ACE 2024

STUDENT TRAINING GUIDE

Using Federated Properties

aras

Revision MARCH 2024

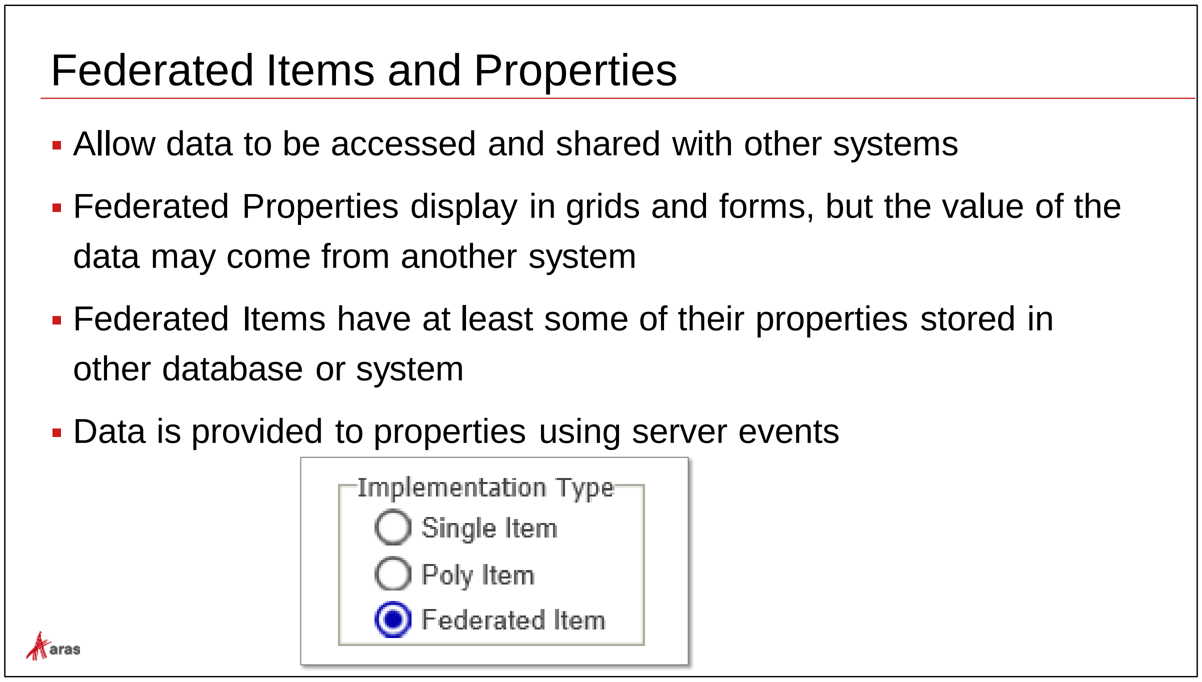
Using Federated Properties

Overview

In this session, we will discuss how to manipulate data kept in an external system, while still presenting the information to the user in the standard client user interface. Federation allows you to integrate an Aras Innovator solution with data that does not reside in the Innovator database.

Objectives

- Review Federated Items and Properties.
- Discuss Federated Data Sources.
- Configure a Federated Property or ItemType.
- Perform CRUD operations with Federated Property values.



Federated Items and Properties

- Allow data to be accessed and shared with other systems
- Federated Properties display in grids and forms, but the value of the data may come from another system
- Federated Items have at least some of their properties stored in other database or system
- Data is provided to properties using server events

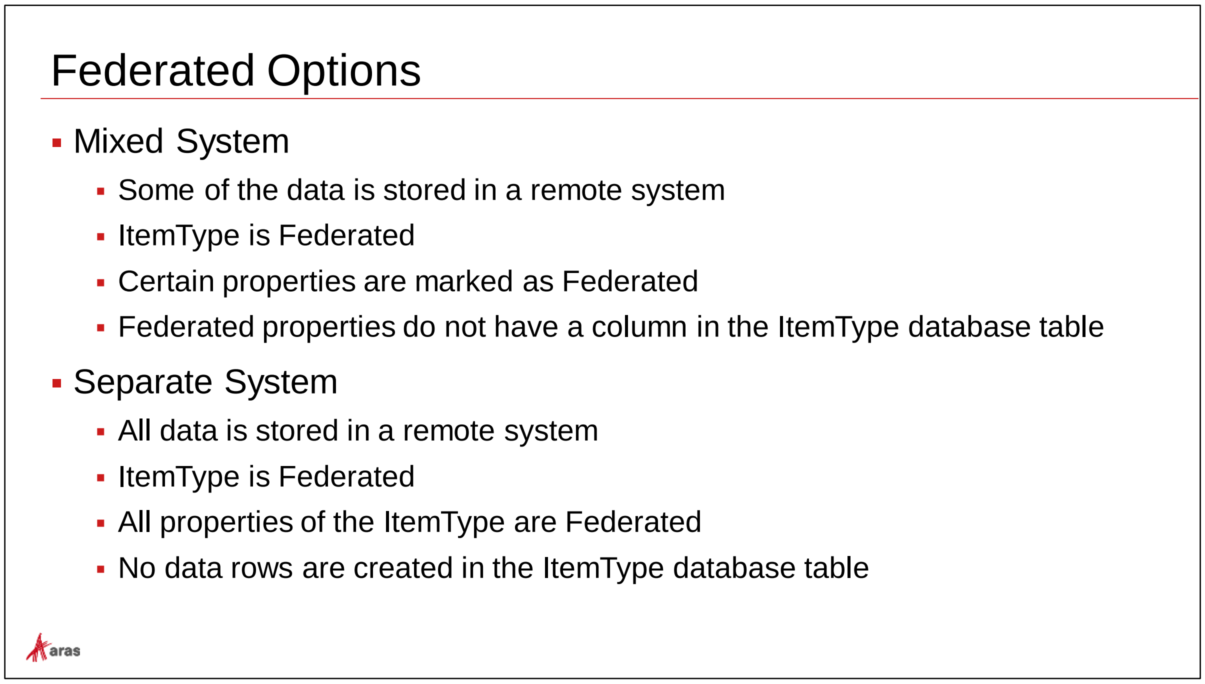
Implementation Type
Single Item
Poly Item
[✓ Federated Item]

