Aras Innovator 35

Programmer's Guide

Document #: *D-009011*

Last Modified: *05/27/2025*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810

Notice of Rights

Notice of Liability

Table of Contents

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.

Note: Starting with Aras Innovator Release 35, the Aras IOM SDK is released as a standalone package and is no longer bundled with the Aras Innovator CD Image. For full documentation of the IOM, refer to the Aras IOM SDK Programmer's Guide located in the Documentation folder within the Aras IOM SDK\<Version> folder on the FTP, or on the Aras Subscriber Portal.

- How Methods work and the Methodology for implementing your own business logic.
- A Cookbook of recipes for performing common tasks.

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 in a similar way 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 Aras Innovator Runs on .NET Core

All server components run on .NET 8.0.1

.NET 8.0 is an open-source, cross-platform framework that runs on Windows, macOS, and Linux operating systems. It is a successor of .NET Framework and was written from scratch to make it modular, lightweight, and fast.

2.1 Platform Differences to Consider

Despite of being a successor of .NET Framework, .NET Core and .NET 8.0 do not provide full feature parity and brings various breaking changes that developers must understand when implementing new server-side code or migrating existing one:

- New web framework

  Aras Innovator has been migrated from classic ASP.NET web framework to run on ASP.NET Core - an open-source and cross-platform framework for building modern web applications with the aim to provide the access to modern features, better development experience and greater performance.

  Aras Innovator now uses Kestrel as a primary web server – lightweight cross-platform web server for ASP.NET Core applications. When installed via MSI package, IIS is used only as a reverse proxy while still being configured via web.config. To learn more about IIS hosting configuration please visit ASP.NET Core Module article.

- Unsupported API and breaking changes

  There are number of APIs and technologies are not yet ported to .NET Core and .NET 8.0, considered as deprecated (thus no longer supported on a target platform and throws "Unsupported" exception at runtime), have breaking changes in functionality behavior, or are completely removed. Please see the following official Microsoft documentation about the differences between .NET Framework and .NET 8.0:

- Assembly Redirection

  In prior releases, when developing a server method that calls custom DLL that references different version of a DLL that is also used by the Innovator, there is a chance to receive runtime errors due to an assembly version mismatch. To address this, assembly binding instructions must be added to the web.config file to instruct the runtime to properly load an assembly.

  In .NET Core and .NET 8.0 there is no longer such concept of binding redirections so no changes in web.config are needed. Aras Innovator will automatically pick up and load an assembly from application binaries folder.

2.2 Cross-Platform Development

Despite of being cross-platform and providing high-level abstractions for low-level APIs, some .NET Core and .NET 8.0 API still may work differently depending on a current platform it is running on.

It is necessary for developers to know and understand these differences to be sure that the application works identically on all supported platforms:

- **Working with native code**

  .NET provides mechanisms to call native code via Platform Invoke.

  There might be an existing code that uses P\Invoke to call Windows-specific API (e.g.via kernel32.dll), and since it is not present on Linux, the code will not work when running in Linux.To avoid compatibility issues, it is better to use available .NET Core and .NET 8.0 functionality or 3rd-party libraries that provide cross-platform capabilities.

  If different libraries need to be loaded or APIs called depending on a current OS, consider decorating the code with platform-specific blocks:

  ```
  if (RuntimeInformation.IsOSPlatform(OSPlatform.Windows))
  {
     // Call Windows-specific API
  }
  else
  {
     // Call Linux-specific API
  }
  ```

  Follow the link to find more about native interoperability: Native interoperability - .NET

- **OS-specific APIs**

  .NET Core and .NET 8.0 exposes several APIs that might not be available for the given platform. Such APIs throw `PlatformNotSupportedException` when called from an unsupported OS. For instance, `Process.MainWindowHandle` throws the exception on Linux and macOS.

  Make sure you do not use one of these APIs if you plan to run the Innovator on multiple platforms.

  Follow the link to find more about unsupported APIs: APIs that always throw exceptions on .NET Core and .NET 8.0

- **File system case-sensitivity**

  Windows uses a case-insensitife file system allowing to access files regardless of path casing, so a single file or folder can be accessed by paths written in different cases, e.g. `c:\Innovator\Aras.Server.exe` **is equal to** `c:\INNOVATOR\aras.server.EXE`

  Unlike Windows, the Linux file system is case-sensitive, meaning that two paths with different casing will point to a two different files\directories, e.g.

  `/app/bin/Aras.Server.exe` **is not equal to** `/app/bin/Aras.server.EXE`

  During development, make sure that all paths in code match corresponding entities on file system to avoid any issues when running Aras Innovator on Linux.

- **Filesystem path delimiter**

  In Windows, it is allowed to use both slash ('/') and backslash ('\') as a path delimiter as they cannot be part of file\folder name, so the next two paths are equal:

  `c:\Innovator\Aras.Server.exe` **is equal to** `c:/INNOVATOR/aras.server.EXE`

  On Linux, because the backslash is a valid file name character, the only allowed path delimiter is slash ('/') so

  `/app/Innovator/Aras.Server.exe` **is not equal to** `/app/Innovator\Aras.server.EXE`

  To avoid any issues with accessing file system, it is strongly recommended to avoid building paths manually, and use special .NET API instead to build platform-specific file path:

  - `Path.Combine()`
  - `Path.Join()`
  - `Path.DirectorySeparatorChar`

  Example:

  Calling `Path.Join(Directory.GetCurrentDirectory(), "data");` will produce the following results:

  On Windows: `"C:\Innovator\data"`

  On Linux: `"/app/Innovator/data"`

- **Cultures and string comparison**

  .NET exposes `CultureInfo` API to work with different cultures. This API is platform-dependent and may work differently on different platforms: Windows and Linux do not use identical lists of cultures. In Windows it is part of the installation while Linux relies on data from International Components of Unicode (ICU). Another difference is that certain cultures may not be installed on a particular OS.

  String comparison with the use of InvariantCulture and CurrentCulture settings works differently on Windows and Linux.

  To avoid cross-platform issues, it is highly recommended to specify string comparison/culture explicitly and follow Best Practices for Comparing Strings in .NET

- **Date and time formatting**

  Date and Time .NET API may return date and time in a different format depending on the current OS and its regional settings. For instance, calling `DateTime.Now.ToString();` will produce the following results:

  On Windows: `"05/21/2021 2:22:55 PM"`

  On Linux: `"05/21/2021 15:22:55"`

  It is recommended to explicitly set culture or date/time format to avoid the differences between the date format on Linux and Windows.

- **Timezones**

  .NET provides an API to work with time zones via `TimeZoneInfo`. Depending on a platform, the result of calling this API will be different: Windows maintains its own list of timezones while Linux uses IANA time zones. These formats are not identical and might have different names for the same time zone. For instance, the result of calling `TimeZoneInfo.Local` will be as follows:

  On Windows: `"(UTC+03:00) Minsk"`

  On Linux: `"(UTC+03:00) GMT+03:00"`

  Please note that the Innovator has a custom implementation for mapping between these formats. It always expects that the time zone name is provided in the Windows format and transforms it to IANA name when TimeZoneInfo API is needed on non-Windows platform. In addition, the time zone name on Aras Innovator Client is always converted to the Windows format from IANA.

- **Line ending**

  Line endings vary by OS. Windows uses carriage return and line feed ("\r\n") as a line ending while UNIX-based OSes use just line feed ("\n").

  In .NET calling Environment.NewLine will also return different values depending on a current OS.

  On Windows:

  ```
  Environment.NewLine => "\r\n";
  ```

  On Linux:

  ```
  Environment.NewLine => "\n";
  ```

3 AML

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:

3.1 <Item> Tag

The `<Item>` tag defines an Item instance. XML is case-sensitive (notice the capitalized Item.) There are three principal 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 5.2 Built in Action Methods for the list of Aras Innovator pre-built actions.

3.2 <Relationships> Tag

Items can have relationships to other Items. The `<Relationships>` tag is a container tag that holds the set of relationship Items. 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>
```

3.3 <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>
```

3.4 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:

3.4.1 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 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.

Attribute	Type	Usage
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 are 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'.

3.4.2 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</td><td rowspan="6">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>ne</td></tr>
<tr><td>ge</td></tr>
<tr><td>le</td></tr>
<tr><td>gt</td></tr>
<tr><td>lt</td></tr>
<tr><td>like
not like</td><td>The value for the Property tag would include wild card symbols, which are % for any set of characters.
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>
<tr><td>Xml:lang</td><td>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></td></tr>
</table>

4 IOM Reference

The Innovator Object Model (IOM) provides a programmatic interface to build and execute AML queries against the Aras Innovator Server. It is widely used in Server Methods, Integrations, and External Applications to perform common operations such as querying Items, manipulating relationships, promoting workflow states, or interacting with the vault.

Starting with Aras Innovator Release 35, the Aras IOM SDK is released as a standalone package and is no longer bundled with the Aras Innovator CD Image. It can be found in the Aras IOM SDK\<Version> on the FTP.

For documentation refer to the Aras IOM SDK Programmer's Guide located in the Documentation folder within the Aras IOM SDK\<Version> folder on the FTP, or on the Aras Subscriber Portal.

A more detailed API reference may be obtained from one of the following:

- **On-line**: Go to https://www.aras.com/community/subscriber-portal/p/documentation
  Under the section **Aras APIs**, click **Aras IOM API**.
- **From Aras Innovator Client UI**: Login as Innovator Administrator. Under the User Menu -> Admin, select API Reference (.NET).

5 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.

