technique

Use *Item*.setAttribute("language","*") to get all language values and *Item*.setProperty(myProperty,value,lang) to set specific language values.

JavaScript

```
var inn = this.getInnovator();

// Add a new List with multilingual Value labels
var listItem = this.newItem("List","add");
listItem.setProperty("name","Numbers");
var valueItem = listItem.createRelationship("Value","add");
valueItem.setProperty("value","1");
valueItem.setProperty("label","One","en");
valueItem.setProperty("label","Ein","de");
var valueItem2 = listItem.createRelationship("Value","add");
valueItem2.setProperty("value","2");
valueItem2.setProperty("label","Two","en");
valueItem2.setProperty("label","Zwei","de");
var resultItem = listItem.apply();
if (resultItem.isError()) {
 return inn.newError("Error adding List: " + resultItem.getErrorDetail());
}

// Retrieve the List with labels in both English and German
listItem = this.newItem("List","get");
listItem.setProperty("name","Numbers");
valueItem = listItem.createRelationship("Value","get");
valueItem.setAttribute("language","en,de");
resultItem = listItem.apply();
if (resultItem.isError()) {
 return inn.newError("Error retrieving List: " +
resultItem.getErrorDetail());
}
```

7.25 How to handle date properties

You want to programmatically get/set date properties

Technique

Convert date values to "yyyy-MM-ddThh:mm:ss" format before setting the property

JavaScript

```
// Get yesterday's date
var myDate = new Date();
myDate.setDate(myDate.getDate()-1);

// Find all methods edited in the past 24 hours
```

```
var myItem = this.newItem("Method","get");
myItem.setAttribute("select","name");
myItem.setProperty("modified_on",dateFormat(myDate));
myItem.setPropertyAttribute("modified_on","condition","gt");
myItem = myItem.apply();

// Loop through the returned methods and return the list
var methodList = new String("");
for (var i=0; i<myItem.getItemCount(); i++) {
 methodList += myItem.getItemByIndex(i).getProperty("name","") + ", ";
}
top.aras.AlertError("The following methods were modified in the past 24
hours: "+methodList.substr(0,methodList.length-2));

function dateFormat(d) {
 var dateString = d.getFullYear()+"-";
 dateString += pad(d.getMonth()+1)+"-";
 dateString += pad(d.getDate())+"T";
 dateString += pad(d.getHours())+":";
 dateString += pad(d.getMinutes())+":";
 dateString += pad(d.getSeconds());
 return dateString;
}

function pad(x) {
 return (x<10) ? "0"+x : ""+x;
}
```

C#

```
Innovator inn = this.getInnovator();

// Get yesterday's date
DateTime myDate = DateTime.Today.AddDays(-1);

// Find all methods edited in the past 24 hours
Item myItem = inn.newItem("Method","get");
myItem.setAttribute("select","name");
myItem.setProperty("modified_on",myDate.ToString("yyyy-MM-ddThh:mm:ss"));
myItem.setPropertyAttribute("modified_on","condition","gt");
myItem = myItem.apply();

// Loop through the returned methods and return the list
string methodList = "";
int myItemCount = myItem.getItemCount();
for(int i = 0; i < myItemCount; i++){
 methodList += myItem.getItemByIndex(i).getProperty("name","") + ", ";
}

return inn.newError("The following methods were modified In the past 24
hours: " + methodList);
```

7.26 How to pass values from onBefore*X* to onAfter*X* events

You want to pass some value from an onBefore (i.e. onBeforeUpdate) event to an onAfter (i.e. onAfterUpdate) event for data process handling

Technique

Use the built in function to add a variable, read a variable and remove a variable from the RequestState. This sample determines if there was a change made to the "Name" property of Part.

C#

OnBeforeUpdate event

```
//Get Part Name before change

Innovator inn = this.getInnovator();
Item myPart = inn.newItem("Part","get");
myPart.setID(this.getID());
myPart.setAttribute("select","name");
myPart=myPart.apply();
string prevName = myPart.getProperty("name");

RequestState.Add("prevName", prevName); //Add value to SessionState
return this;
```

OnAfterUpdate event

```
string prevName = (string)RequestState["prevName"];
string curName = this.getProperty("name");

if (prevName != curName)
{
   //Do some logic here
}

//Perform cleanup of RequestState
RequestState.Remove("prevName");  //removes single key
// to remove all keys use RequestState.Clear();
return this;
```

7.27 How to reference custom DLL from server method

You want to reference a custom library from inside an Aras Innovator Server Method

Technique

Create a custom DLL that references the IOM. This DLL is then added to the BIN folder and referenced in the method-config.xml, allowing it to be called from a server side method.

Create the custom DLL

To create the custom DLL, it is necessary to create a Visual Studio C# class library project using .NET 4. The project needs to reference the IOM of the version of Aras Innovator you are connecting to.

Class Code

```
namespace CookBookCustomDLL
{
    public class CustomDLLFunct
    {
        public static string returnUser(Innovator inn)
        {
            string userID = inn.getUserID();
            return userID;
        }
    }
}
```

Code tree setup with new DLL

Once you have created the DLL and built the assembly, you need to use the steps below to add the DLL to your Aras Innovator code tree

1. Copy the CookBookCustomDLL.dll and CookBookCustomDLL.pdb into the \Innovator\Server\bin folder
2. Open for Edit the \Innovator\Server\Method-Config.xml and add the highlighted line

```
...
<ReferencedAssemblies>
      <name>System.dll</name>
      <name>System.XML.dll</name>
      <name>System.Web.dll</name>
      <name>System.Data.dll</name>
      <name>System.Core.dll</name>
      <name>System.Configuration.dll</name>
      <name>$(binpath)/IOM.dll</name>
      <name>$(binpath)/InnovatorCore.dll</name>
      <name>$(binpath)/CoreCS.dll</name>
      <name>$(binpath)/SPConnector.dll</name>
      <name>$(binpath)/ConversionManager.dll</name>
         <name>$(binpath)/CookBookCustomDLL.dll</name>
...
```

3. Search for the `<Template>` tag for the language you are using and include the additional namespace you need by adding additional "using" lines.
   a. When adding lines, ensure you update the `line_number_offset` for accurate debugging messages.

```
   <Template name="CSharp" line_number_offset="41">
      <![CDATA[
using System;
…
using CookBookCustomDLL;
```

4. Save the Method-Config.xml

Call new DLL from Innovator Server method.

Code snippet that can be used as a reference for how to call your assembly and function

Code Snippet

```
Innovator inn = this.getInnovator();
```

```
string userID = CustomDLLFunct.returnUser(inn);
```

Assembly Redirection

If your custom DLL references the current version of the IOM, and you upgrade Aras Innovator to a newer version, your function fills to execute due to an assembly mismatch on the IOM. A quick workaround until the DLL can be rebuilt referencing the latest IOM is to add an assembly redirect.

To add an assembly redirect, you need to add the following into the <configuration> section of the \Innovator\Server\web.config file.

```
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="IOM"
                        publicKeyToken="524d880b05474146"
                        culture="neutral" />
      <bindingRedirect oldVersion="9.4.0.5804" newVersion="9.4.0.5815" />
    </dependentAssembly>
  </assemblyBinding>
</runtime>
```

The oldVersion, highlighted in yellow above, is the original version of the IOM.dll referenced by the custom DLL. The newVersion, highlighted in red text above, is the version of the IOM.dll that is located in the upgraded instance of Aras Innovator.

7.28 How to add sub menus to context menu in search grid

There is a possibility of adding sub menus to the context menu in the search grid of Aras Innovator menu items.

`Aras.Client.Controls.ContextMenu` control supports submenus. It provides two methods for adding menu items:

1. `add(id, label, parentMenuId, args);`
2. `addRange(items, parentMenuId);`

Below are the examples given for `itemsGrid.html`:

Example #1

```
var menu = grid.getMenu();
menu.add("item_0", "Item 0");
menu.add("item_0.0", "Item 0.0", "item_0", { onClick: function () { alert(0);
} });
menu.add("item_0.1", "Item 0.1", "item_0", { onClick: function () { alert(1);
} });
menu.add("item_0.2", "Item 0.2", "item_0", { onClick: function () { alert(2);
} });
```

Example #2

```
var menu = grid.getMenu(),
       subMenus = [
              {
                     id: "item_0.0",
```

```
                        name: "Item 0.0",
                        onClick: function () { alert(0); }
                },
                {
                        id: "item_0.1",
                        name: "Item 0.1",
                        onClick: function () { alert(1); }
                },
                {
                        id: "item_0.2",
                        name: "Item 0.2",
                        onClick: function () { alert(2); }
                }
        ];
menu.add("item_0", "Item 0");
menu.addRange(subMenus, "item_0");
```

7.29 How to Create a Dynamic List

Use the following code to create and populate a dynamic list:

```
var listDropdown = getFieldComponentById("{ID}");

//var listDropdown = getFieldComponentByName("{NAME}");

//you can use either getFieldComponentByName or getFieldComponentById to find the list

function addOption(selectbox, txt, val){

    var options = selectbox.component.state.list


    options.push({label: txt, value: val})

    selectbox.component.setState({list: options});

}


function removeOption(selectbox, val) {

    var options = selectbox.component.state.list


    for (var i = 0; i < options.length; i++) {

        if (options[i].value == val) {

            options.splice(i, 1)

            break;

        }
```

```
    }


    selectbox.component.setState({
        list: options
    });
}
```

You can add and remove options using the following:

```
addOption(listDropdown, "option 1", "opt-1");removeOption(listDropdown, "opt-1");
```

7.30 How to Download a File from Aras Innovator to a Client Machine

The purpose of aras.downloadFile is to download a file from Aras Innovator to a client machine. This function is a Javascript function and can only be called from a client-side Method.