Federated Items and Properties

Federated Items and Properties are useful when integrating data from other systems into an Aras Innovator solution.

When you mark an ItemType as Federated, you are indicating that at least one property of the Item will not be stored in the Innovator database. Keep in mind that an ItemType that is created in Aras Innovator is represented by a table in the database and each of its property is represented by a column. Columns that would have represented Federated properties are dropped from the database.

With Federation, you can seamlessly perform all CRUD operations (Create, Read, Update and Delete) within the Aras Innovator solution.



Federated Options

- Mixed System
  - Some of the data is stored in a remote system
  - ItemType is Federated
  - Certain properties are marked as Federated
  - Federated properties do not have a column in the ItemType database table
- Separate System
  - All data is stored in a remote system
  - ItemType is Federated
  - All properties of the ItemType are Federated
  - No data rows are created in the ItemType database table

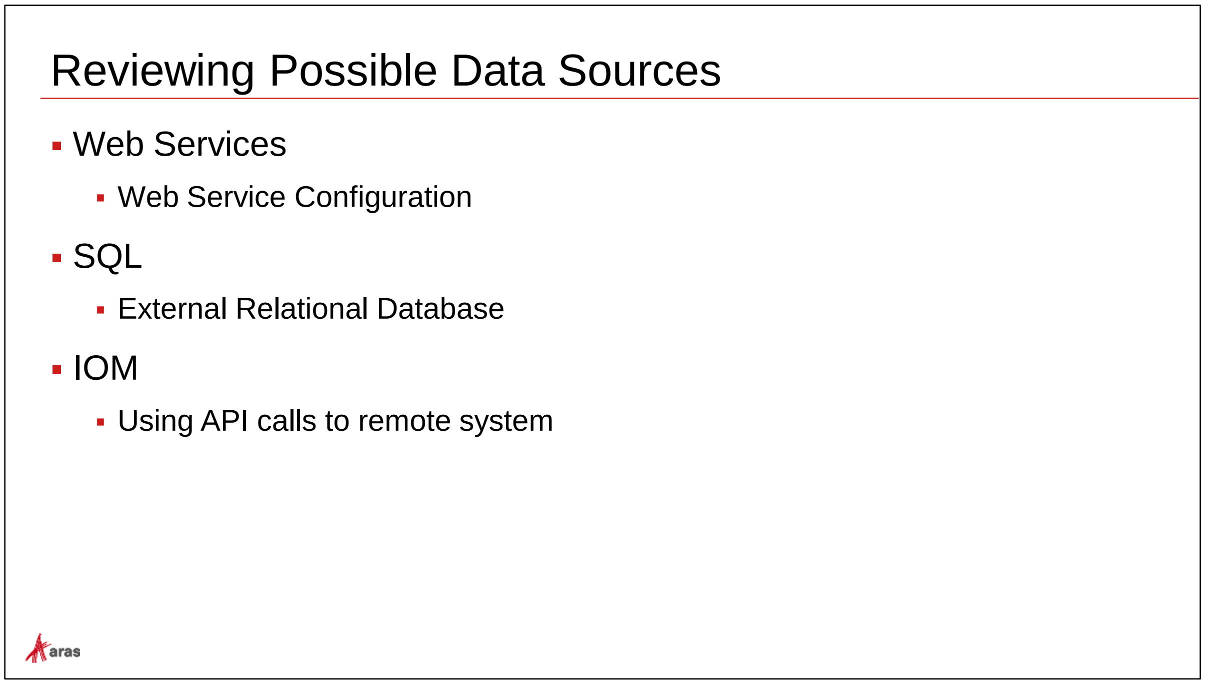
aras

Federated Options

You may have two scenarios when implementing Federation to an Aras solution. Some elements of the data reside in the Aras Innovator database, or all elements of the data are kept in an external repository.

In what could be called a “separate system”, for training purposes, no rows (Items) are created in the Aras Innovator database and all data must be brought into the system using custom logic. All system properties still exist but are not used.

On the other hand, in a so-called “mixed system”, some elements of the dataset, such as the item identification number, system properties such as “**created_on**”, “**created_by**” may be stored in the Aras Innovator database, while some other elements may reside in an external repository.