5.1 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();
```

5.1.1 Context Item

As mentioned before Item Action Methods are executed on an instance of Item which is called a context item (read section 4.2 Built in Action Methods for more information on how a context item is obtained for Item Action Methods). The context item must be referenced inside Item Action Methods as `this` keyword in JavaScript, and C#, and the `Me` keyword Object in VB.NET. The context item is an instance of the IOM Item class; correspondingly any methods of the IOM Item class (see section **Error! Reference source not found.** for more details) could be called on the context item, e.g this.getProperty("foo") (C#) or Me.getProperty("foo") (VB.NET).

Note: To be able to execute a 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.

5.1.2 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>.");
```

5.1.3 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;
```

5.1.4 Methodology

One of the principal 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. Like 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 the result of work done by the Item Action Method. An Item Action Method might work with a 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 the Method code chooses to modify the context item and not create a new item, the context item must be returned from the method.

5.2 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	Adds the Item as an instance of an ItemType.
update	Updates the Item. o The Item must be locked. o 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. Performance improvement is gained by limiting the features typically available in Innovator GetItem (no server events or access checking on the sub level Items).

Built in Action Method	Comments
GetItemRepeatConfig	Performs a recursive query on a related item. It will continue getting all 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.

5.3 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.

5.3.1 Context Item

The context Item is `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.

5.3.2 Methods are Item Factories

The Generic Method must return an Item or an Error like '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());
```

5.3.3 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.

5.4 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 5.4.3.

5.4.1 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 `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.

5.4.2 Methodology

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

C#

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

In the case where a 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 the server is about to send back to the client and Server Event Methods that fully replace server actions (`OnXXX`) get client request AML as a context item and must replace it with the 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 the client is able to parse it.)

5.4.3 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` or `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.
- `OnLock`
  - Runs in place of the built-in lock action. Neither `OnBeforeLock` nor `OnAfterLock` events are called when using `OnLock`.
  - An `OnLock` method completely replaces the built-in lock action and is typically used for federated items. The method is expected to mark appropriate records in the database as locked.
  - When event handling method uses CSharpInOut template, OnLockEventArgs eventData parameter is available. It has BaseLockItem(outDom) method, which executes default lock mechanism, which calls `OnBeforeLock` and `OnAfterLock` event handlers.
- `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.
- `OnUnlock`
  - Runs in place of the built-in unlock action. Neither `OnBeforeUnlock` nor `OnAfterUnlock` events are called when using `OnUnlock`.
  - An `OnUnlock` method completely replaces the built-in unlock action and is typically used for federated items. The method is expected to mark appropriate records in the database as unlocked.
  - When event handling method uses CSharpInOut template, OnUnlockEventArgs eventData parameter is available. It has BaseUnlockItem(outDom) method, which executes default unlock mechanism, which calls `OnBeforeUnlock` and `OnAfterUnlock` event handlers.
- `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.

5.4.3.1 Example of custom OnLock server event handler

Code sample below demonstrates custom Onlock event handler, which consists of 3 steps:

1. Method "ValidateInput" validates input item before lock action.
2. "eventData.BaseLockItem(outDom)" call executes default Innovator OnLock handler, which mark corresponding database items as locked and call OnBeforeLock and OnAfterlock event handlers.
3. Method "PostProcessOutput" changes output AML after items have locked.

If itemtype has no assigned OnLock handlers, default lock implementation consists of only second step, called by Innovator. First and third steps represent custom additional business logic.

It is possible not to call second step at all (for example, for federated itemtypes, which do not have corresponding database tables). However, in such case OnBeforeLock and OnAfterlock event handlers will not be called.

```
//MethodTemplateName=CSharpInOut;
    ValidateInput(inDom, eventData);

    // call base implementation with Before- and After- events
    eventData.BaseLockItem(outDom);

    PostProcessOutput(outDom);
}

private void ValidateInput(XmlDocument inDom)
{
    // perform input AML validation
}

private void PostProcessOutput(XmlDocument outDom)
{
    // perform output AML post-processing
```

5.4.4 Polymorphic ItemTypes Server Event Inheritance

Administrators can define a server event against a Polymorphic ItemType. The benefits of defining the server event against the Polymorphic ItemType is that the event will be processed for all poly-sources.

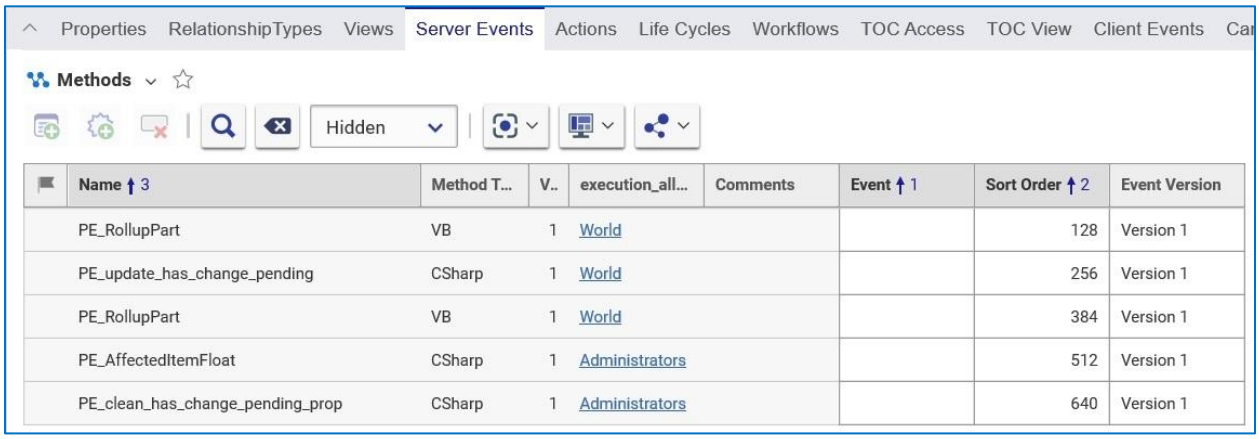
For example, if you want a single server event to be triggered against all `CAD`, `Document`, and `Part` ItemTypes, you could simply add an event handler to a server event on the `Change Controlled Item` ItemType. Once defined against the Polymorphic ItemType, all poly sources will inherit the event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

5.4.5 Required Server Events

Administrators can mark a server event as "required". 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.



Name 3	Method T...	V..	execution_all...	Comments	Event 1	Sort Order 2	Event Version
PE_RollupPart	VB	1	World			128	Version 1
PE_update_has_change_pending	CSharp	1	World			256	Version 1
PE_RollupPart	VB	1	World			384	Version 1
PE_AffectedItemFloat	CSharp	1	Administrators			512	Version 1
PE_clean_has_change_pending_prop	CSharp	1	Administrators			640	Version 1

5.4.6 Server Event Version

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.

5.4.6.1 onAfterUpdate, onAfterAdd, and onAfterVersion Version 2

The onAfterUpdate, onAfterAdd, and onAfterVersion Version 2 event is designed for use when the *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.

5.4.6.2 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.

5.5 Client Events

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

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

5.5.1 Context Item

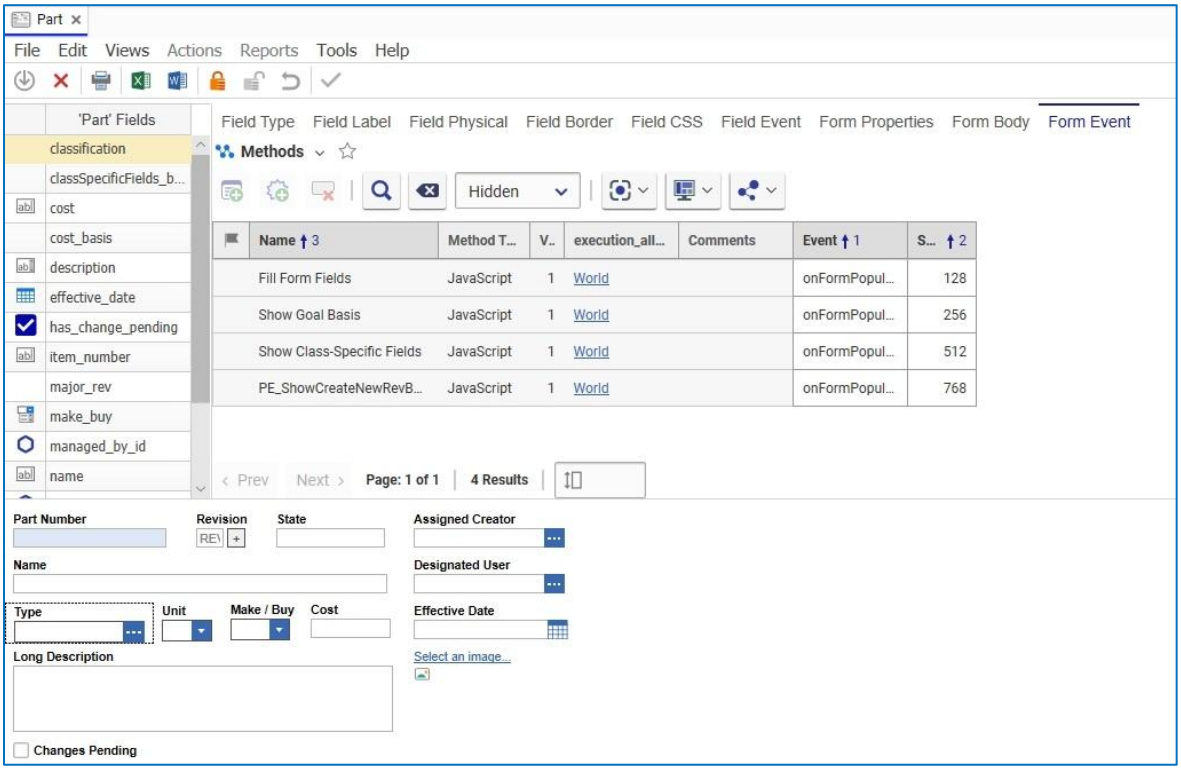
The `this` keyword context Object is an Item Object for Item Actions. However, the context Object is not the Item Object for Form, Field, and Grid Events. This context Object is the browser document (DOM) Object for the Form and Grid Events and is the Field Object for Field Events.