The function takes in up to two properties:

- fileNd – an XML Node of the File to be downloaded. It contains the filename, ID, and all pertinent properties.
- preferredName – (Optional) a string to replace the file's name during download.

```
let item = aras.newIOMItem('File', 'get');
...
item = item.apply();

aras.downloadFile(item.node, 'sample.txt');
```

7.31 How to Convert Strings

The ArasModules.xml.parseString function takes a string and converts it into an XmlDOMDocument.

The function takes the following argument:

- itemND – a string that conforms to a valid XML structure.

```
...
let document = ArasModules.xml.parseString('<Item action="get"
type="File"><filename>sample.txt</filename></Item>');
...
```

7.32 How to Load an XML File URL

The ArasModules.xml.parseFile function loads the URL of an XML file into a local XmlDOMDocument object.

The function takes the following argument:

- url – a valid URL for an XML file.

```
...
let document =
ArasModules.xml.parseFile('http://sampleserver/samplexml.xml');
...
```

7.33 How to Create a List of Nodes

ArasModules.xml.selectNodes selects a list of nodes that match the query. It functions identically to the standard XmlDOMDocument.selectNodes(xPathString).

The function takes the following arguments:

- xmlDocument – a valid XML document.
- nodeName – the name of a node in the xmlDocument.

```
...
let xmlNodes = ArasModules.xml.selectNodes(xmlDoc, 'sample');
...
```

7.34 How to Select a Single Node

ArasModules.xml.selectSingleNode selects the first XML node that matches the XPath expression. It functions identically to the standard XmlDOMDocument.selectSingleNode(xPathString).

The function takes the following arguments:

- xmlDocument – a valid xmlDocument.
- nodeName – The name of a node in the xmlDocument.

```
...
let xmlSingleNode = ArasModules.xml.selectSingleNode(xmlDoc, "created_on");
...
```

7.35 How to Transform a Node

ArasModules.xml.transform processes the document node using the specified XSL stylesheet. It functions identically to the standard XmlDOMDocument.transformNode(objXSLTStylesheet).

The function takes the following arguments:

- xmlDocument – a valid XmlDOMDocument.
- xsltStylesheet – a valid XSLT stylesheet that is applied to the xmlDocument.

```
...
let transformedXmlDocument = ArasModules.xml.transform(xmlDoc,
xsltStylesheet);
...
```

7.36 How to Create a NodeType

ArasModules.xml.createNode creates a NodeType. It functions identically to the standard XmlDOMDocument.createNode(Type, name, namespaceURL).

The function takes the following arguments:

- xmlDocument – a valid XmlDOMDocument.
- Type – The IXMLDOMNodeType that is being applied. For example, 1 is a NODE_ELEMENT.
- Name – The name of the NodeType being created.
- url – A string defining the namespace URL.

```
...
let newNode = ArasModules.xml.createNode(xmlDoc, 1, 'sample', '');
...
```

7.37 How to Get Text Associated with a Node

ArasModules.xml.getText retrieves the text from an XmlDOMDocument object node.

The function takes the following argument:

- xmlNode – A valid XmlDocument object's node.

```
...
let nodeValue = ArasModules.xml.getText(xmlNode);
...
```

7.38 How to Set Text For a Node

ArasModules.xml.setText sets the text for an XmlDOMDocument object node.

The function takes the following arguments:

- xmlNode – a validXmlDOMDocument object's node.
- nodeValue – The value to be inserted into the node.

```
...
ArasModules.xml.setText(xmlNode, 'sample');
...
```

7.39 How to Convert an Object into a String

ArasModules.xml.getXml converts an XmlDOMDocument object into a string.

The function takes the following argument:

- xmlDocument – a valid XmlDocument.

```
...
let xmlString = ArasModules.xml.getXml(xmlDoc);
```

```
...
```

7.40 How to Return an Error

ArasModules.xml.getError returns an error when given an XMLDOMDocument object.

The function takes the following argument:

- xmlDocument – A valid XmlDocument.

```
...
   var error = ArasModules.xml.getError(xmlDoc);

   if (error.errorCode !== 0) {
      return aras.getResource('', 'tz.parse_update_fail_details',
error.reason);
   }
...
```

8 Aras Innovator Solution Studio

The Aras Innovator Solution Studio is the user interface you use to author the code for your Innovator Methods. It is automatically launched when you open a Method Item in Aras Innovator client.

8.1 Features

The Solution Studio offers several useful features to help aid you in the development of your Methods.

- A robust text editor with search/replace and undo/redo capabilities.
- Color coding the keywords for the language you are developing in.
- An IOM API code guide.
- An online help interface that is context sensitive for the methods selected in the code guide.
- Syntax checking based on the language you are developing in.

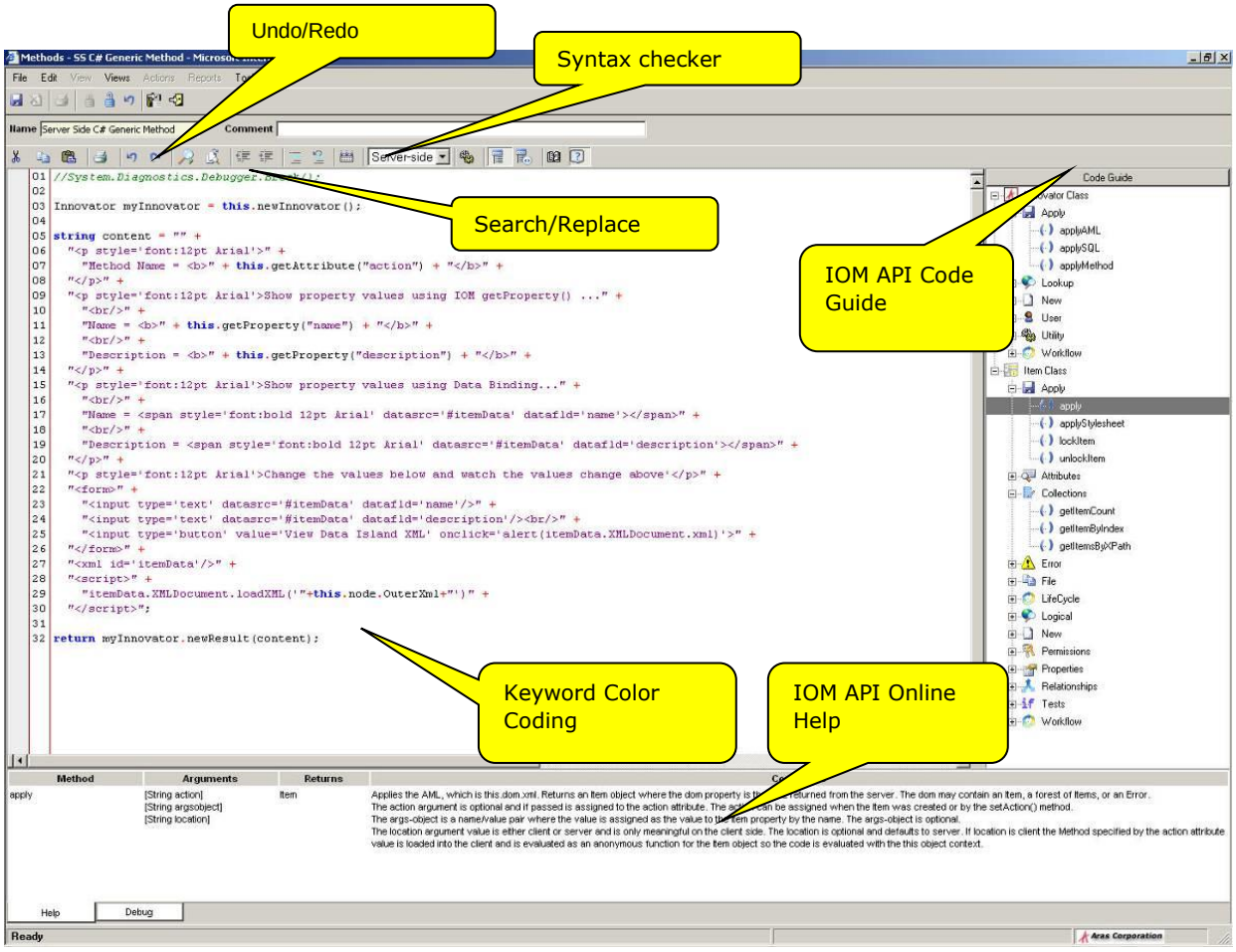


Figure 10.

9 Debugging

Aras Innovator has the following features for debugging/logging your Methods:

- Visual Studio to debug Methods either client or server.
- Logging of debug messages to an XML formatted log file.

9.1 Enabling Debugging in Aras Innovator

Server method debugging is disabled by default in Aras Innovator 12.0. This setting saves disk space as the temporary dlls used for debugging the methods do not need to be created.

To enable the debugging of Server Methods, you can add the following line to your "InnovatorServerConfig.xml", located in the root folder of your installation.

```
<operating_parameter key="DebugServerMethod" value="true" />
```

Server method debugging can then be disabled again by changing the value of this line to false.