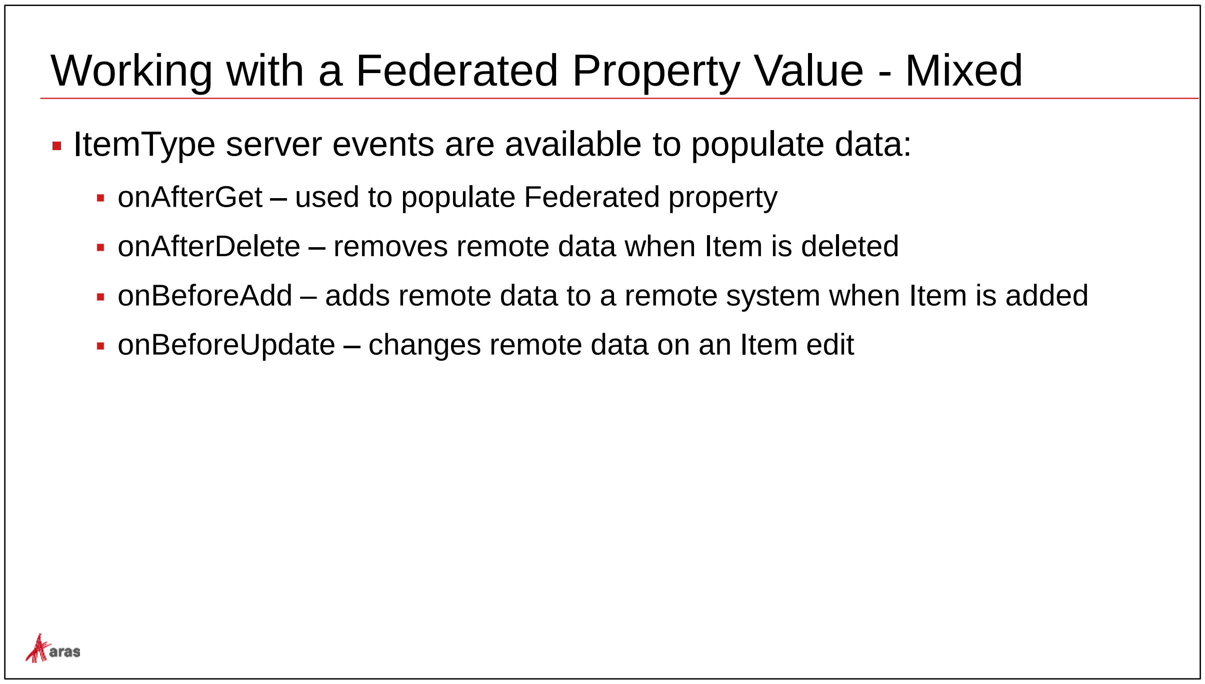
Reviewing Possible Data Sources

- Web Services
  - Web Service Configuration
- SQL
  - External Relational Database
- IOM
  - Using API calls to remote system

Reviewing Possible Data Sources

Several different types of implementations may be effective to establish the connection between the external system and Aras Innovator server. If a generalization could be done, it would be safe to say that in all cases, we would need to establish some sort of “connector”, which will contain the basic elements for the external system to authenticate and establish a session, with the associated configured permissions allocated to the connecting identity. This connector would also contain the URL of the server supporting such handshake, as well as the database to be used in the configured integration.

For our hands-on exercise in this session, we will use an external repository to retrieve and update data using flat files. Server events will be used to trigger the execution of server methods to properly manipulate the desired dataset, according to the implemented business requirements.



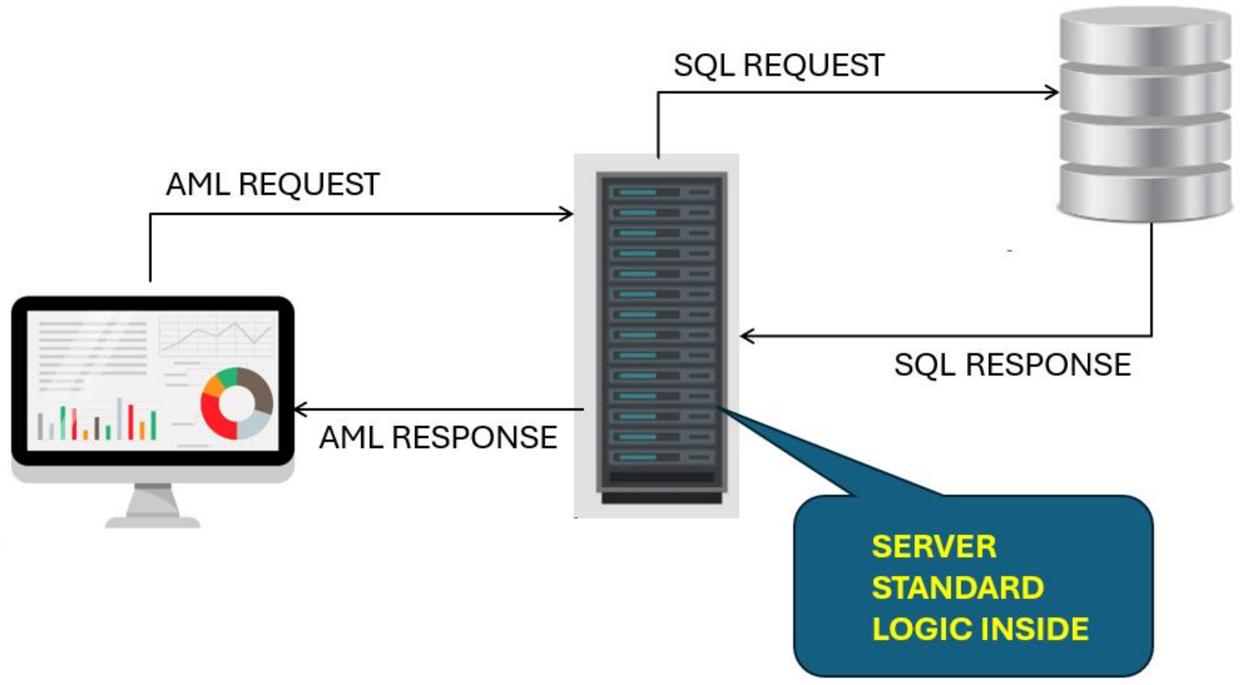
Working with a Federated Property Value - Mixed