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.

5.5.2 Form Events

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

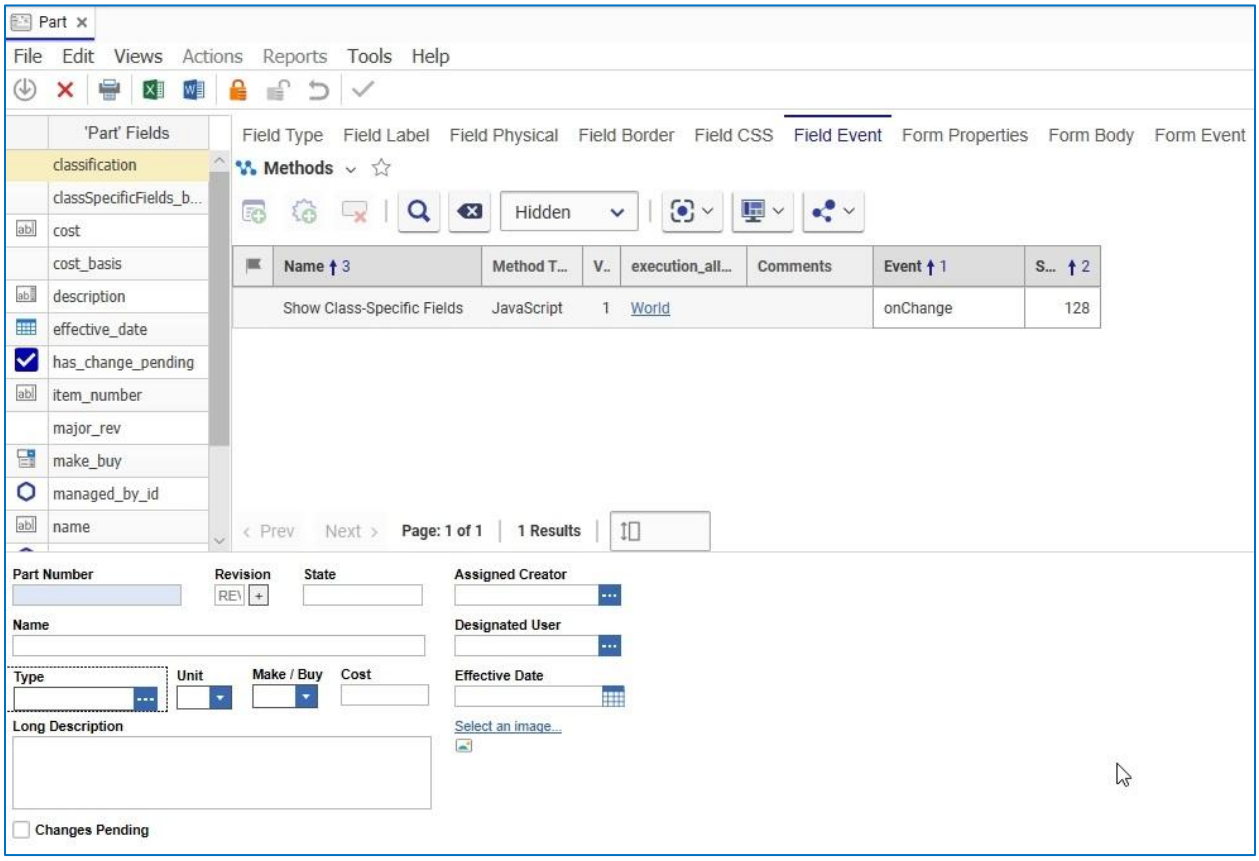
You bind your Method to the Form Event using the Form Tool. Select the Form Event tab and add the event relationships as shown below:



5.5.3 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 procedure:

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 here:

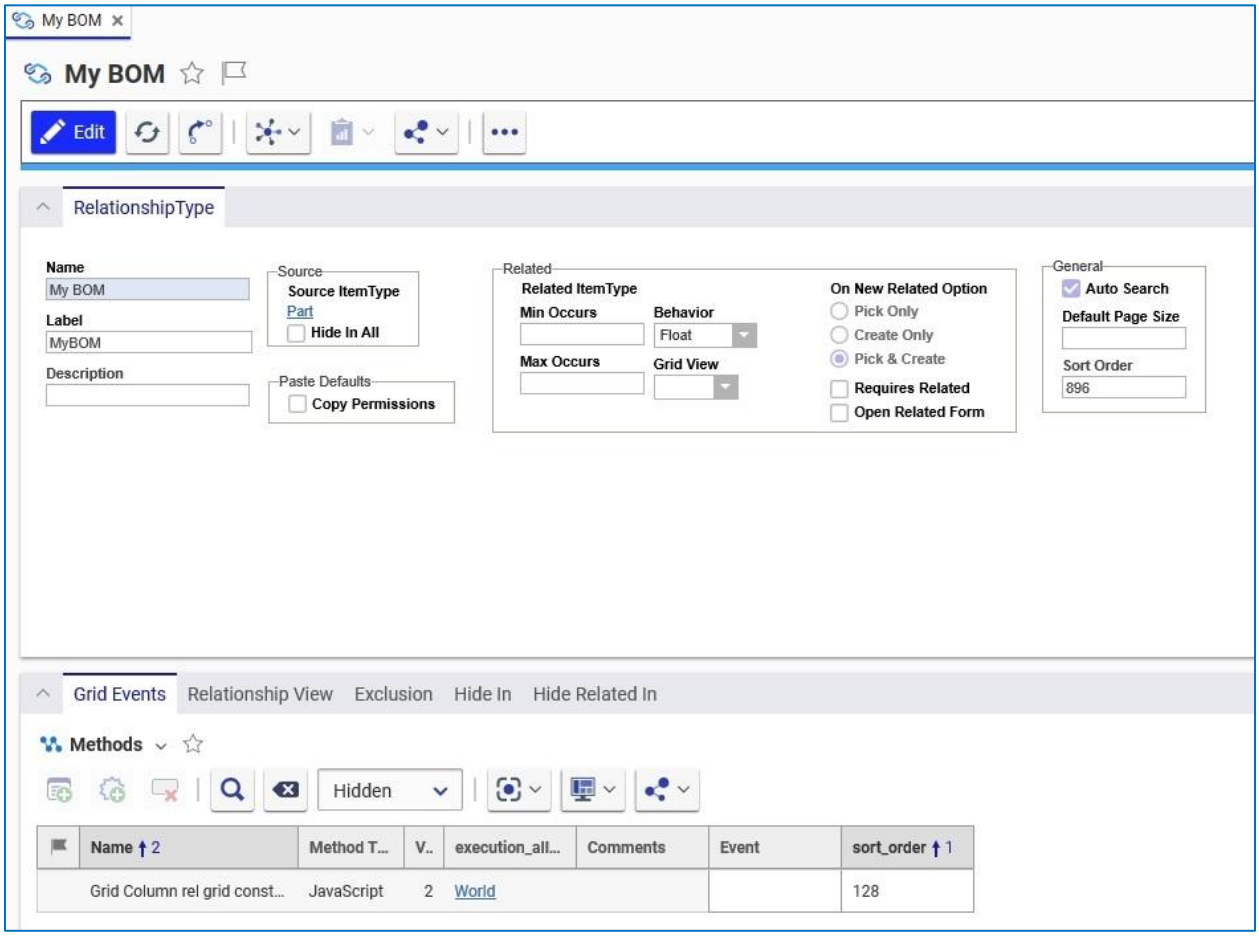


5.5.4 Grid Events

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 5.5.4.1) and on the cell (section 5.5.4.2.)

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 may be 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 here:



5.5.4.1 Row Events

The row events include:

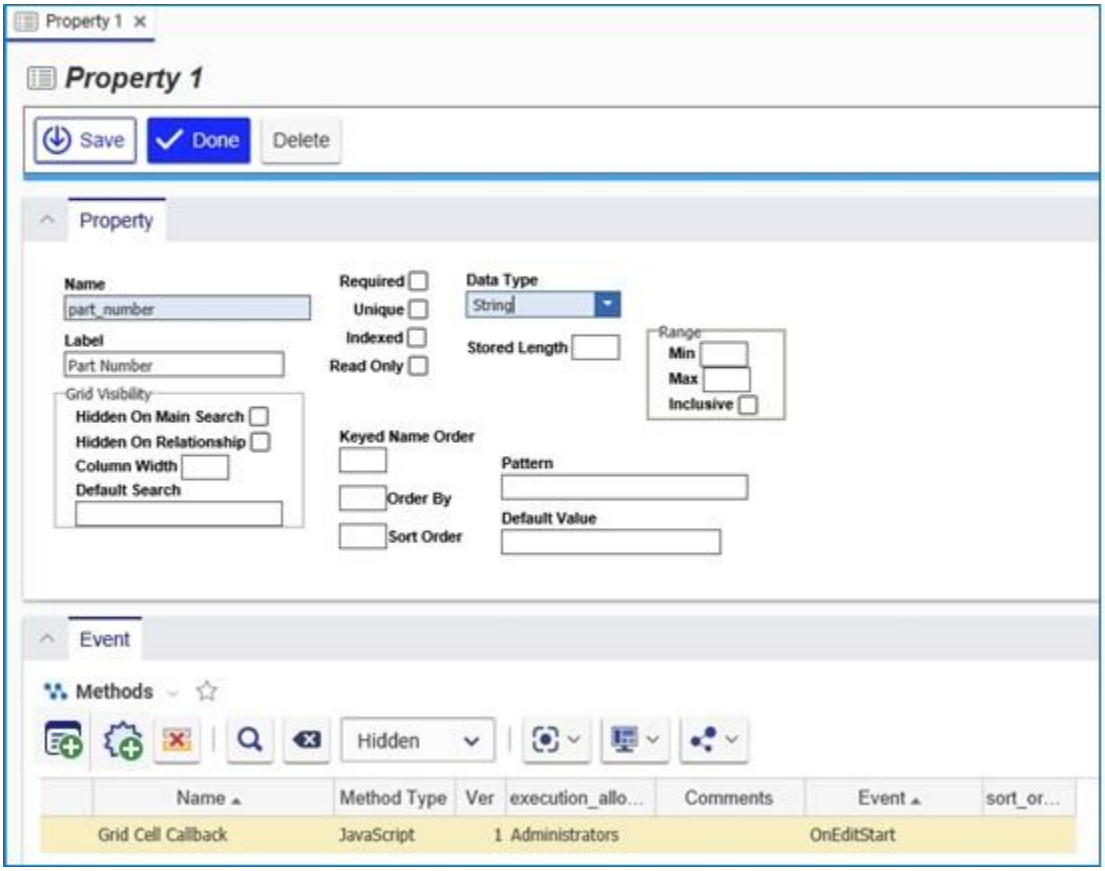
Event	Comment
onSelectRow	Fires when the row is selected.
onInsertRow	Fires when the row is inserted.
onDeleteRow	Fires when the row is deleted.