After changing this value, you should restart IIS to confirm the change is applied.

9.2 Setting up VS.NET to debug Server side Methods

1. Open "Microsoft Visual Studio .NET" debugger.
2. Choose **Tools --> Attach to Processes…**.

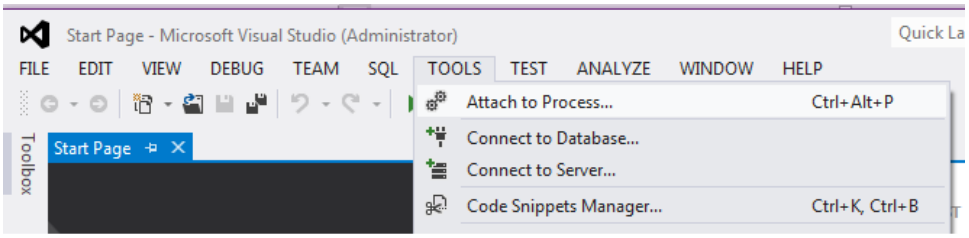


Figure 11.

3. Choose `w3wp.exe` and click **'Attach'**.

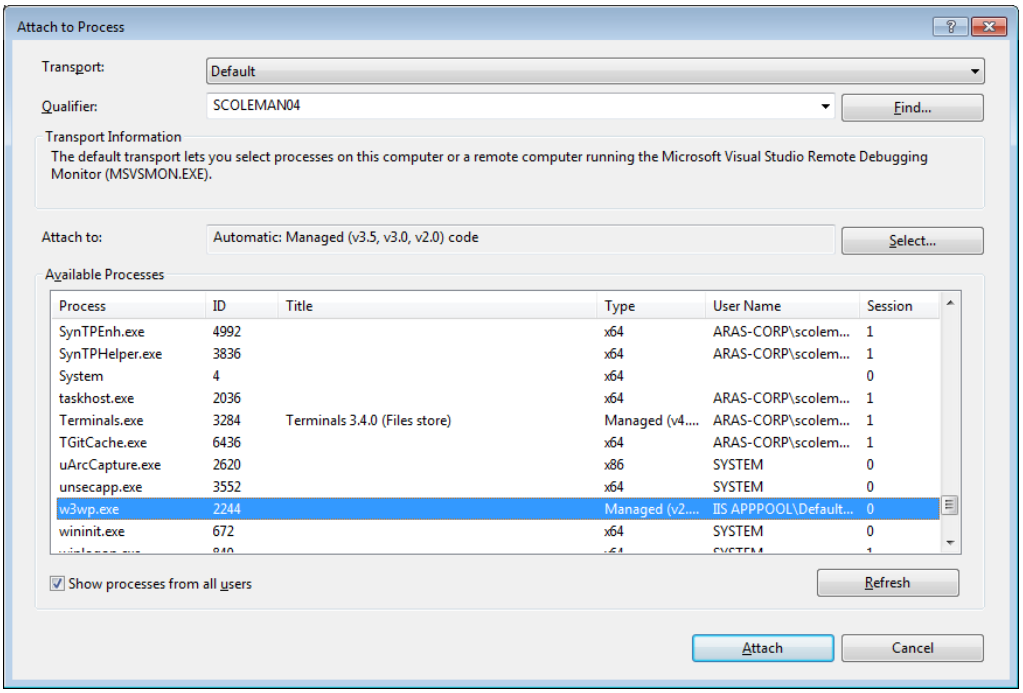


Figure 12.

4. Choose **Debug --> Exceptions** to force the method into the debugger when an exception occurs and choose the check boxes "Thrown".

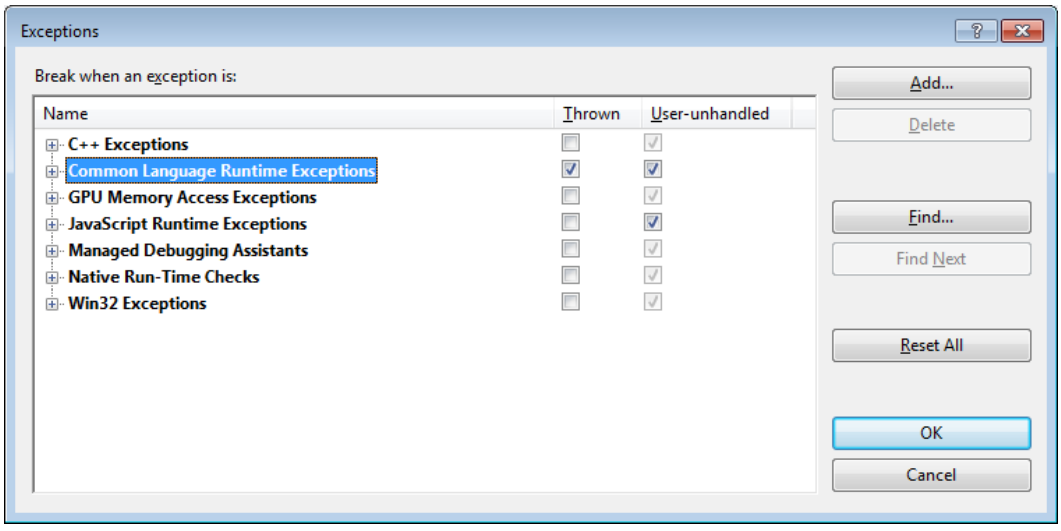


Figure 13.

5. Optionally, you can include the line `System.Diagnostics.Debugger.Break()` to force a breakpoint at a specific line in your method.

9.3 Debugging Client side Methods

1. Include the line **`debugger;`** to your Method.
2. Press the F12 key while in your Internet Browser, or right-click and select 'Inspect', or 'Inspect Element'. This will open a new dialog.

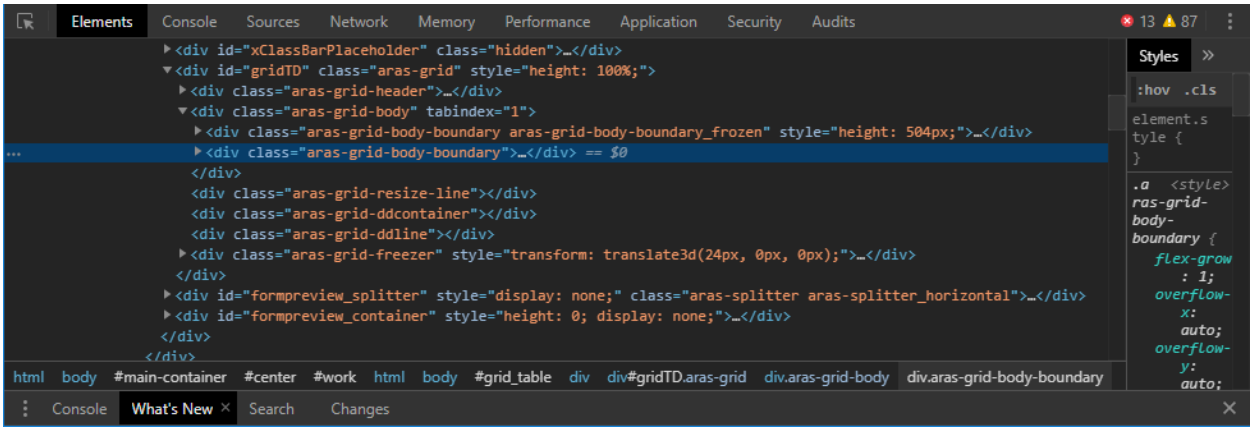


Figure 14.

3. Perform the action that runs the method code, and the code should pause itself once it reaches the **`debugger`**; statement.

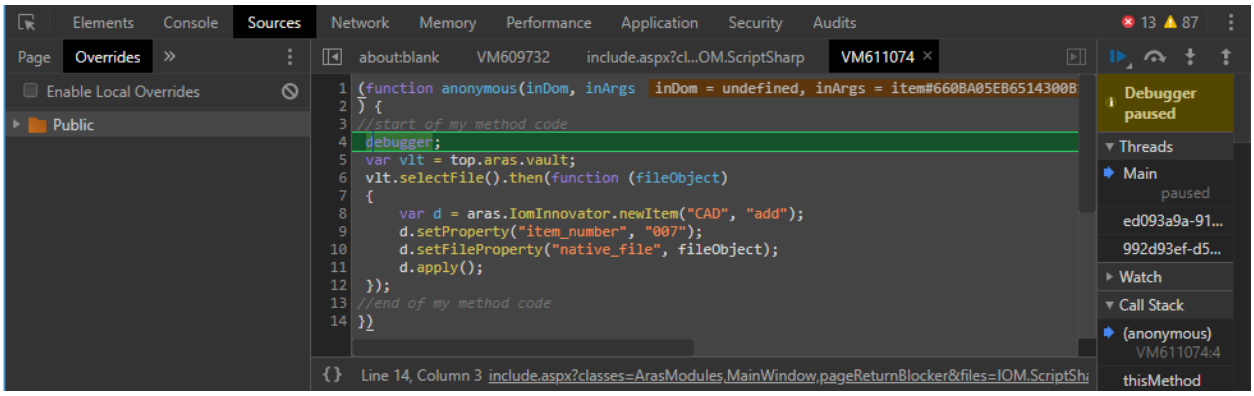


Figure 15.

4. Afterwards, you can press F8 to continue running normally, F10 to perform another step of method code, and find javascript variable values using the Debugger Console.