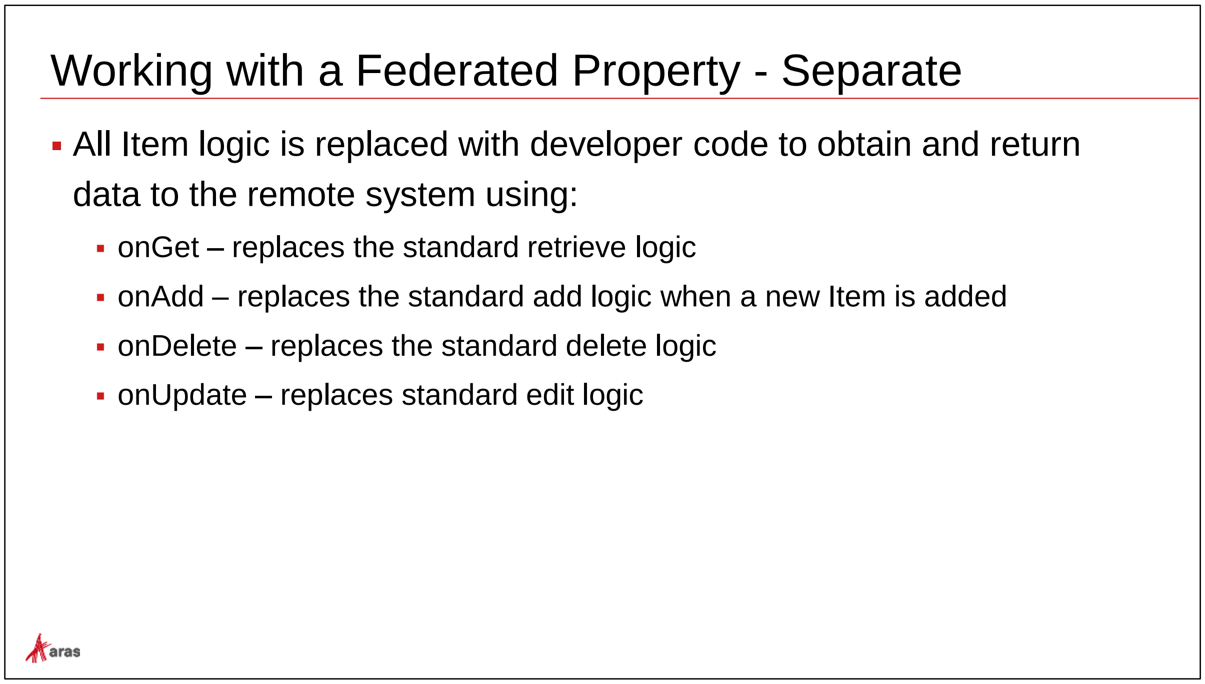
- ItemType server events are available to populate data:
  - onAfterGet – used to populate Federated property
  - onAfterDelete – removes remote data when Item is deleted
  - onBeforeAdd – adds remote data to a remote system when Item is added
  - onBeforeUpdate – changes remote data on an Item edit

Federated Property Value – Mixed

To support Federation, we will manipulate the CRUD operations associated to the Item dataset using a set of server events on the ItemType. These events allow you to intercept the AML messages being passed to and from the Innovator Server whenever creating, retrieving, updating, or deleting external data.