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.
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.

5.5.4.2 Cell Events

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



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.
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 may be 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

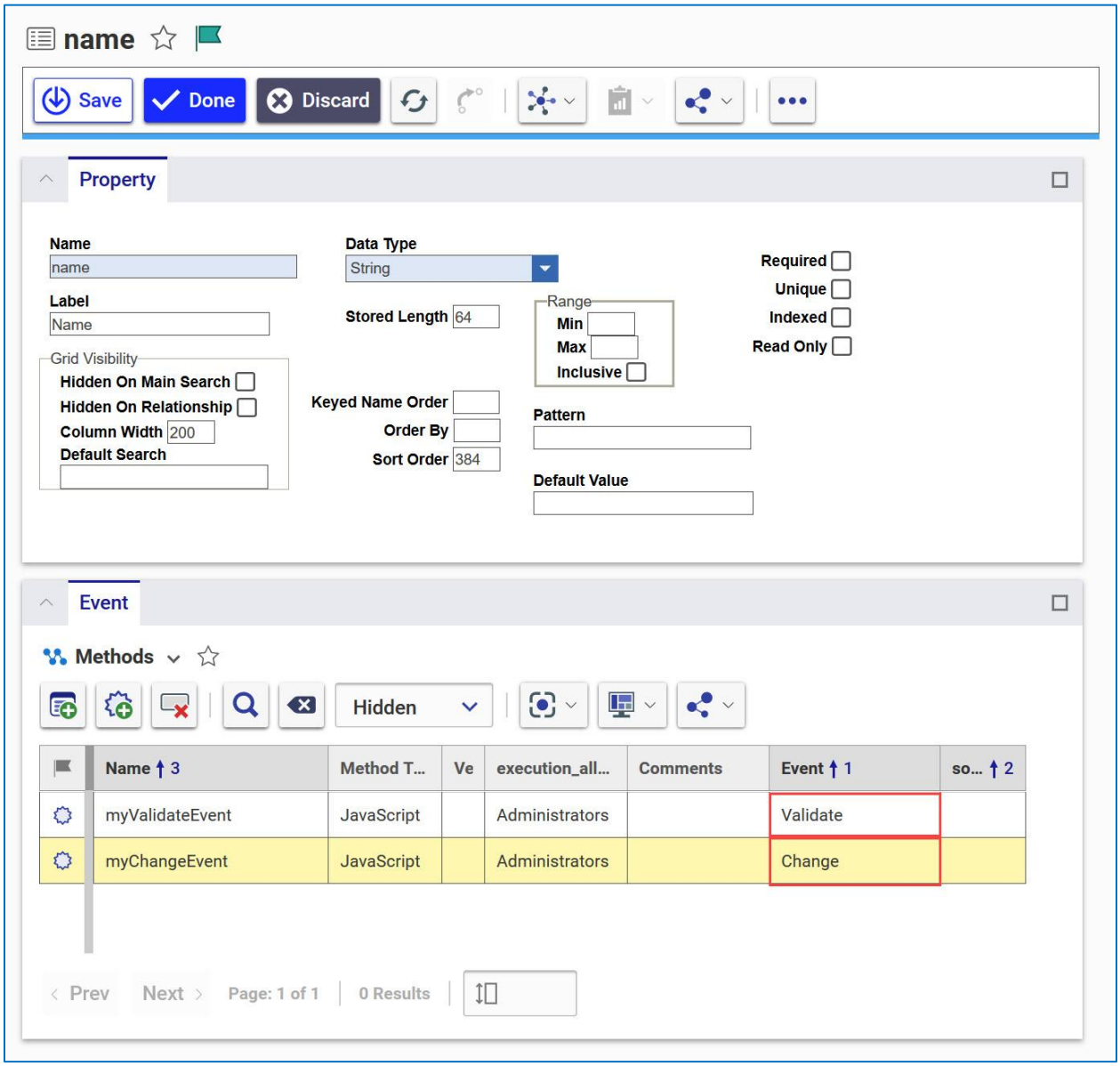
5.5.5 Data Store Events

Data Store Events are client-side events that trigger when the Aras Innovator client tries to update its internal data store. The data store is a mechanism that helps sync the state of data across the client. This means that a data store event will trigger regardless of the UI control that updates the data store. So instead of creating multiple client events for grids, forms, fields, etc., a developer can create a single event that will execute if data is updated in any of those controls.

There are 2 event types implemented for the data store.

- `Validate`: Method runs prior to updating the property in the data store. The method logic may cancel the update and return an error message to the user.
- `Change`: Method runs after a property has been updated in the data store. The method logic may manipulate the item in the data store.

Data store events are configured similarly to grid cell events – on an ItemType's properties.



It is important to note the following about data store events:

- Data store events are client-side events and do not directly update the data in the database.
- Data store events cannot directly manipulate UI controls in the client. For example, a `Validate` data store event could prevent a user from updating a property and return an error message telling them why. However, the event could not disable a grid cell or form field for the property, because those are UI controls.
- Data store events should not be configured on the properties of Poly ItemTypes. These events should be configured on the properties of the poly source ItemTypes instead.

5.5.6 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.

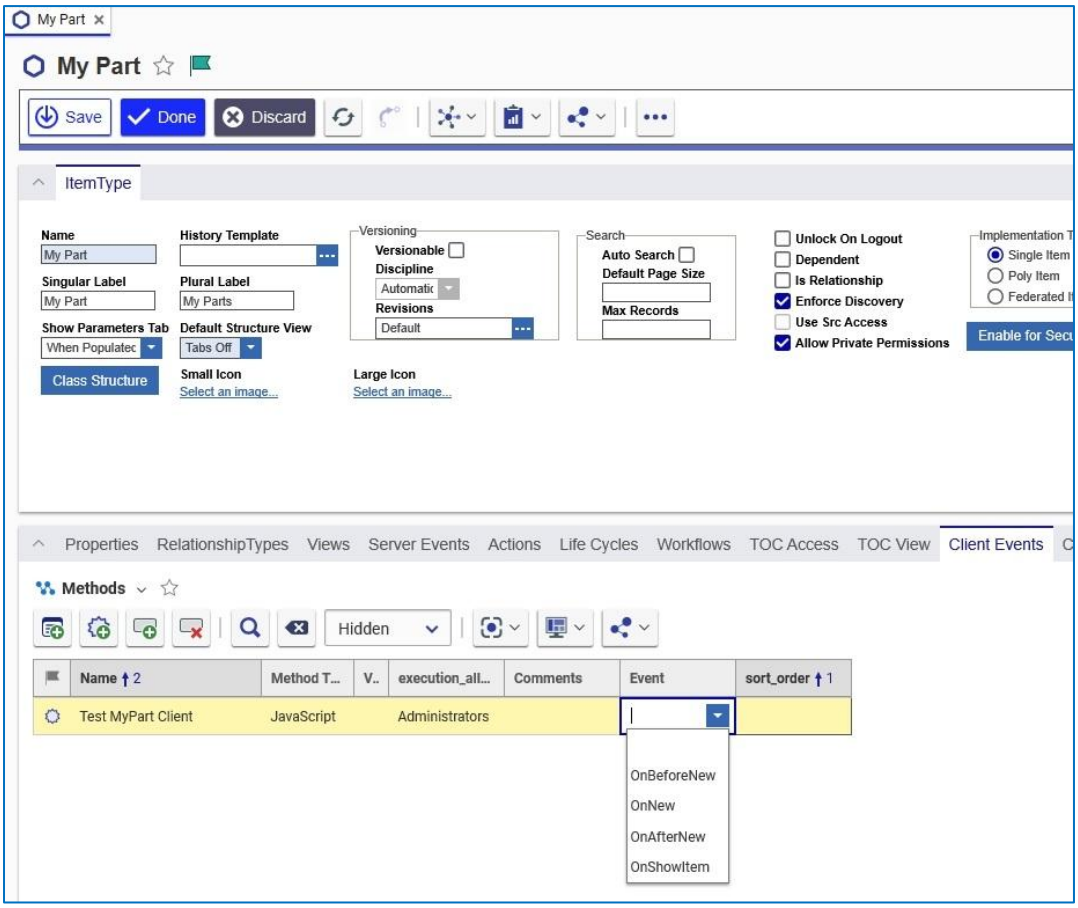
- `onBeforeNew`: Method runs prior to a new Item creation. It can cancel subsequent client operations (i.e., form opening). It can cancel creation of new Items.