9.4 Setting up the server side logging

Add the following lines to the InnovatorServerConfig61.xml file:

```
<operating_parameter key="debug_log_flag" value="true" />
<operating_parameter key="debug_log_prefix" value="C:/TEMP/DEBUG-" />
<operating_parameter key="debug_log_limit" value="1000" />
```

In order to write messages to the C:/TEMP/DEBUG-* log file, add the following line of code to the server side method:

```
CCO.Utilities.WriteDebug("logFileName", "logMessage");
```

10 External APIs

There are a few ways to connect to the Aras Innovator server from other applications. Aras provides two assemblies (.NET and COM-compatible) that implement the IOM API and can be used by other applications. Alternatively, simple SOAP communication can be used to connect to Aras Innovator.

10.1 .NET IOM

A .NET version of IOM.dll can be found in the `\Utilities\Aras Innovator <Version> IOM SDK\.NET` directory on the CD Image. You can copy this to another location and reference it using .NET project. IOM APIs belong to Aras.IOM namespace. First, every application must create a connection with the Aras Innovator server and login to the Aras Innovator server using the connection. If the login succeeds then an instance of the Innovator class must be created with the connection.

Here is a small sample:

```
…
using Aras.IOM;
…
String url = "https://myserver/MyInnovator/Server/InnovatorServer.aspx";
// database, username and password values should be entered by user
String db;
String userName;
String password;

HttpServerConnection conn =
   IomFactory.CreateHttpServerConnection( url, db, userName, password );
Item login_result = conn.Login();
if( login_result.isError() )
   throw new Exception("Login failed");
Innovator inn = IomFactory.CreateInnovator( conn );
…
```

Note: You can pass either encrypted (if required, use static method `Innovator.ScalcMD5(string pwd)` for encryption) or non-encrypted password (as shown in the previous example) to the method `IomFactory.CreateHttpServerConnection(…)`. If you pass a non-encrypted password it'll be encrypted inside the method. Otherwise the password is passed to the server without modifications.

It's highly recommended to logout of Aras Innovator prior to exiting the application:

```
…
conn.Logout();
…
```

10.1.1 Obtaining an Access Token

IOM API for authentication supports multiple authentication grants to obtain access token: a Resource Owner Password Credentials grant (further on Password grant), an Authorization Code grant and our custom Impersonate grant (a custom grant based on Client Credentials grant). You can use an obtained access token in subsequent requests to the Aras Innovator server.

Note: See more about OAuth 2.0 authorization grants in the OAuth 2.0 RFC specification: https://tools.ietf.org/html/rfc6749#section-4. Refer to the Impersonate grant chapter for more information.

The IOM API for authentication includes a public *ITokenProvider* interface and its default implementations: *PasswordTokenProvider, ImpersonateTokenProvider, AuthorizationFlowTokenProvider* and *WindowsTokenProvider*. If the default implementation does not cover your needs, it is possible to provide a custom implementation of the *ITokenProvider*.

Warning Before creating a token provider, it is necessary to get a *DiscoveryDocument* that contains information about OAuth server configuration.

10.1.1.1 Discovery of OAuth Server

The Discovery mechanism allows clients to find the OAuth server and request its configuration.

The IOM OAuth API includes a public *IDiscoveryDocumentProvider* interface and its default implementation of the *DiscoveryDocumentProvider*.

DiscoveryDocumentProvider does the following:

1. Sends a request to http://<company.net>/<Innovator>/Server/OAuthServerDiscovery.aspx to determine the OAuth server location. OAuth server URLs are specified in InnovatorServerConfig.xml and look like the following:

```
<OAuthServerDiscovery>
    <Urls>
        <Url value="http://company.net/Innovator/OAuthServer/" />
        <!-- Url value="http://company.local/Innovator/OAuthServer/" /-->
    </Urls>
</OAuthServerDiscovery>
```

2. Requests the OAuth server discovery configuration.

The *DiscoveryDocumentProvider* receives all OAuth server URLs that are configured in *InnovatorServerConfig.xml* and prioritizes the list of OAuth server URLs according to the following order:

1 Last accessible OAuth server URL if presented.

2 URLs with the same host as the Aras Innovator server host.

3 All other URLs received from *OAuthServerDiscovery.aspx*.

After receiving the OAuth server URLs, the *DiscoveryDocumentProvider* tries to find an accessible path by requesting */.well-known/openid-configuration* paths from the list according to priority. After an accessible path is found, *DiscoveryDocumentProvider* creates an instance of *DiscoveryDocument* based on the response from the OAuth server and saves the last accessible URL.

The DiscoveryDocument contains information about:

- OAuth server endpoints (e.g. authorization and token endpoints),
- Protocol type (standard or custom),
- Protocol version.

This information is used while creating token providers in cases where it is necessary to provide the *DiscoveryDocument* by itself or some data from it.

The following example shows how to use the *DiscoveryDocumentProvider*:

```
var discoveryDocumentProvider = new DiscoveryDocumentProvider(
    "https://myserver/MyInnovator/Server/InnovatorServer.aspx");
```

```
DiscoveryDocument discoveryDocument =
   await discoveryDocumentProvider.GetDiscoveryDocumentAsync();
```

Note: The GetDiscoveryDocumentAsync method can take CancellationToken as a parameter to provide the possibility of canceling a task.

The default DiscoveryDocumentProvider implementation does not use a cache for DiscoverDocument. If caching is necessary for your application, it is possible to implement a wrapper with the cache.

The *DiscoveryDocument* instance has the following properties:

- `Issuer` – token issuer identifier (OAuth server).
- `JwksUri` – JSON Web Key Set document URI.
- `AuthorizeEndpoint` – URL of authorization endpoint.
- `TokenEndpoint` – URL to token endpoint.
- `EndSessionEndpoint` – URL of end session endpoint.
- `ProtocolVersion` – version of protocol that is used by the OAuth server.
- `ProtocolInfo` – information about protocol that is used by the OAuth server.
- `ProtocolType` – type of protocol that is used by the OAuth server.
- `KeySetJson` – JSON web key set.

A Protocol is used to determine what authentication header is used by the OAuth server: the *Standard* protocol type uses the `"Authorization"` header and 401 HTTP status code for `"Unauthorized"`, the *Custom* protocol type uses `"X-Aras-Authorization"` and 498 HTTP status code for `"Unauthorized"`. The *Custom* protocol type is necessary in case the *Standard* protocol type conflicts with other protection mechanisms (e.g. when all server resources are protected by Windows authentication).

The *DiscoveryDocument* class also makes it possible to get any other property from a discovery document with the following methods:

- `TryGetString(string name)` – gets a string value from the discovery document by name. If the property is not found, it returns *null*.
- `TryGetBoolean(string name)` – gets a boolean value from the discovery document by name. If the property is not found or is not of the boolean type, it returns *false*.
- `TryGetStringArray(string name)` – gets a string array value from the discovery document by name. If the property is not found or is not of the string array type, it returns *null*.

Note: For more information about discovery see https://openid.net/specs/openid-connect-discovery-1_0.html.

10.1.1.2 Password Grant

The Password grant is useful in cases where the client is capable of obtaining the user credentials. Password grants are often used for legacy or migration reasons. It is recommended to use other grant types whenever possible.

Note: See more about Password grants in the OAuth 2.0 RFC specification: https://tools.ietf.org/html/rfc6749#section-4.3

You must use *the PasswordTokenProvider* to get an access token using a Password grant... You must pass *PasswordTokenProviderOptions* to create the *PasswordTokenProvider:*

```
// database, username and password values should be entered by user
String db;
String userName;
String password;

var tokenProvider = new PasswordTokenProvider(
   new PasswordTokenProviderOptions()
   {
      TokenEndpoint = discoveryDocument.TokenEndpoint,
      ClientId = "IOMApp",
      Scope = "Innovator",
      Database = db,
      UserName = userName,
      Password = password
   });
```

PasswordTokenProviderOptions use the following options:

- `TokenEndpoint` – URL of OAuth server token endpoint.
- `ClientId` – ID of OAuth client that requests token.
- `Scope` – scope that will be used to request token. Currently all Aras Innovator resources require *Innovator* scope.
- `Database` – name of database to login to.
- `UserName` – user name.
- `Password` – user password.

Note: A password can be passed either as plain text or MD5/SHA256 hash (depending on FIPS mode).

To get an access token it is necessary to call the *GetAccesTokenAsync* method. This method sends a request to the OAuth server token endpoint and returns the access token created by a Password grant.

```
string accessToken = await tokenProvider.GetAccessTokenAsync();
```

Note: The GetAccessTokenAsync method can take CancellationToken as a parameter to provide the possibility of cancelling a task.

10.1.1.3 Impersonate Grant

The Client Credentials grant is usually used for communication between servers.. Communication between Aras Innovator servers requires user information. The token requested by the Client Credentials grant does not contain user information, so the Impersonate grant has been implemented.

Note: Learn more about the Client Credentials grant in the OAuth 2.0 RFC specification: https://tools.ietf.org/html/rfc6749#section-4.4.

The Impersonate grant is a custom grant similar to the Client Credentials grant. The Impersonate grant is designed to authenticate client by client assertion tokens and provide access tokens by username and database. To request an access token using the Impersonate grant it is necessary to send the client assertion token, scope, database, and username to the token endpoint of the OAuth server.