Reviewing Server Events





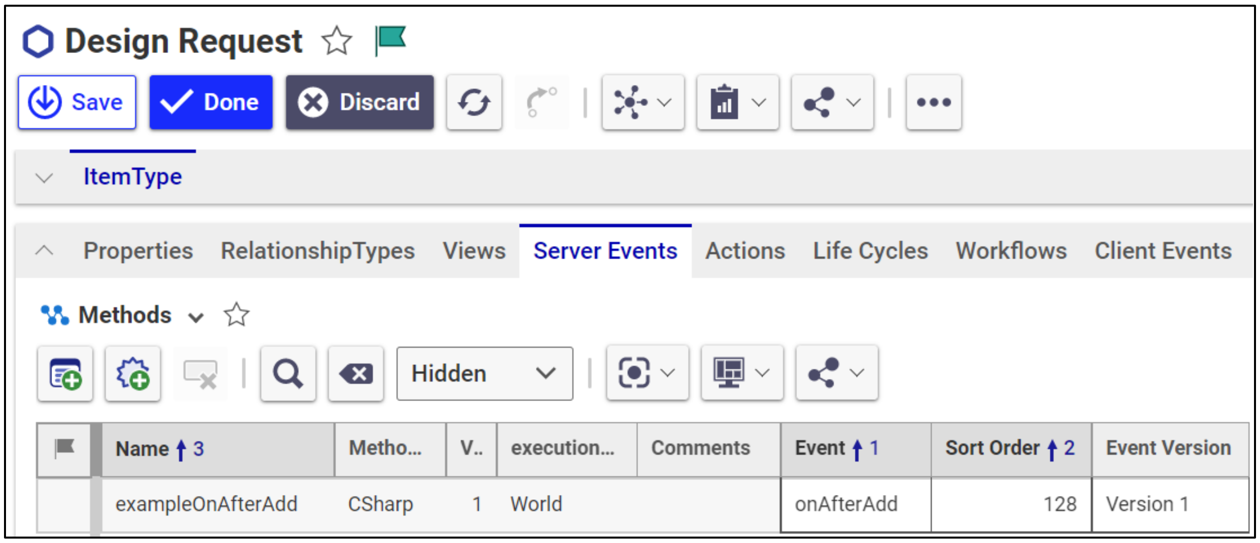
Working with a Federated Property - Separate

- All Item logic is replaced with developer code to obtain and return data to the remote system using:
  - onGet – replaces the standard retrieve logic
  - onAdd – replaces the standard add logic when a new Item is added
  - onDelete – replaces the standard delete logic
  - onUpdate – replaces standard edit logic

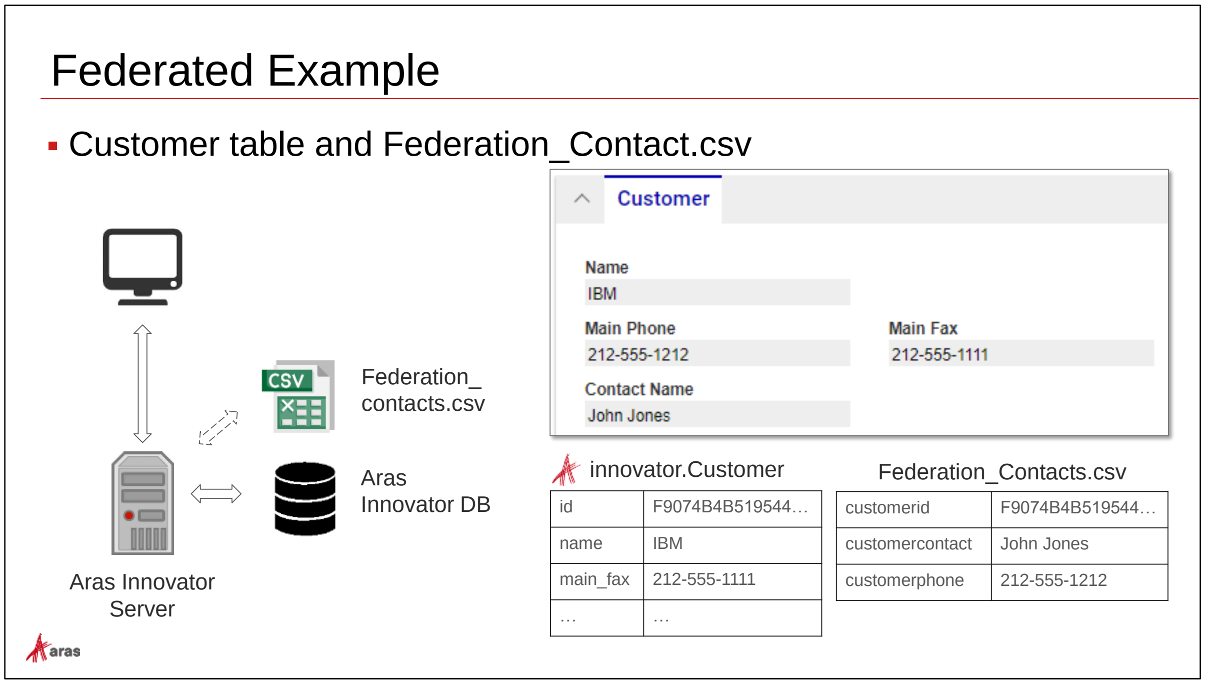
Federated Property Value – Separate

If the entire Item dataset is not stored in the Aras Innovator database, you will then replace the current server logic for CRUD operations with your own customized code. These server events will completely replace the Add, Edit, Get and Delete standard behavior in the Aras Innovator system.