  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.



5.5.7 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.

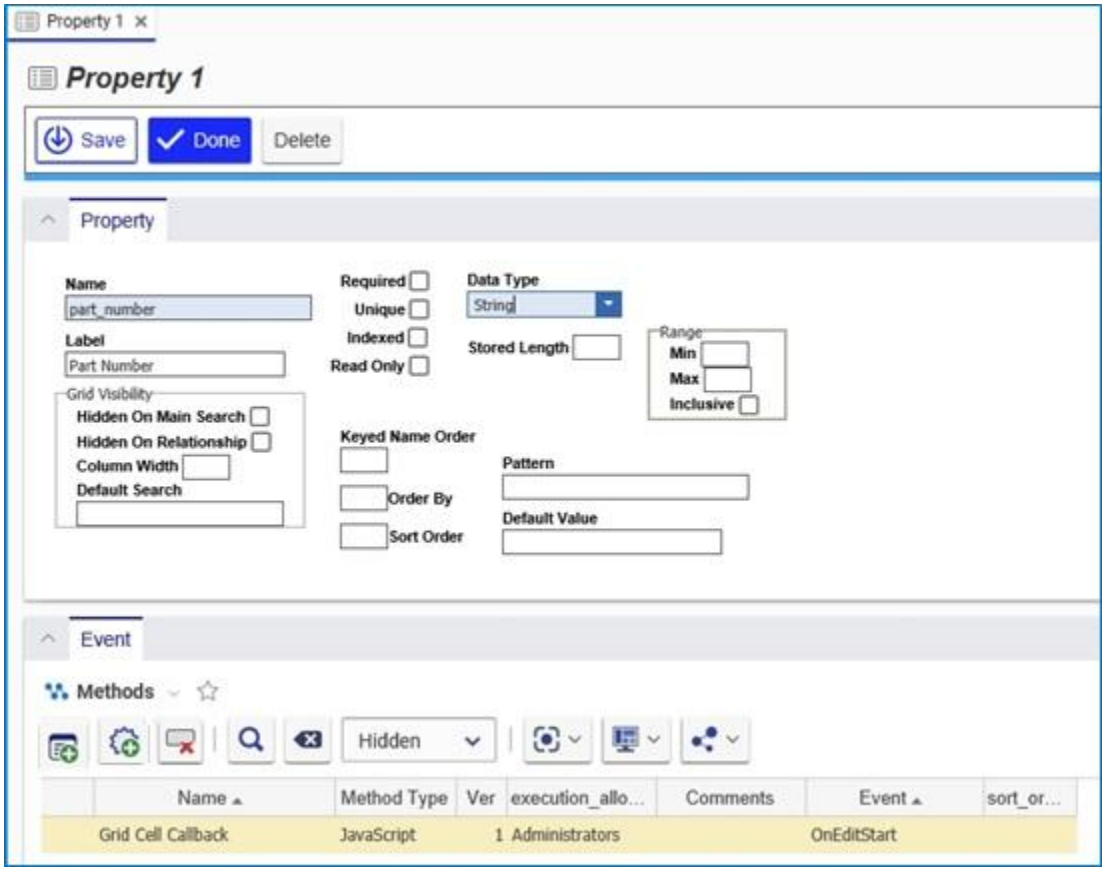
Review the Generic Methods by searching the Method Items.

6 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. Several Aras solutions use this functionality, such as QMS, MPP, and Technical Publications. The following steps 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.

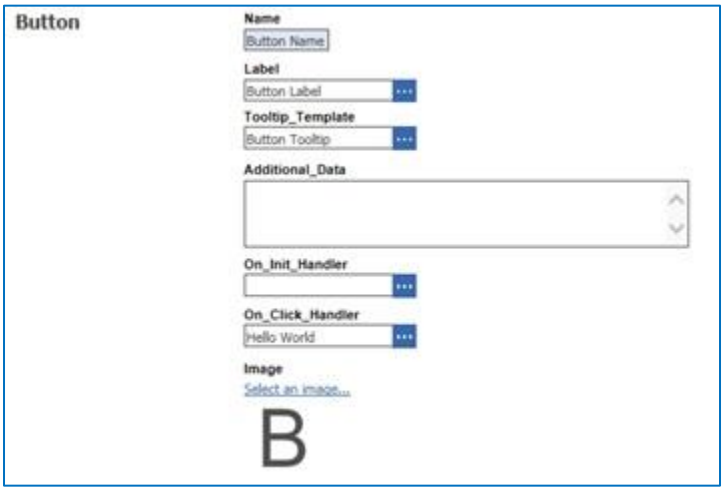
6.1 Adding a Button on a Global Scope

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



4. Lock the **Command Bar Section Item** for editing.
5. In the Command Bar Item tab, select **Create Related** from the dropdown.
6. Click **New relationship**.
7. Select **Button** in the new dialog.
8. Click **OK**. A new line will appear in the Command Bar Item tab.
9. Set **Action to Add**, and **Identity to World**.
10. Right-click and select "**View Command Bar Item**."

11. Set the values for the fields in the form.



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

6.2 Removing a Button from ItemType Scope

1. Go to the **Table of Contents: Administration > ItemTypes**.
2. Run a search for the ItemType.
3. Open the ItemType for editing.
4. In the Client Style tab, right-click on the first entry > **View "Presentation Configuration".**
5. In the Presentation Configuration, create a New Related Command Bar Section Item with Classification: Data Model.
6. In the new row, set the Identity to **World**.
7. Right-click on the new row > **View "Command Bar Section".**
8. Set the property values. In the following example, we are removing the Print button from the Tearoff Window.
   - Name: Remove Print
   - Location: Tearoff Window Toolbar
9. In the Command Bar Item tab, select **Pick Related** from the dropdown.
10. Select the Item you wish to remove. For example, run a search for '*twt_normal_print', and return the selected Item.
11. Set the Action of the new row to **Remove**.
12. Save, Unlock, and Close all items.
13. Log out and back in to see the changes.

6.3 Defining an Identity on the Button

1. Go to the Table of Contents: **Administration > Configuration > Client Presentation**.
2. Select **Global Presentation Configuration**.
3. In the Command Bar Section tab, right click on an Item with the location: **Tearoff Window Toolbar > View "Command Bar Section"**.

4. Lock the Command Bar Section for editing.
5. In the Command Bar Section tab, set your chosen Identity on the button of your choice.
6. Save, Unlock, and Close all items.
7. Log out and back in to see the resulting changes.

7 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; it is also 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 routine 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. You can 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.

8 Cookbook

This section is a Cookbook of recipes to help you solve common tasks while developing Methods. The examples are shown in JavaScript. For C# examples and general best practices for working with IOM (most of which are applicable to both JavaScript and C#), refer to the Aras IOM SDK Programmer's Guide.

8.1 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 calling the Class constructor.

JavaScript

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

8.2 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 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);
```

8.3 Query Using AML to Construct the Query Criteria

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);
}
```

8.4 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);
}
```

8.5 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;
}
```

8.6 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>
```

8.7 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;
}
```

8.8 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);
}
```

8.9 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();
```

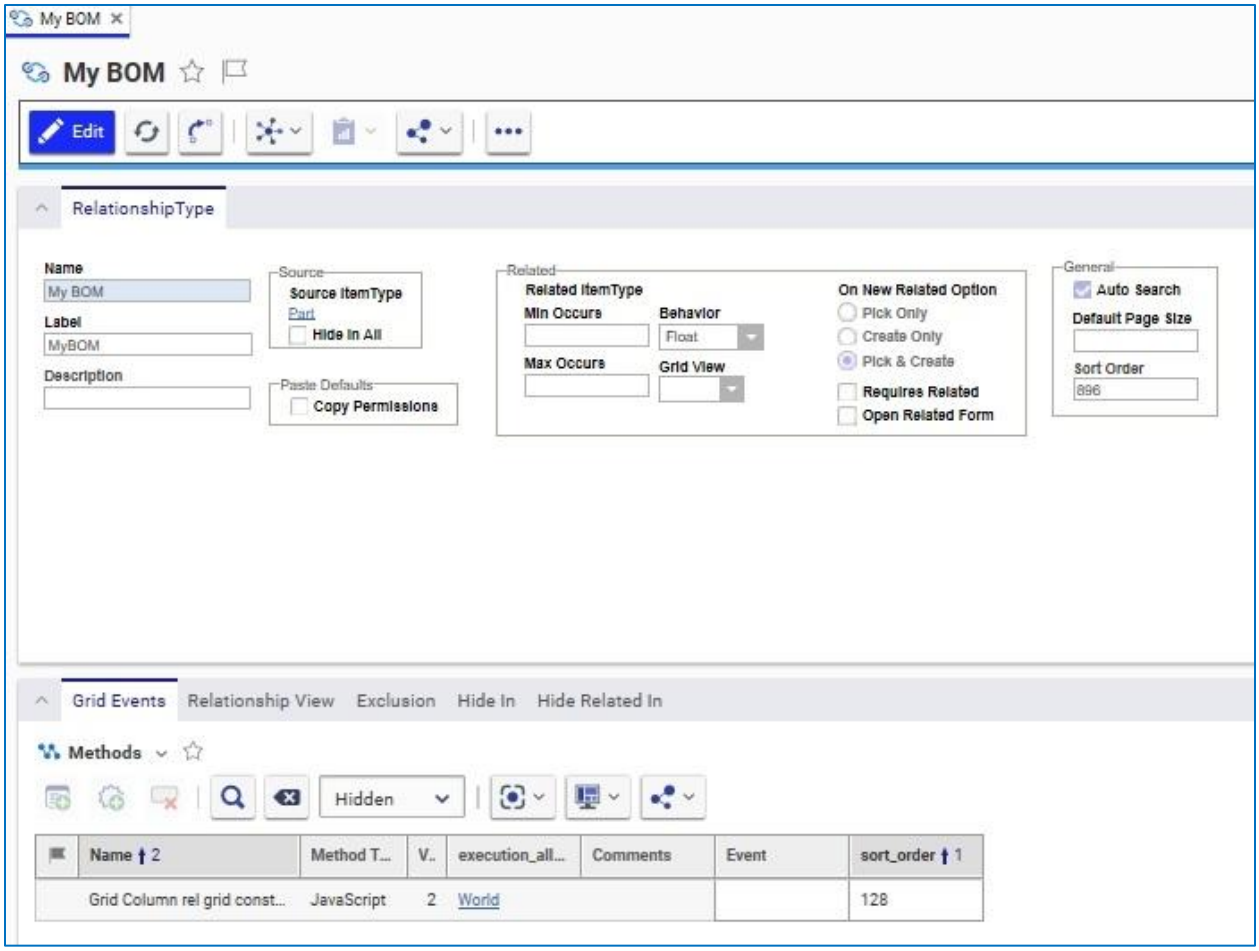
```
 if (resultItem.isError()) {
top.aras.AlertError(resultItem.getErrorDetail()); }
}

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);
}
```

8.10 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.