1. Before you can use the Impersonate Grant with custom applications, you must do the following: Use OpenSSL to generate the certificates pair (PFX, CER files).
2. Put the public certificate (CER file) in the *{Installation_Dir}**OAuthServer**App_Data**Certificates*\ directory.
3. Use a private certificate (PFX file) in the custom application.
4. Configure the new OAuth client based on the InnovatorServer *clientRegistry* node using the allowed impersonate grant located in the *{Installation_Dir}**OAuthServer**OAuth.config* ***directory*.**

Use the configured OAuth client for authentication with the Impersonate grant.

Note: See more information about assertion tokens in JWT specification: https://tools.ietf.org/html/rfc7523

Warning The Impersonate grant is used by the trusted server application.

IClientAssertionProvider is used to provide an assertion token for the ImpersonateTokenProvider. The default implementation of the IClientAssertionProvider is JwtBearerClientAssertionProvider. To create the JwtBearerClientAssertionProvider it is necessary to provide JwtBearerClientAssertionProviderOptions.

```
var certificate = new X509Certificate2(
   privateCertificateFilePath,
   certificatePassword,
   keyStorageFlags);
var assertionProvider = new JwtBearerClientAssertionProvider(
   new JwtBearerClientAssertionProviderOptions
   {
      ClientId = "ClientId",
      Audience = discoveryDocument.Issuer,
      Certificate = certificate,
      AssertionTokenLifetime = TimeSpan.FromSeconds(300)
   });
```

Note: You can load the Certificate in *X509Certificate2* from Certificate Store. For more information see the other contructors of *X509Certificate2* https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.x509certificates.x509certificate2.-ctor

JwtBearerClientAssertionProviderOptions have the following options:

- `ClientId` – the OAuth client ID that requests the token.
- `Audience` – determines where the assertion token will be used. In the case of the Impersonate grant, the audience is the OAuth server. To get the correct value use the *Issuer* property of *DiscoveryDocument*.
- `Certificate` – the *X509Certificate2* instance of the client private certificate.
- `AssertionTokenLifeTime` – lifetime of the assertion token. The default value is 300 seconds.

To get the client assertion you must call the *GetClientAssertionAsync* method:

```
ClientAssertion assertion =
   await assertionProvider.GetClientAssertionAsync();
```

Client assertion contains the following properties:

- `Type` – assertion token type.
- `Value` – assertion token.

To create the ImpersonateTokenProvider it is necessary to pass ImpersonateTokenProviderOptions:

```
// database and username values should be entered by user
String db;
String userName;

var tokenProvider = new ImpersonateTokenProvider(
   new ImpersonateTokenProviderOptions()
   {
      TokenEndpoint = discoveryDocument.TokenEndpoint,
      ClientAssertionProvider = assertionProvider,
      Scope = "Innovator",
      Database = db,
      UserName = userName
   });
```

ImpersonateTokenProviderOptions uses the following options:

- `TokenEndpoint` – URL of OAuth server token endpoint.
- `ClientAssertionProvider` – provider of assertion token.
- `Scope` – scope used to request the token.
- `Database` – name of the database to login to.
- `UserName` – user name.

Note: Custom *IClientAssertionProvider* implementation can be provided. There is no need to provide a password because authentication is performed for client-by-client credentials. In this case client credentials are assertion tokens.

You must call the GetAccessTokenAsync method to get an access token. This method sends a request to the OAuth server token endpoint and returns the access token created by the Impersonate grant.

```
string accessToken = await tokenProvider.GetAccessTokenAsync();
```

Note: The *GetAccessTokenAsync* method can take the *CancellationToken* as a parameter for cancelling a task.

10.1.1.4 Authorization Code Grant

The Authorization Code grant type is used to obtain access tokens and is optimized for confidential clients. Since this is a redirection-based flow, the client must be capable of interacting with the resource owner's user-agent (typically a web browser) and capable of receiving incoming requests (via redirection) from the authorization server.

Note: See more about the Authorization Code grant in the OAuth 2.0 RFC specification: https://tools.ietf.org/html/rfc6749#section-4.1

To request an access token using an authorization code grant it is necessary to use the *AuthorizationFlowTokenProvider*. To create the *AuthorizationFlowTokenProvider* it is necessary to create an *INavigator* instance.

The default implementation of *INavigator* is *WpfBrowserNavigator* which can be found in the IOM.OAuth.WpfBrowserNavigator.dll library. This navigator is used to allow browser based login from WPF and WinForms applications. *WpfBrowserNavigator* uses *WebBrowser* control to process navigation to URLs.

To create the *WpfBrowserNavigator* it is necessary to call a parameterless constructor. It is possibility to define values for displaying a browser window:

```
var navigator = new WpfBrowserNavigator
{
   Width = 900,
   Height = 600,
   Title = "Innovator"
};
```

WpfBrowserNavigator has the following properties:

- `Width` – width of browser window. Default value is 900.
- `Height` – height of browser window. Default value is 600.
- `Title` – title of browser window. Default value is "Innovator".
- `Owner` – owner of browser window in case of using *WpfBrowserNavigator* in WPF application. The browser window will be centered within the owner window. If this property is not set, the browser window will be centered within the current screen working area.
- `OwnerHandle` – owner of browser window in case of using WpfBrowserNavigator in not WPF application. The browser window will be centered within the owner window. If this property is not set, the browser window will be centered within the current screen working area.

Note: The IOM.OAuth.WpfBrowserNavigator.dll is stored next to .NET version of IOM.dll.

To create the AuthorizationFlowTokenProvider it is necessary to pass AuthorizationFlowTokenProviderOptions:

```
// database value should be entered by user
String db;

var tokenProvider = new AuthorizationFlowTokenProvider(
   new AuthorizationFlowTokenProviderOptions()
   {
      DiscoveryDocument = discoveryDocument,
      ClientId = "IOMApp",
      RedirectUrl = "iomapp://token",
      Scope = "Innovator",
      Database = db,
      AuthenticationType = "Windows",
      ResponseMode = ResponseMode.Query,
      Navigator = navigator
   });
```

AuthorizationFlowTokenProviderOptions have the following options:

- `DiscoveryDocument` – instance of *DiscoveryDocument*.
- `GrantType` – a grant type that is used to request an access token. The default value is *AuthorizationCode*. Only the *AuthorizationCode* grant is currently supported.
- `ClientId` – the ID of the OAuth server client that requests a token.
- `RedirectUrl` – URL to return a token.
- `Scope` – scope used to request a token.
- `Database` – name of the database to login to.
- `AuthenticationType` – authentication type to use to authenticate.
- `ResponseMode` – response mode. The Default value is Query.
- `Navigator` – instance of *INavigator*.

Note: `Database` and `AuthenticationType` options are implemented to allow direct login. If both values are set and the required authentication type allows login without the Aras Innovator login page, the OAuth server will try to redirect to the specified authentication type automatically and log the external user into the specified database.

You must call the GetAccessTokenAsync method to get an access token. This method sends a request to the OAuth server Authorize endpoint, receives an authentorization code, sends the authorization code to the token endpoint and gets the access token issued by the Authorization Code grant.

```
string accessToken = await tokenProvider.GetAccessTokenAsync();
```

Note: The *GetAccessTokenAsync* method can take the *CancellationToken* as a parameter to provide the possibility of cancelling a task.

10.1.1.5 Authorization Code Grant with Windows Identity Provider

It is possible to use the IOM API for login via Windows authentication without user interaction. To request an access token using Windows authentication you must use the *WindowsTokenProvider*. This provider uses an Authorization Code grant.

To create a *WindowsTokenProvider* it is necessary to provide *WindowsTokenProviderOptions*:

```
// database value should be entered by user
String db;

var tokenProvider = new WindowsTokenProvider(
   new WindowsTokenProviderOptions()
   {
      ClientId = "IOMApp",
      DiscoveryDocument = discoveryDocument,
      RedirectUrl = "iomapp://token",
      Scope = "Innovator",
      AuthenticationType = "Windows",
      Database = db
   });
```

WindowsTokenProviderOptions have the following options:

- `DiscoveryDocument` – an instance of *DiscoveryDocument*.
- `ClientId` – ID of OAuth server client that requests the token.
- `RedirectUrl` – URL to return token.
- `Scope` – scope used to request the token.
- `Database` – name of the database to login to.
- `AuthenticationType` – the name of the authentication type used for Windows authentication in the OAuth server.
- `MaxRedirectsCount` – maximum number of redirects during authentication. This option is required to limit the number of redirects in a redirection-based flow to prevent eternal redirections. Default value is *10*.

Note: The "iomapp://token" is the default *RedirectUri* value for the IOMApp client of the OAuth server.

You must call the GetAccessTokenAsync method to get an access token.. This method sends a request to the OAuth server Authorize endpoint, receives the authorization code, sends the authorization code to the token endpoint and gets an access token issued by the Authorization Code grant.

```
string accessToken = await tokenProvider.GetAccessTokenAsync();
```

Note: The *GetAccessTokenAsync* method can take *CancellationToken* as a parameter to nake it possible to cancel a task.