Linking Server Events to an ItemType



Name 3	Metho...	V..	execution...	Comments	Event 1	Sort Order 2	Event Version
exampleOnAfterAdd	CSharp	1	World		onAfterAdd	128	Version 1



Federated Example

- Customer table and Federation_Contact.csv

Customer

Name
IBM

Main Phone
212-555-1212

Main Fax
212-555-1111

Contact Name
John Jones

innovator.Customer

id	F9074B4B519544…
name	IBM
main_fax	212-555-1111
…	…

Federation_Contacts.csv

customerid	F9074B4B519544…
customercontact	John Jones
customerphone	212-555-1212

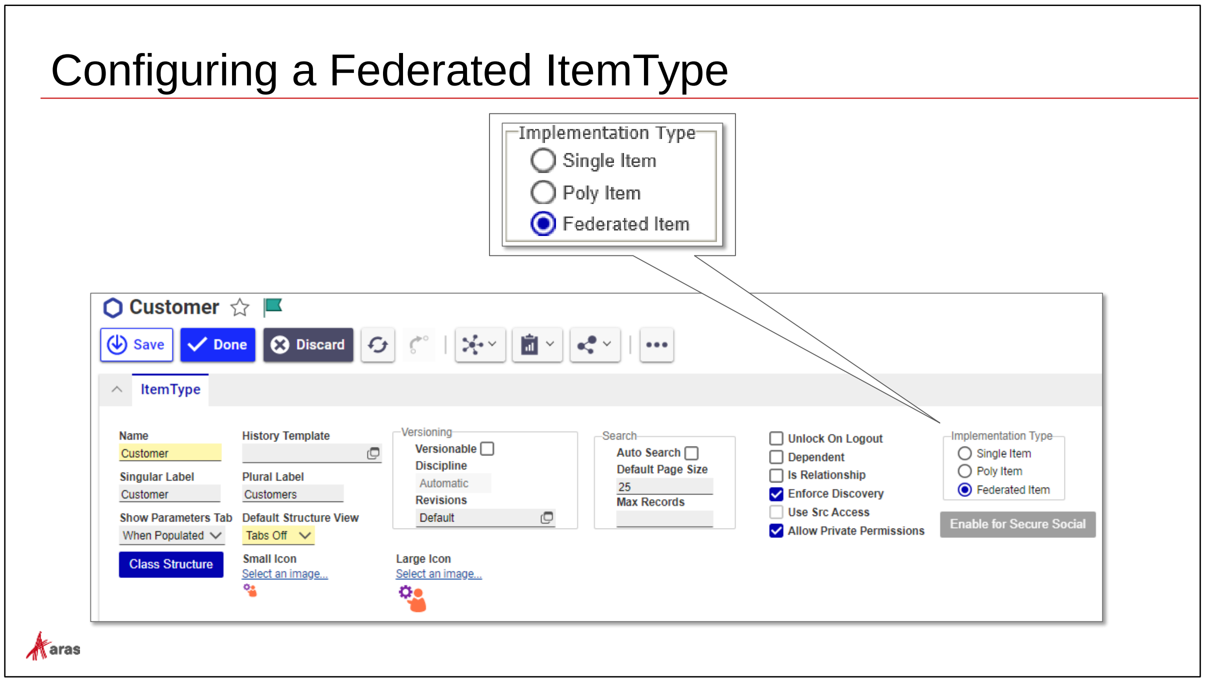
Federated Example (Federated Customer ItemType)

The Aras Innovator solution implemented for this session contains an ItemType called “**Federated Customer**” configured in a way that the values for both “**contact name**” and “**phone number**” properties will be stored in an external repository, representing the integration with another system. For this exercise, we will consider that the external system periodically updates this data with Aras Innovator.

The “**Federated Customer**” ItemType is implemented as a "mixed" Federated Item with its associated dataset stored in a flat csv-file named “**pc_CRM_customers_repository.csv**”. The “**customerid**“ property will be used to synchronize the external repository dataset with the data in the Aras Innovator solution. The fields “**customername**” and “**customerphone**” will store the federated property values of the “**Federated Customer**” ItemType in our example.

The mapping between the property name of the “Federated Customer” ItemType in Aras Innovator and its representation in the external repository is shown below:

Federated Customer property name	**column name in csv-file**
id	customerid
contact_name	customername
main_phone	customerphone



Configuring a Federated ItemType

Implementation Type
Single Item
Poly Item
[✓ Federated Item]

Customer

Name: Customer
History Template:
Singular Label: Customer
Plural Label: Customers
Show Parameters Tab: When Populated
Default Structure View: Tabs Off
Small Icon: Select an image...
Large Icon: Select an image...

Versioning
Versionable ☐
Discipline: Automatic
Revisions: Default

Search
Auto Search ☐
Default Page Size: 25
Max Records:

☐ Unlock On Logout
☐ Dependent
☐ Is Relationship
☑ Enforce Discovery
☐ Use Src Access
☑ Allow Private Permissions

Implementation Type
Single Item
Poly Item
[✓ Federated Item]