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");
```

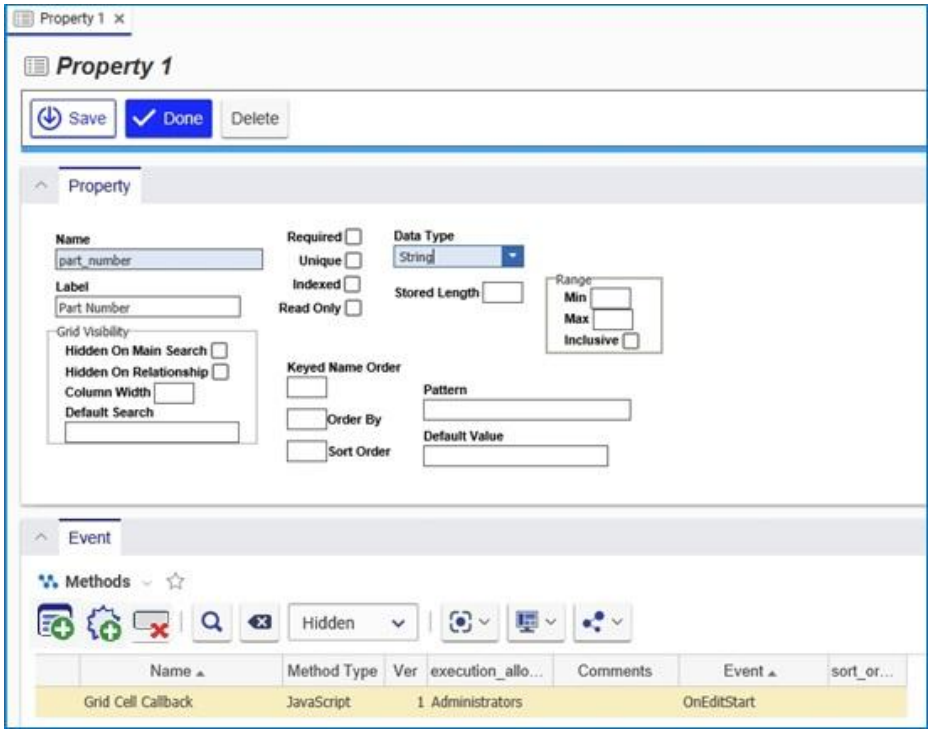
```
if (!isDirty) { gridApplet.Deselect(); }
```

8.11 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.



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;
}
```

8.12 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>
```

8.13 Want the Identities for the User

You want to get the Identity IDs for the User.

Technique

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

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(',');
```

8.14 Want a Field to Either Be 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>
```

8.15 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();
```

8.16 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

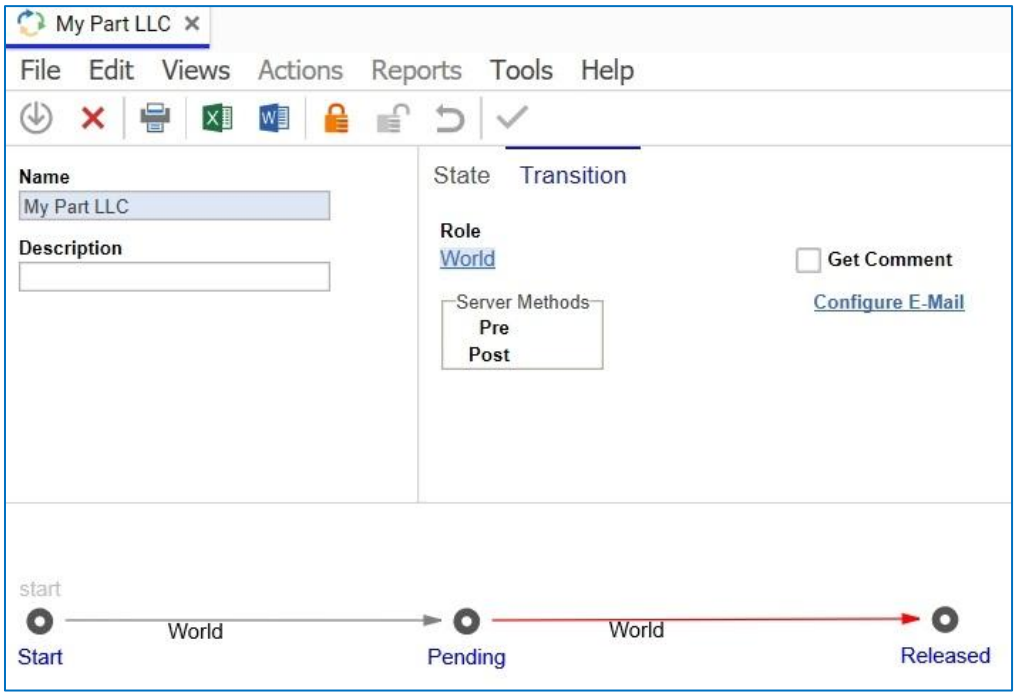
```
// Create the Document item
var docItem = this.newItem("Document","add");
docItem.setProperty("item_number","456");

// 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);
}
```

8.17 Need to Reject an Item Promote

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

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.



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;
```

8.18 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());
}
```

8.19 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;
}
```

8.20 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;
```

8.21 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.

Referencing the IOM SDK

Your DLL must reference an IOM version compatible with the Aras Innovator instance where it will run.

Since IOM SDK is now released independently from Aras Innovator, there is no longer a 1:1 mapping between IOM and Innovator versions. However, each Aras Innovator release includes and uses a specific version of the IOM SDK in its Server Methods. Your custom DLL must be compatible with the version of IOM used by the target Aras Innovator release.

How to get the correct IOM version:

By using the version of IOM.dll bundled in the /Innovator/Server/bin folder of the installed Innovator instance — this can help you identify the exact IOM SDK version used.

Best Practice for Compatibility:

- If you are targeting a single version of Aras Innovator, reference the same version of IOM SDK that comes with that Innovator release.
- If you want your DLL to support multiple Aras Innovator versions, reference the lowest IOM minor-patch version that is compatible with the earliest Aras Innovator version you intend to support. At runtime the DLL will use the version of IOM provided by the Aras Innovator.

Create the Custom DLL

To create the custom DLL, follow these steps:

1. Create a Visual Studio C# class library project targeting .NET 8.0.
2. Add a reference to the IOM.dll (see Referencing the IOM SDK for details).

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 following steps 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 the \Innovator\Server\Method-Config.xml for editing and add the highlighted line.

```
...
   <ReferencedAssemblies>
         <name>$(binpath)/Aras.Server.Core.dll</name>
         <name>$(binpath)/Aras.TDF.Base.dll</name>
         <name>$(binpath)/Aras.TDF.Base.Extensions.dll</name>
         <name>$(binpath)/Conversion.Base.dll</name>
         <name>$(binpath)/ConversionManager.dll</name>
         <name>$(binpath)/FileExchangeService.dll</name>
```

```
            <name>$(binpath)/IOM.dll</name>
            <name>$(binpath)/Newtonsoft.Json.dll</name>
            <name>$(binpath)/SixLabors.ImageSharp.dll</name>
            <name>$(binpath)/Microsoft.Data.SqlClient.dll</name>
            <name>Microsoft.Extensions.Logging.Abstractions.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.

   When adding lines, make sure you update the `line_number_offset` for accurate debugging messages.

```
        <Template name="CSharp" line_number_offset="37">
           <![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);
```

8.22 How to Add Sub Menus to Context Menu in Search Grid

There is the 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);`

The following 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");
```

8.23 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");
```

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

The purpose of the 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');
```

8.25 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>');
...
```

8.26 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');
...
```

8.27 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');
...
```

8.28 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");
...
```

8.29 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);
...
```

8.30 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', '');
...
```

8.31 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);
...
```

8.32 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');
...
```

8.33 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);
...
```

8.34 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);
   }
...
```

8.35 How to Use the "Validate" Data Store Event

The following sample code demonstrates how to prevent a user from updating a property in the data store. Validate event methods have the following parameters available:

- itemId – the GUID of the current context item
- name – the name of the property being updated
- value – the property value being validated. For item properties, this is an object with the selected item's id, type, and keyedName.

8.35.1 Example 1

Use Case: We don't want users to update prop_a anywhere in the client – on forms, relationship grids, or search grids.

Solution: Create the following method and configure it as a Validate event on prop_a of an ItemType.

```
...
   if (name === 'prop_a') {
      return {
         valid: false,
         validationMessage: `${name} property can't be changed`
      };
   }
...
```

8.35.2 Example 2

Use Case: The owner and manager of an item cannot be the same identity.