10.1.1.6 Using token providers with OData requests

To make OData requests valid for the Aras Innovator server it is necessary to have an access token. The access token should be presented in the `"Authorization"` header in the case of the *Standard* protocol type or in `"X-Aras-Authorization"` in the case of the *Custom* protocol type. The following example demonstrates using the token provider for creating an OData request to the Aras Innovator server:

```
// database, username and password values should be entered by user
String db;
String userName;
String password;

var discoveryDocumentProvider = new DiscoveryDocumentProvider(
   "https://myserver/MyInnovator/Server/InnovatorServer.aspx");
var discoveryDocument =
   await discoveryDocumentProvider.GetDiscoveryDocumentAsync();

var tokenProvider = new PasswordTokenProvider(
   new PasswordTokenProviderOptions()
   {
      TokenEndpoint = discoveryDocument.TokenEndpoint,
      ClientId = "IOMApp",
      Scope = "Innovator",
      Database = db,
      UserName = userName,
      Password = password
```

```
   });

string token = await tokenProvider.GetAccessTokenAsync();

WebRequest request =
   WebRequest("https://myserver/MyInnovator/Server/odata/" + resourcePath);
request.Method = "GET";
request.Headers.Add(
   discoveryDocument.ProtocolInfo.AuthorizationHeader, "Bearer " + token));

WebResponse response = request.GetResponse();
```

10.1.2 Using Token Provider with HttpServerConnection

The *IomFactory* class has a public method for creating a connection to the Aras Innovator server based on the provided *ITokenProvider* implementation (from 12.0):

```
public static HttpServerConnection CreateHttpServerConnection(
   string innovatorServerUrl,
   ITokenProvider tokenProvider,
   ProtocolType protocolType);
```

Note: There is CreateHttpSeverConnection overload that requires database and does not require ProtocolType and uses Standart ProtocolType by default. It is not recommended to use this overload because this method is obsolete. It's recommended to use DiscoveryDocument to get current ProtocolType and pass it to CreateHttpSeverConnection method.

There are also public methods for creating the connection without providing the *ITokenProvider.* They use default *ITokenProvider* implementations: *CreateHttpServerConnection* uses *PasswordTokenProvider*, *CreateWinAuthHttpServerConnection* uses WindowsTokenProvider.

Note: Connections that are created by *CreateHttpServerConnection* and *CreateWinAuthHttpServerConnection* methods without providing *ITokenProvider* also support legacy authentication for backward compatibility with Aras Innovator 11 SP14 and earlier.

Here is the example of code using the *CreateHttpServerConnection* method with *PasswordTokenProvider*:

```
// database, username and password values should be entered by user
String db;
String userName;
String password;

var discoveryDocumentProvider = new DiscoveryDocumentProvider(
   "https://myserver/MyInnovator/Server/InnovatorServer.aspx");
var discoveryDocument =
   await discoveryDocumentProvider.GetDiscoveryDocumentAsync();

var tokenProvider = new PasswordTokenProvider(
   new PasswordTokenProviderOptions()
   {
      TokenEndpoint = discoveryDocument.TokenEndpoint,
      ClientId = "IOMApp",
      Scope = "Innovator",
      Database = db,
      UserName = userName,
```

```
      Password = password
   });

HttpServerConnection httpServerConnection = CreateHttpServerConnection(
   "https://myserver/MyInnovator/Server/InnovatorServer.aspx",
   tokenProvider,
   discoveryDocument.ProtocolType);

httpServerConnection.Login();
```

Here is an example of code using the CreateHttpServerConnection method with AuthorizationFlowTokenProvider:

```
// database value should be entered by user
String db;

var discoveryDocumentProvider = new DiscoveryDocumentProvider(
   "https://myserver/MyInnovator/Server/InnovatorServer.aspx");
var discoveryDocument =
   await discoveryDocumentProvider.GetDiscoveryDocumentAsync();

var navigator = new WpfBrowserNavigator();

var tokenProvider = new AuthorizationFlowTokenProvider(
   new AuthorizationFlowTokenProviderOptions()
   {
      DiscoveryDocument = discoveryDocument,
      ClientId = "IOMApp",
      RedirectUrl = "iomapp://token",
      Scope = "Innovator",
      Database = db,
      AuthenticationType = "Windows",
      ResponseMode = ResponseMode.Query,
      Navigator = navigator
   });

HttpServerConnection httpServerConnection = CreateHttpServerConnection(
   "https://myserver/MyInnovator/Server/InnovatorServer.aspx",
   tokenProvider,
   discoveryDocument.ProtocolType);

httpServerConnection.Login();
```

Note: It is recommended to set values that are necessary for authentication from config files or user input so changing OAuth client settings does not affect your application.

10.2 COM-compatible IOM

A COM-compatible version of the IOM.dll can be found in the `\Utilities\Aras Innovator <Version> IOM SDK\COM` directory on the CD Image. Use this assembly to build Windows applications in VB6 or VC++ or write, for example, VBA office macros or Windows Scripting Host applications (VBScript or Jscript).

In order to use the DLL, it must be registered in Windows Registry. Use the following procedure:

1. Copy IOM.dll into a local folder.
2. Open a Command Prompt with Administrator privileges and run the RegAsm.exe tool to register the DLL for x86 or x64 architechtures respectively:

```
%windir%\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe .\IOM.dll /codebase
%windir%\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe .\IOM.dll /codebase
```

Note: IOM type library (IOM.tlb) can be found on the CD Image near IOM.dll. It is recommended to use type library provided on CD Image instead of generating new one with /tlb argument for RegAsm.exe tool.

3. If the assembly was successfully registered it must appear in the list of available COM references as "Aras IOM 12.0 API (COM-compatible)".
4. Add the reference to your project:

```
#import "{SDK_Dir}\COM\IOM.tlb"
```

5. Similar to .NET you must first create a connection, login and then create an Aras Innovator object.

Here is the sample code in C++:

```
_bstr_t url = L"https://myserver/MyInnovator/Server/InnovatorServer.aspx";
// database, username and password values should be entered by user
_bstr_t db;
_bstr_t userName;
_bstr_t password;

IOM::IIomFactoryComIncomingPtr iomFactory(__uuidof(IOM::IomFactory));
IOM::IHttpServerConnectionComIncomingPtr connection =
   iomFactory->CreateHttpServerConnection(url, db, userName, password);

IOM::IItemComIncomingPtr loginResult = connection->Login();

if (loginResult->isError())
{
   std::cout << "Failed to login.";
}
else
{
   IOM::IInnovatorComIncomingPtr innovator =
      iomFactory->CreateInnovator(connection);
}
```

10.2.1 Obtaining an Access Token from COM applications

10.2.1.1 OAuthFactory for creating COM objects

COM-compatible IOM API introduces *OAuthFactory* class. Using this factory class user can create COM instance of discovery document provider or token providers.

OAuthFactory is a CoClass with "Aras.IOM.OAuthFactory.12.0" progId, that implemets *IOAuthFactory* interface with the following methods:

```
IDiscoveryDocumentProvider* CreateDiscoveryDocumentProvider(BSTR
innovatorServerUrl)
```

The method creates *DiscoveryDocumentProvider* COM object based on Innovator server URL.

```
IPasswordTokenProviderOptions* CreatePasswordTokenProviderOptions()
```

The method creates *PasswordTokenProviderOptions* COM object which will be used while *PasswordTokenProvider* creation.

```
ITokenProvider* CreatePasswordTokenProvider(IPasswordTokenProviderOptions*
options)
```

The method creates *PasswordTokenProvider* COM object based on provided *IPasswordTokenProviderOptions*.

```
IWindowsTokenProviderOptions* CreateWindowsTokenProviderOptions()
```

The method creates *WindowsTokenProviderOptions* COM object which will be used while *WindowsTokenProvider* creation.

```
ITokenProvider* CreateWindowsTokenProvider(IWindowsTokenProviderOptions*
options)
```

The method creates *WindowsTokenProvider* COM object based on provided *IWindowsTokenProviderOptions*.

```
IAuthorizationFlowTokenProviderOptions*
CreateAuthorizationFlowTokenProviderOptions()
```

The method creates AuthorizationFlowTokenProviderOptions COM object which will be used while AuthorizationFlowTokenProvider creation.

```
ITokenProvider*
CreateAuthorizationFlowTokenProvider(IAuthorizationFlowTokenProviderOption
s* options)
```

The method creates AuthorizationFlowTokenProvider COM object based on provided IAuthorizationFlowTokenProviderOptions.

Note: Option objects are used to pass arguments to factory methods creating corresponding provider objects. Developer should set property values of option object as necessary before passing it to corresponding factory method.

Here is the example of creating *PasswordTokenProvider* with *OAuthFactory* in C++:

```
_bstr_t innovatorServerUrl =
   L"https://myserver/MyInnovator/Server/InnovatorServer.aspx";
// database, username and password values should be entered by user
_bstr_t db;
_bstr_t userName;
_bstr_t password;

IOM::IOAuthFactoryPtr oauthFactory(__uuidof(IOM::OAuthFactory));

IOM::IDiscoveryDocumentProviderPtr discoveryDocumentProvider =
   oauthFactory->CreateDiscoveryDocumentProvider(innovatorServerUrl);
```

```
IOM::IDiscoveryDocumentPtr discoveryDocument =
   discoveryDocumentProvider->GetDiscoveryDocument();

IOM::IPasswordTokenProviderOptionsPtr options =
   oauthFactory->CreatePasswordTokenProviderOptions();
      options->TokenEndpoint = discoveryDocument->GetTokenEndpoint();
      options->ClientId = L"IOMApp";
      options->Scope = L"Innovator";
      options->Database = db;
      options->UserName = userName;
      options->Password = password;

IOM::ITokenProviderPtr passwordTokenProvider =
   oauthFactory->CreatePasswordTokenProvider(options);
```

10.2.1.2 COM-compatible WpfBrowserNavigator

A COM-compatible version of IOM.OAuth.WpfBrowserNavigator.dll can be found near COM-compatible IOM.dll in the `\Utilities\Aras Innovator <Version> IOM SDK\COM` directory on the CD Image.