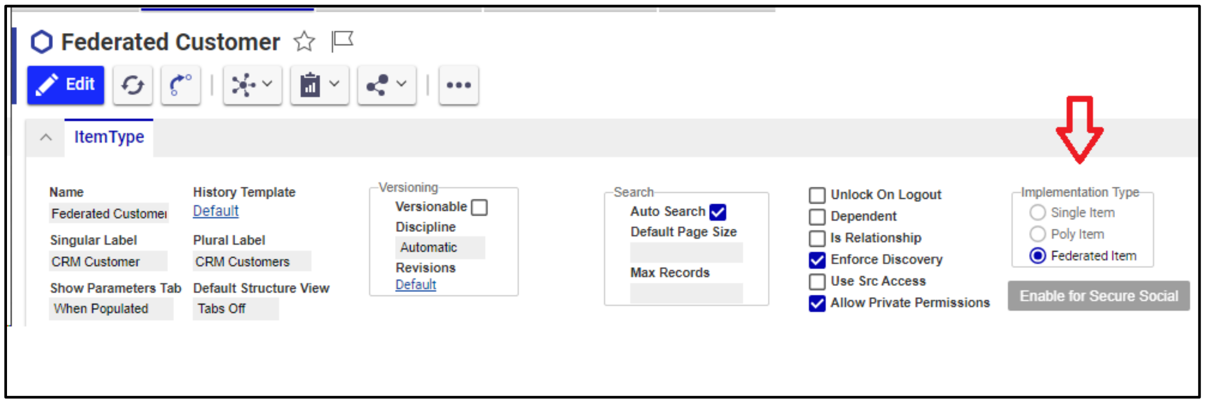
Configuring a Federated ItemType

A Federated Item is represented in the Innovator Solutions configuration by selecting the "Federated Item" radio button in the Implementation Type group, as shown above.

Selecting this option affects system relationships behavior and indicates that some or all the data for the items will be stored in an external repository. Methods associated to server events of the ItemType will help us handling the implementation of Federation on this ItemType.

Try it:

1. Open the "Federated Customer" ItemType and verify the Implementation Type



Federated Customer

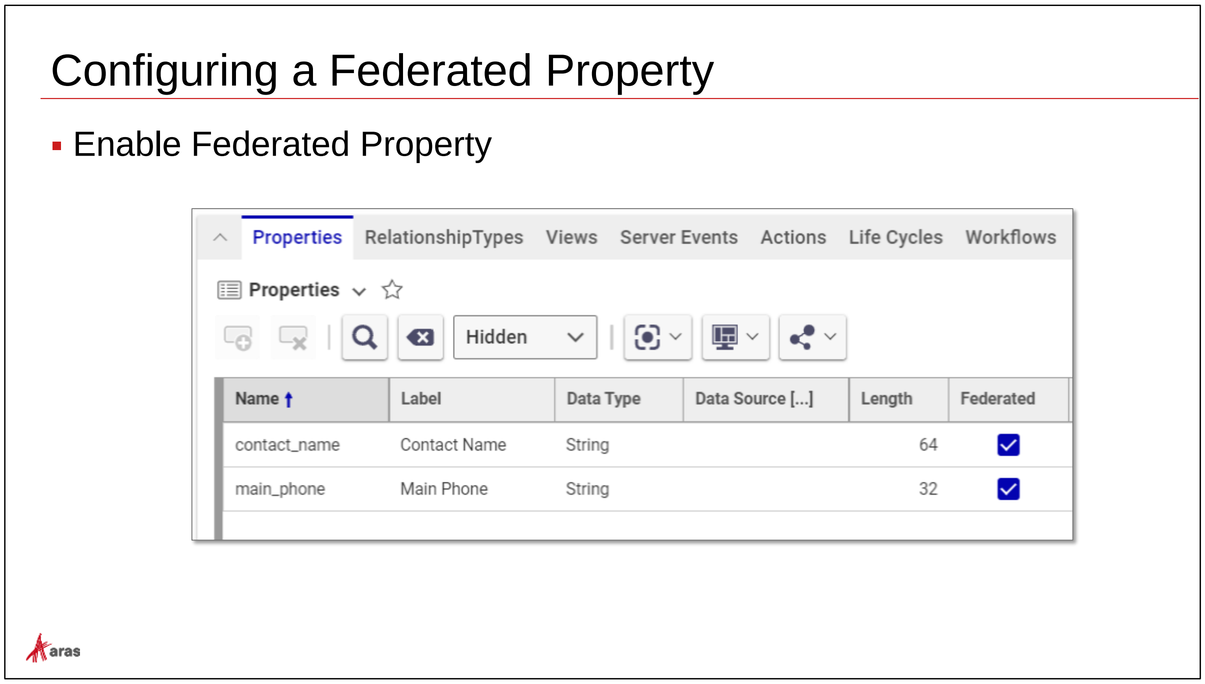
Name: Federated Customer
History Template: Default
Singular Label: CRM Customer
Plural Label: CRM Customers
Show Parameters Tab: When Populated
Default Structure View: Tabs Off

Versioning
Versionable ☐
Discipline: Automatic
Revisions: Default

Search
Auto Search ☑
Default Page Size:
Max Records:

☐ Unlock On Logout
☐ Dependent
☐ Is Relationship
☑ Enforce Discovery
☐ Use Src Access
☑ Allow Private Permissions

Implementation Type
Single Item
Poly Item
[✓ Federated Item]



Configuring a Federated Property

- Enable Federated Property

Properties | RelationshipTypes | Views | Server Events | Actions | Life Cycles | Workflows

Properties

Hidden

Name	Label	Data Type	Data Source [...]	Length	Federated
contact_name	Contact Name	String		64	☑
main_phone	Main Phone	String		32	☑

Configuring a Federated Property

Enable Federation at the property level using the "Federated" checkbox for each property containing a value that is kept away from the Innovator database.