Solution: Create the following method and configure it as a Validate event on both the owned_by_id and managed_by_id properties of an ItemType.

```
...
   const currentItem = aras.itemsCache.getItem(itemId) ||
   aras.itemsCache.getItemByXPath(`//Item[@id="${itemId}"]`);
   const owner = aras.getItemProperty(currentItem, 'owned_by_id', '');
   const manager = aras.getItemProperty(currentItem, 'managed_by_id', '');

   // one or both of the fields may be null
   if (value === null) {
       return {
           valid: true
       };
   }
   if (owner === value.id || manager === value.id) {
      return {
         valid: false,
         validationMessage: `Cannot update ${name}. Owner and manager cannot
   be the same.`
      };
   }
   return {
      valid: true
   };
...
```

8.36 How to Use the "Change" Data Store Event

The following sample code demonstrates how to update a property's value in the data store. Change event methods have the following parameters available:

- itemId – the GUID of the current context item
- name – the name of the property being updated
- value – the property value being validated. For item properties, this is an object with the selected item's id, type, and keyedName.

Use Case: All Parts with the "Software" classification should have a name prefixed with "SW – ".

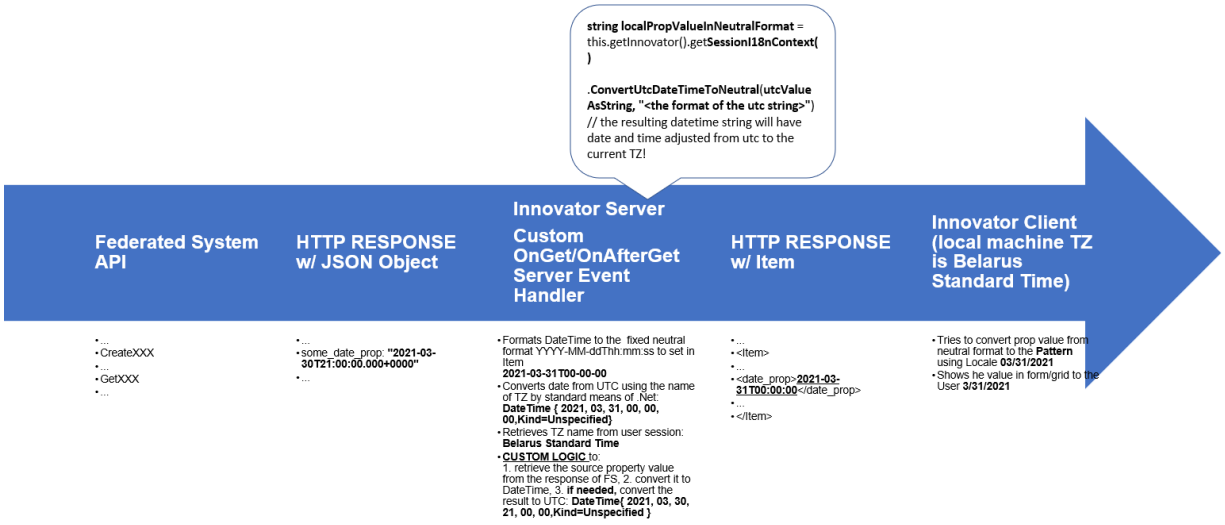
Solution: Create the following method and configure it as a Change event on the name property of the Part ItemType.

```
...
   const itemNode = aras.itemsCache.getItem(itemId) ||
   aras.itemsCache.getItemByXPath(`//Item[@id="${itemId}"]`);
   if (aras.getItemProperty(itemNode, 'classification', '') !== 'Software') {
       return;
   }

   const prefix = 'SW - ';
   if (value.substring(0,5) === prefix) {
      return;
   }

   aras.setItemProperty(itemNode, name, prefix + value);
...
```

8.37 How to Use Different Data Types for Federated Properties

Federated property is used for Item types to access data from an external system instead of the Aras Innovator database. It can be configured via turning the Federated flag on, but not all data types are supported. Currenty the following data types can be used for Federated properties:

- **Integer**.
- **Text**.
- **Date**. It is expected that the date value is set in Item according to the neutral format of Innovator yyyy-MM-ddTHH:mm:ss (or yyyy-MM-dd) in the current TZ in local or Corporate TimeZone (if Corporate TimeZone is set). Otherwise, the correct work of federated dates with other logic of Innovator (e.g., presentation in the Client) cannot be guaranteed. The customization developers should consider the potential differences in the date format and TimeZones of Innovator Server and a Federated System and implement corresponding adjustments before setting the value to Innovator Items (especially in cases when time part is important). As recommendation: custom approach to convert dates from the TZ of Federated System to UTC can be implemented and then call IOM method *ConvertUtcDateTimeToNeutral* which will produce the result in a proper format and TZ for Innovator.



And in the opposite direction when data should be sent from Innovator to Federated System, IOM method *ConvertNeutralToUtcDateTime* will provide the datetime in UTC and then the customization logic (if needed) will adjust it to the TZ of Federated System (see page "Data Flow-2a-FS-NoCorpTZ" in the attached excel file "Date - Pattern and TZ" for illustrations of data flows)

Propery setting **Pattern (pattern)** for date properties is used only in Innovator Client to present the date value for the end user with the order of display for month, day and year following internationalization and localization rules dependent on client settings.

- **Float**. IOM means *i18NSessionContext.ConvertToNeutral* can be used to convert strings with float values to neutral format to prepare float string in proper format if needed. Please do not specify Precision and Scale for federated Float properties to avoid inconsistent restrictions on the client side.
- **String.** For regular String properties there are two special settings in Innovator:
  - **Length (stored_length) –** required, the maximum length of the string property
  - **Pattern (pattern) –** optional, is used to prepare a Regex (ignore-case, single line) which the property value should match

It's up to the Administrator/Developer creating the federated String property to set it up in Innovator according to the specification and requirements of the Federated System. Innovator's Client will provide built-in validation based on these settings for bi-directional data exchange. As for additional validation in Innovator Server, the Developers can implement it themselves in OnBeforeAdd/Update or OnAdd/Update event handlers before sending the data with the string property value to the Federated System.If a federated property is supposed just to show information from the Federated System and no input is expected on Innovator side, then set Readonly setting for the federated property to true.

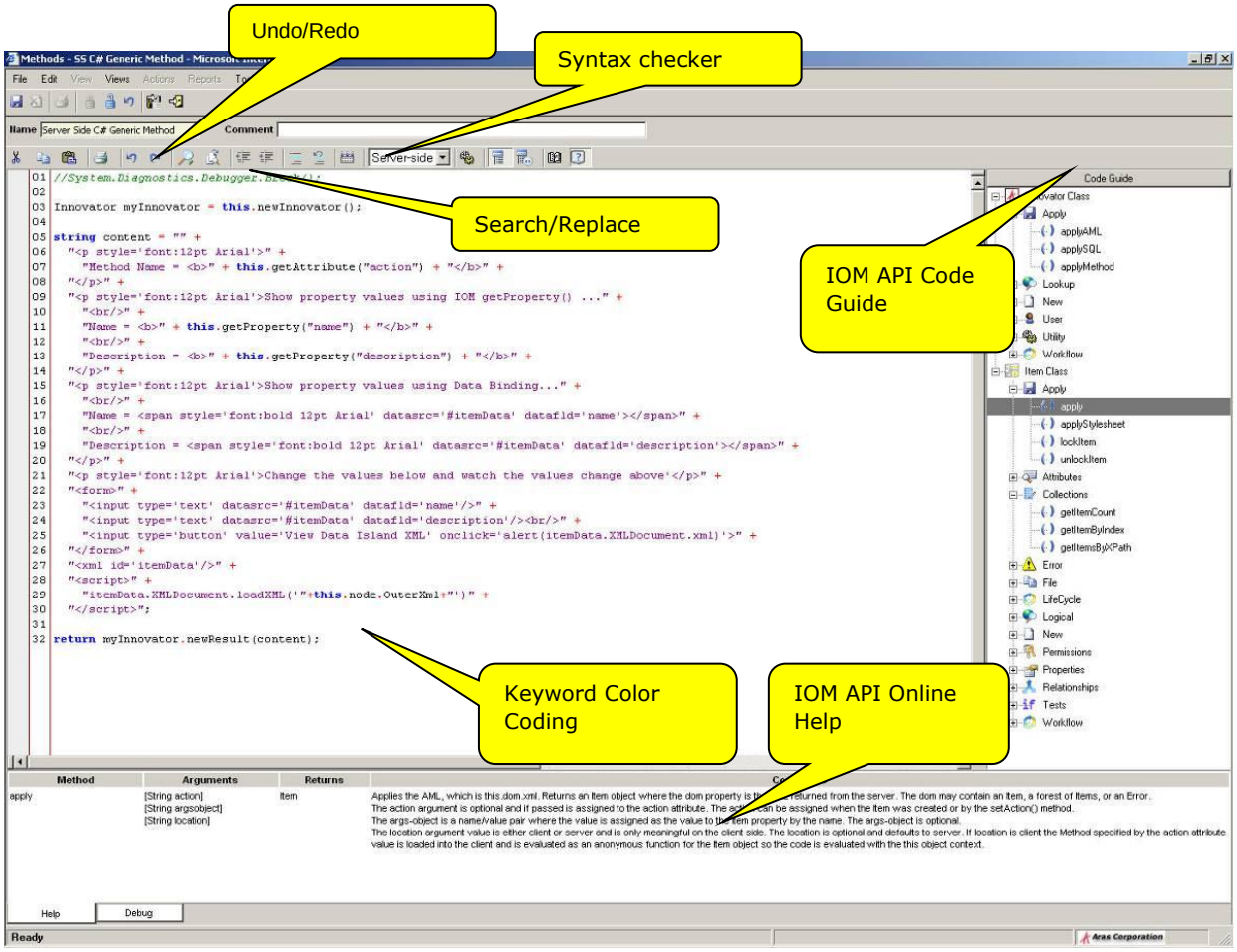
9 Aras Innovator Solution Studio

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.