This assembly contains classes that provide possibility to use browser-based authentication for applications in VB6, VC++, VBScript, JScript etc.

In order to use the WpfBrowserNavigator DLL it must be registered in the same way as IOM.dll. See steps for registration in section 10.2.

Note: WpfBrowserNavigator type library (IOM.OAuth.WpfBrowserNavigator.tlb) can be found on the CD Image and it is recommended to use it in the same way as IOM.COM type library.

Here is the sample code of creating *WpfBrowserNavigator* in C++:

```
// windowHandle should be set by developer
long windowHandle;

IOM_OAuth_WpfBrowserNavigator::IWpfBrowserNavigatorPtr wpfBrowserNavigator
   (__uuidof(IOM_OAuth_WpfBrowserNavigator::WpfBrowserNavigator));

wpfBrowserNavigator->Height = 600;
wpfBrowserNavigator->Width = 980;
wpfBrowserNavigator->Title = L"Innovator";
wpfBrowserNavigator->OwnerHandle = windowHandle;
```

Note: The browser window will be centered within the owner window specified by OwnerHandle property. If this property is not set, the browser window will be centered within the current screen working area.

10.2.2 Using Token Provider with HttpServerConnection

The *IomFactory* class has a public *CreateHttpServerConnection2* method for creating a connection to the Aras Innovator server based on the provided *ITokenProvider* implementation.

CreateHttpServerConnection2 method takes the following parameters:

- `innovatorServerUrl` – URL to Innovator server.

- `tokenProvider` – instance of ITokenProvider.
- `protocolType` – information about protocol that is used by the OAuth server.

Here is an example of creating an HttpServerConnection with AuthorizationFlowTokenProvider in C++:

```
IOM::IOAuthFactoryPtr oauthFactory(__uuidof(IOM::OAuthFactory));

_bstr_t innovatorServerUrl =
   L"https://myserver/MyInnovator/Server/InnovatorServer.aspx"

IOM::IDiscoveryDocumentProviderPtr discoveryDocumentProvider =
   oauthFactory->CreateDiscoveryDocumentProvider(innovatorServerUrl);
IOM::IDiscoveryDocumentPtr discoveryDocument =
   discoveryDocumentProvider->GetDiscoveryDocument();

// windowHandle should be set by developer
long windowHandle;

IOM_OAuth_WpfBrowserNavigator::IWpfBrowserNavigatorPtr wpfBrowserNavigator
   (__uuidof(IOM_OAuth_WpfBrowserNavigator::WpfBrowserNavigator));

wpfBrowserNavigator->Height = 600;
wpfBrowserNavigator->Width = 980;
wpfBrowserNavigator->Title = L"Innovator";
wpfBrowserNavigator->OwnerHandle = windowHandle;

IOM::IAuthorizationFlowTokenProviderOptionsPtr options =
   oauthFactory->CreateAuthorizationFlowTokenProviderOptions();
options->DiscoveryDocument = discoveryDocument;
options->ClientId = L"IOMApp";
options->Scope = L"Innovator";
options->RedirectUri = L"iomapp://token/";
options->ResponseMode = IOM::ResponseMode::ResponseMode_Query;
options->Navigator = wpfBrowserNavigator;

IOM::ITokenProviderPtr tokenProvider =
   oauthFactory->CreateAuthorizationFlowTokenProvider(options);

IOM::IHttpServerConnectionComIncomingPtr connection =
   iomFactory->CreateHttpServerConnection2(
      innovatorServerUrl,
      tokenProvider,
      discoveryDocument->GetProtocolType());
```

10.3 iOS IOM

An iOS version of IOM.IOS.dll can be found in the `\Utilities\Aras Innovator <Version> IOM SDK\iOS` directory on the CD Image. You can copy it to another location and reference it by an iOS Mobile App using Xamarin's existing framework to generate an iOS App project. IOM APIs belong to the Aras.IOM namespace. First, every application must create a connection with the Aras Innovator server and login to the server using the connection. If the login succeeds then you must create an instance of class `Innovator` with the connection. Here is a small sample:

```
…
using Aras.IOM;
```

```
…
   //create connection to Innovator server
   string server = "https://myserver/MyInnovator";
   // database, username and password values should be entered by user
   string db;
   string username;
   string password;
   HttpServerConnection connection =
IomFactory.CreateHttpServerConnection(server, db, username, password);
   //login
   Item user = await connection.LoginAsync();
   if (user.isError())
   {
      throw new Exception(user.getErrorString());
   }
```

10.4 Android IOM

A RT version of IOM.Android.dll can be found in the `\Utilities\Aras Innovator <Version> IOM SDK\Android` directory on the CD Image. You can copy it to another location and reference it by an Android App using Xamarin's existing framework to generate an Android project. IOM APIs belong to Aras.IOM namespace. First, every application must create a connection with the Aras Innovator server and login to the server using the connection. If the login succeeds then create an instance of class `Innovator` with the connection. Here is a small sample:

```
…
using Aras.IOM;
…
   //create connection to Innovator server
   string server = "https://myserver/MyInnovator";
   // database, username and password values should be entered by user
   string db;
   string username;
   string password;
   HttpServerConnection connection =
IomFactory.CreateHttpServerConnection(server, db, username, password);
   //login
   Item user = await connection.LoginAsync();
   if (user.isError())
   {
      throw new Exception(user.getErrorString());
```

11 Using CheckinManager

The CheckInManager is a function built into the IOM.dll which allows for the loading of large data structures into Aras Innovator that contain Images, Drawings, Documents, or other Items of the File Itemtype.

The following use cases describe how to use CheckInManager to submit images and drawings, either individually or as a collection.

11.1 Submitting Images

The following is the first step in submitting a File structure to CheckInManager is to create a CAD item and add properties to it. The following code shows a typical AML query for creating a CAD item:

```
Item cad0 = innovator.newItem("CAD", "add");
cad0.setProperty("item_number", innovator.getNewID()); // required by CAD ItemType
cad0.setFileProperty("native_file", Path.Combine(directory, "input/native-0.cad"));
cad0.setFileProperty("thumbnail", Path.Combine(directory, "input/thumbnail-0.png"));
cad0.setPropertyAttribute("thumbnail", "checkinManager-type", "Image");
cad0.setProperty("name", "submitting-images-cad0");
```

In this scenario, the setFileProperty method is required in order to add a property whose type is File. The parameters of setFileProperty include the Property's name and the fully qualified path of the File.

If you need to use the CheckinManager to load an Image Property, you must set the Attribute *checkinManager-type* on the Property (such as in the thumbnail, below) to "Image". The following example shows the AML for the query:

```
<Item isNew="1" isTemp="1" type="CAD" action="add" id="FF678B0D9D43474E82305E232B5FF448">
        <item_number>57B354B0FEBF4F5BA5CCCCC6361A892B</item_number>
        <native_file>
                <Item isNew="1" isTemp="1" type="File" action="add" id="A276434AABD6420AA6640D0CAD4E97C3">
                        <filename>native-0.cad</filename>
                        <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\native-0.cad</actual_filename>
                        <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                        <Relationships>
                                <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                </Item>
                        </Relationships>
                </Item>
        </native_file>
        <thumbnail checkinManager-type="Image">
                <Item isNew="1" isTemp="1" type="File" action="add" id="ED6FF3D391E045C5A9D53424036B4862">
                        <filename>thumbnail-0.png</filename>
                        <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\thumbnail-0.png</actual_filename>
                        <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                        <Relationships>
                                <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                </Item>
                        </Relationships>
                </Item>
        </thumbnail>
        <name>submitting-images-cad0</name>
</Item>
```

If you are only importing one Item, you can then pass the Item directly to the CheckinManager. The following example uses `cad0` from the previous scenario:

```
using (CheckinManager manager = new IomFactory().CreateCheckinManager(cad0))
{
        Item response = manager.Checkin(numThreads);
        // …
}
```

You can also use this code to import more than one item. For example, given two CAD Items, referenced as `cad0` and `cad1` in the following example, you can generate 1 AML that contains both Items, as demonstrated here:

```
Item configuration = innovator.newItem();
configuration.loadAML("<AML>" + cad0.node.OuterXml + cad1.node.OuterXml + "</AML>");
```

The following is the AML code which would result from the `loadAML` statement:

```
<AML>
        <Item isNew="1" isTemp="1" type="CAD" action="add" id="FF678B0D9D43474E82305E232B5FF448">
                <item_number>57B354B0FEBF4F5BA5CCCCC6361A892B</item_number>
                <native_file>
                        <Item isNew="1" isTemp="1" type="File" action="add" id="A276434AABD6420AA6640D0CAD4E97C3">
                                <filename>native-0.cad</filename>
                                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\native-0.cad</actual_filename>
                                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                                <Relationships>
                                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                                <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                        </Item>
                                </Relationships>
                        </Item>
                </native_file>
                <thumbnail checkinManager-type="Image">
                        <Item isNew="1" isTemp="1" type="File" action="add" id="ED6FF3D391E045C5A9D53424036B4862">
                                <filename>thumbnail-0.png</filename>
                                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\thumbnail-0.png</actual_filename>
                                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                                <Relationships>
                                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                                <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                        </Item>
                                </Relationships>
                        </Item>
                </thumbnail>
                <name>submitting-images-cad0</name>
        </Item>
        <Item isNew="1" isTemp="1" type="CAD" action="add" id="4F2A8F67E6274E648AEDBD423637DBAE">
                <item_number>7120951EA16243D0BF28A0DDA32A0187</item_number>
                <native_file>
                        <Item isNew="1" isTemp="1" type="File" action="add" id="FE29E80F224F44EBA993E8FF54859DFE">
                                <filename>native-1.cad</filename>
                                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\native-1.cad</actual_filename>
                                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                                <Relationships>
                                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                                <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                        </Item>
                                </Relationships>
                        </Item>
                </native_file>
                <thumbnail checkinManager-type="Image">
                        <Item isNew="1" isTemp="1" type="File" action="add" id="14367296E7ED45E284800F6F28127E3A">
                                <filename>thumbnail-1.png</filename>
                                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\thumbnail-1.png</actual_filename>
                                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                                <Relationships>
                                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                                <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                        </Item>
                                </Relationships>
                        </Item>
                </thumbnail>
                <name>submitting-images-cad1</name>
        </Item>
</AML>
```

The following code can then be used to apply the query contained in the `configuration` value:

```
using (CheckinManager manager = new IomFactory().CreateCheckinManager(configuration))
{
       Item response = manager.Checkin(numThreads);
       // …
}
```

You must make sure you observe the following requirements in order for the query to work correctly:

- You must have at least one File or Image property otherwise CheckinManager will throw an exception.
- The Item passed to `CreateCheckinManager` cannot be null.
- A File item must have either an Add or a Create action associated with it.
- CAD items only support the following actions:
    - Add
    - Update
    - Delete
    - Version
    - Skip

  Specifying any other type of action results in an exception being thrown.

11.2 Submitting Drawings

In this use case, the Items are set up the same way they were set up in the previous use case. The difference is that the items created here have multiple parents. You must instantiate the MultiParentConfigurationBuilder class to add more than one parent. The following code relies on an item `configuration`, as shown in the previous examples:

```
MultiParentConfigurationBuilder builder =
factory.CreateMultiParentConfigurationBuilder(configuration);

Item drawingA = innovator.newItem("CAD", "add");
drawingA.setProperty("item_number", innovator.getNewID()); // required by CAD ItemType
drawingA.setFileProperty("native_file", Path.Combine(directory, "input/native-
A.cad"));
drawingA.setProperty("name", "submitting-images-drawingA");
builder.addParent(cadA.getID(), drawingA, "CAD Structure");
```

The following is the AML for the `cadA` Item from the example:

```
<Item isNew="1" isTemp="1" type="CAD" action="add" id="4C8DD7F5785E4E9D97461BDF2D1228D3">
        <item_number>7C9FD065FAAC4D4F955820E0FEC5CB5A</item_number>
        <name>submitting-drawings-cadA</name>
        <Relationships>
                <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                        <related_id>
                                <Item isNew="1" isTemp="1" type="CAD" action="add" id="774A4C99DDC7454E9D5A0BB84EA60459">
                                        <item_number>C8CB6EB081A24945A100C3CE8FE0BFA6</item_number>
                                        <name>submitting-drawings-cadB</name>
                                </Item>
                        </related_id>
                </Item>
                <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                        <related_id>
                                <Item isNew="1" isTemp="1" type="CAD" action="add" id="B236FBFD63CA4BD89AEF8A443E5942FE">
                                        <item_number>DCCA69BB52AE4CC185FBD3B450EF314B</item_number>
                                        <name>submitting-drawings-cadC</name>
                                </Item>
                        </related_id>
                </Item>
        </Relationships>
</Item>
```

The following shows the AML code for the drawingA Item:

```
<Item isNew="1" isTemp="1" type="CAD" action="add" id="AF1763AA429D41119C896E5170D1A725">
        <item_number>4816709F699946B595B4FBDF55FCE313</item_number>
        <native_file>
                <Item isNew="1" isTemp="1" type="File" action="add" id="8C83A9DD1EED4AA586CF6A53E7D542B3">
                        <filename>native-A.cad</filename>
                        <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-drawings\bin\Debug\input\native-A.cad</actual_filename>
                        <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-drawings\bin\Debug\input</checkedout_path>
                        <Relationships>
                                <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                </Item>
                        </Relationships>
                </Item>
        </native_file>
        <name>submitting-drawings-drawingA</name>
</Item>
```

The addParent function creates a link between these two structures that generates the complete structure, as shown in the final CAD structure shown below:

The following is the AML code for *multiParentConfiguration*:

```
<AML>
    <Item isNew="1" isTemp="1" type="CAD" action="add" id="4C8DD7F5785E4E9D97461BDF2D1228D3">
        <item_number>7C9FD065FAAC4D4F955820E0FEC5CB5A</item_number>
        <name>submitting-drawings-cadA</name>
        <Relationships>
            <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                <related_id>
                    <Item isNew="1" isTemp="1" type="CAD" action="add" id="774A4C99DDC7454E9D5A0BB84EA60459">
                        <item_number>C8CB6EB081A24945A100C3CE8FE0BFA6</item_number>
                        <name>submitting-drawings-cadB</name>
                    </Item>
                </related_id>
            </Item>
            <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                <related_id>
                    <Item isNew="1" isTemp="1" type="CAD" action="add" id="B236FBFD63CA4BD89AEF8A443E5942FE">
                        <item_number>DCCA69BB52AE4CC185FBD3B450EF314B</item_number>
                        <name>submitting-drawings-cadC</name>
                    </Item>
                </related_id>
            </Item>
        </Relationships>
    </Item>
    <Item isNew="1" isTemp="1" type="CAD" action="add" id="AF1763AA429D41119C896E5170D1A725">
        <item_number>4816709F699946B595B4FBDF55FCE313</item_number>
        <native_file>
            <Item isNew="1" isTemp="1" type="File" action="add" id="8C83A9DD1EED4AA586CF6A53E7D542B3">
                <filename>native-A.cad</filename>
                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                    drawings\bin\Debug\input\native-A.cad</actual_filename>
                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                    drawings\bin\Debug\input</checkedout_path>
                <Relationships>
                    <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                    </Item>
                </Relationships>
            </Item>
        </native_file>
        <name>submitting-drawings-drawingA</name>
        <Relationships>
            <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                <related_id>4C8DD7F5785E4E9D97461BDF2D1228D3</related_id>
            </Item>
        </Relationships>
    </Item>
    <Item isNew="1" isTemp="1" type="CAD" action="add" id="D0AD5B11F58F4CB79978C2FB4BFD1A71">
        <item_number>D54200E995B9410F808EEB26FB9E04AB</item_number>
        <native_file>
            <Item isNew="1" isTemp="1" type="File" action="add" id="DFE1AA07106B4AC4B63595E8CC89833F">
                <filename>native-B.cad</filename>
                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                    drawings\bin\Debug\input\native-B.cad</actual_filename>
                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                    drawings\bin\Debug\input</checkedout_path>
                <Relationships>
                    <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                    </Item>
                </Relationships>
            </Item>
        </native_file>
        <name>submitting-drawings-drawingB</name>
        <Relationships>
            <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                <related_id>774A4C99DDC7454E9D5A0BB84EA60459</related_id>
            </Item>
```

```
            </Relationships>
        </Item>
        <Item isNew="1" isTemp="1" type="CAD" action="add" id="E5F643DCC71A4740AC29189B064C9AC6">
            <item_number>5FE746744D8C433281F32C9588F2D64B</item_number>
            <native_file>
                <Item isNew="1" isTemp="1" type="File" action="add" id="8C4C44D934B14CE0A8779008F4395F89">
                    <filename>native-C.cad</filename>
                    <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                        drawings\bin\Debug\input\native-C.cad</actual_filename>
                    <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
drawings\bin\Debug\input</checkedout_path>
                    <Relationships>
                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                            <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                        </Item>
                    </Relationships>
                </Item>
            </native_file>
            <name>submitting-drawings-drawingC</name>
            <Relationships>
                <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                    <related_id>B236FBFD63CA4BD89AEF8A443E5942FE</related_id>
                </Item>
            </Relationships>
        </Item>
</AML>
```

In this case, `cadA` is equal to *configuration*. The last line of code creates a `CAD Structure` relationship between `drawingA` and `cadA`. The example repeats similar logic for `cadB` and `cadC`. The following code then applies the query:

```
Item multiParentConfiguration = builder.GetItemConfiguration();

using (CheckinManager manager =
factory.CreateCheckinManager(multiParentConfiguration))
{
        Item response = manager.Checkin(numThreads);

        // …

}
```

The first line converts the Builder's query into a query that supports multiple parents as determined by the calls to *builder.addParent*. After the first line of code, query execution is applied in the same way as a standard CheckInManager query.

11.3 Async Processing

The previous use cases allow the main thread to do the processing. In this use case, the class offloads all logic to a separate thread from the executing thread. Because of this, you need to keep the following in mind:

- Using the Dispose pattern as the main thread continues execution and disposes of the `CheckInManager` will cause a `NullReferenceException`.
- Retrieving the results from the `CheckinManager` requires a callback for either `UploadFilesCompleted` or `CheckinCompleted`. `UploadFilesCompleted` loops through all the files and writes out which files have been added. If an error is encountered, an exception is thrown. `CheckInCompleted` stores a list of all the results.
- In `CheckinCompleted`, you must dispose of the `CheckInManager` to prevent a resource leak