9.1 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.



10 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.

10.1 Enabling Debugging in Aras Innovator

Server method debugging is disabled by default in Aras Innovator 30. 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.

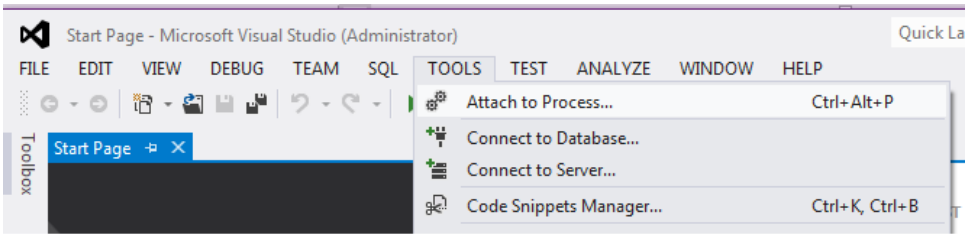
```
<operating_parameter key="DebugServerMethod" value="true" />
```

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

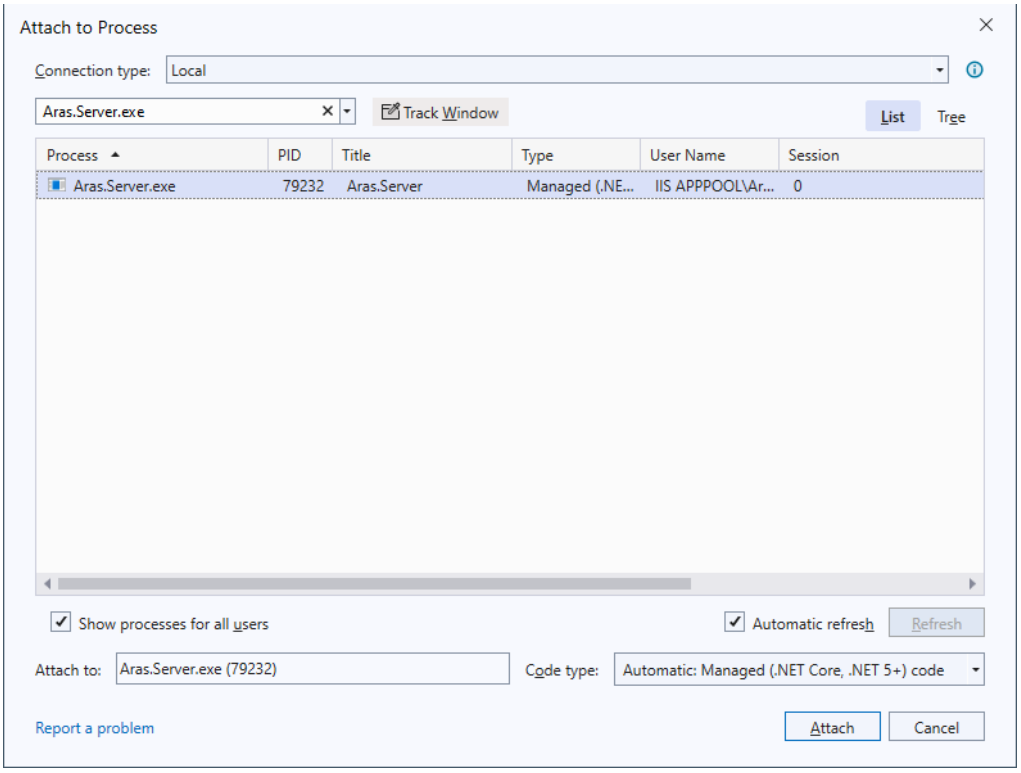
After changing this value, you should restart IIS to confirm the change is applied.

10.2 Setting up Visual Studio to Debug Server-Side Methods

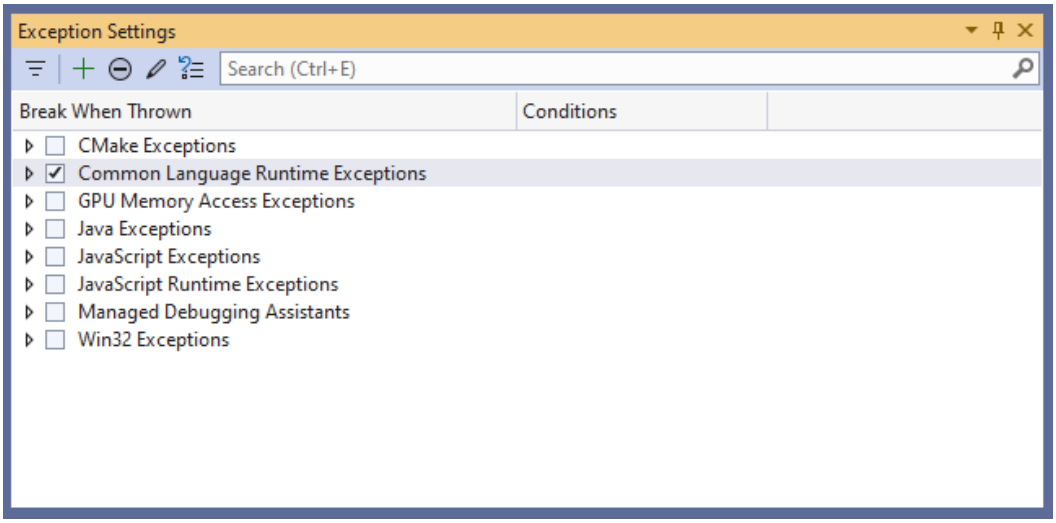
1. Open "Microsoft Visual Studio .NET" debugger.
2. Choose **Tools --> Attach to Processes…**.



3. Choose `Aras.Server.exe` and click **'Attach'**.



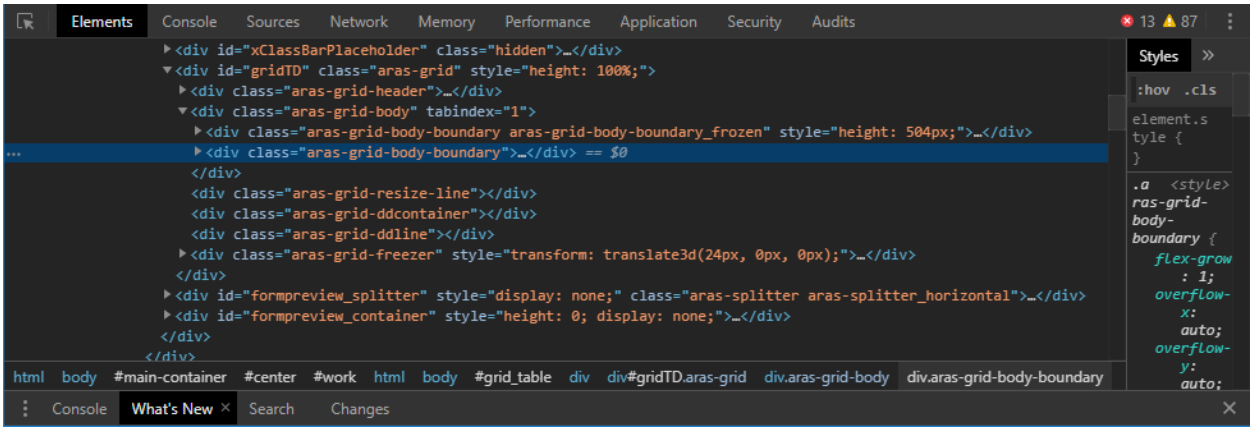
4. Choose **Debug -->Windows-> Exception Settings** to force the method into the debugger when an exception occurs and choose the check boxes "Thrown".



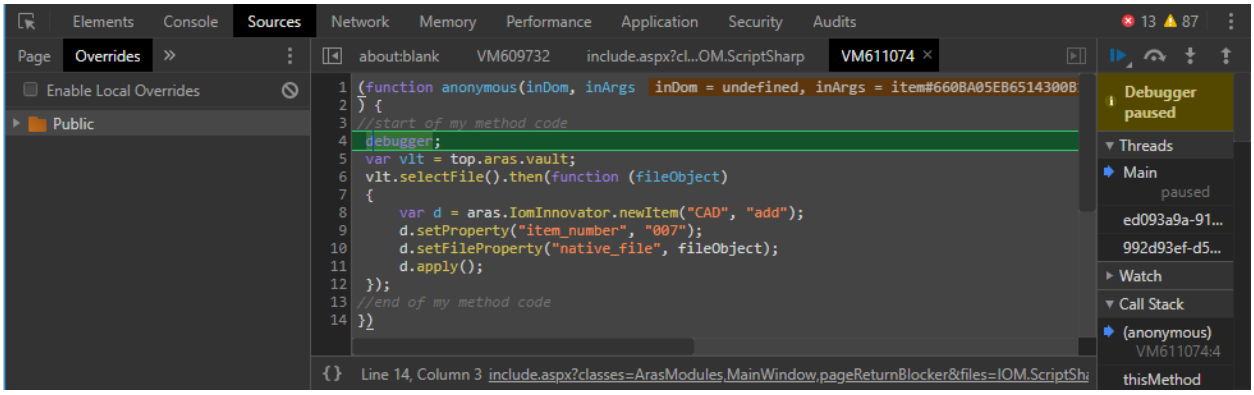
5. Optionally, you can include the line `System.Diagnostics.Debugger.Break()` to force a breakpoint at a specific line in your method.

10.3 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.



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



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.

10.4 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" />
```

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");
```