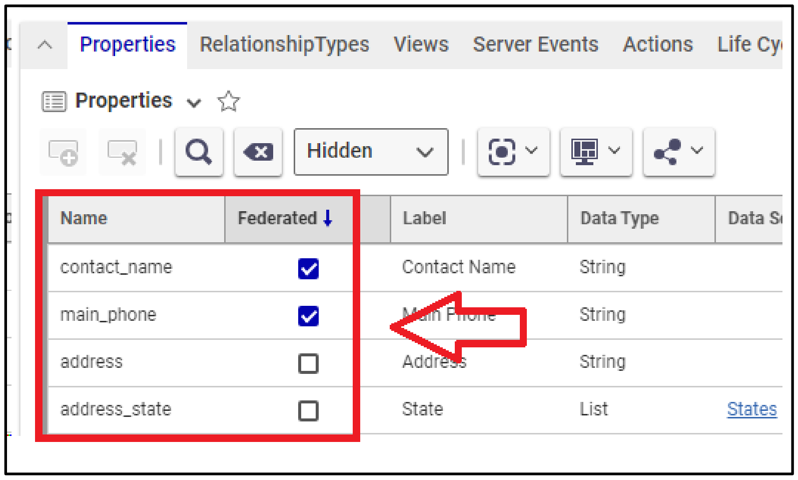
In this example, the "**contact_name**" and "**main_phone**" properties would no longer be stored in the Aras Innovator database.

Warning about setting a property/item as Federated

- Choosing this option will drop the database columns for these marked properties from the "Federated Customer" ItemType table in the Aras Innovator database as soon as you save the changes.
- Note the dropped columns on your local SQL Server.

Try it:

1. Note that the ItemType is configured with both Federated and Non-Federated Properties

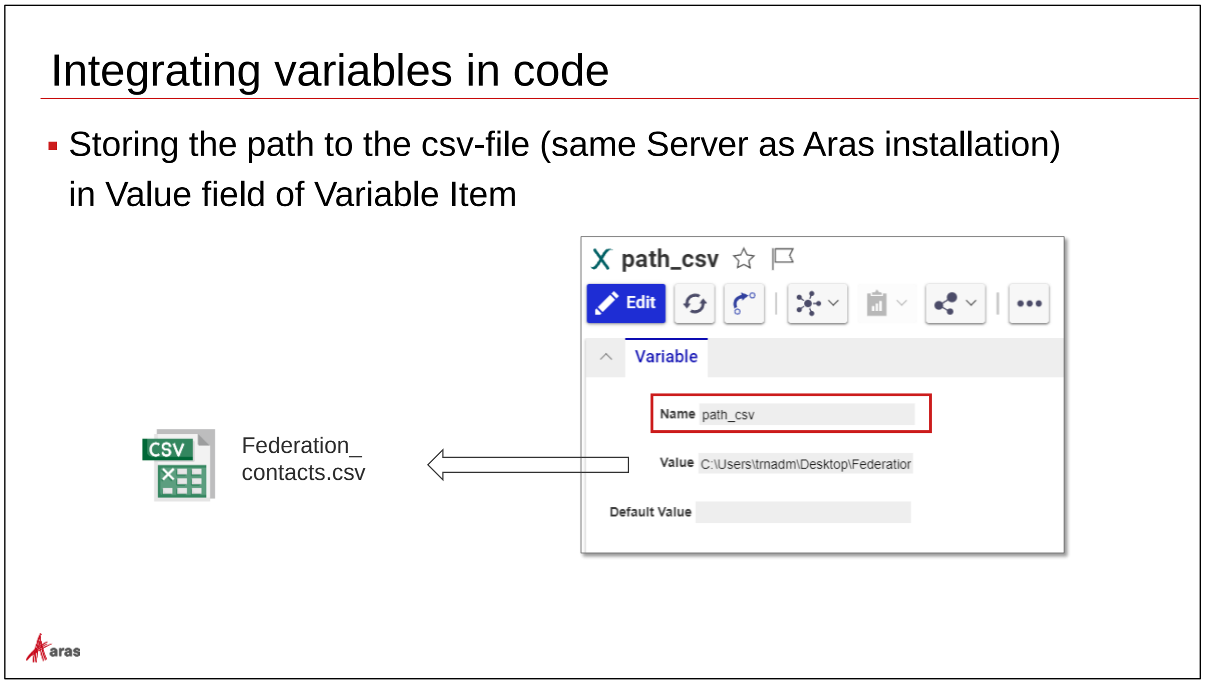


Properties | RelationshipTypes | Views | Server Events | Actions | Life Cy

Properties

Hidden

Name	Federated	Label	Data Type	Data S
contact_name	☑	Contact Name	String	
main_phone	☑	Main Phone	String	
address	☐	Address	String	
address_state	☐	State	List	States



Integrating variables in code

- Storing the path to the csv-file (same Server as Aras installation) in Value field of Variable Item

path_csv

Name path_csv

Value C:\Users\trnadm\Desktop\Federatior

Default Value

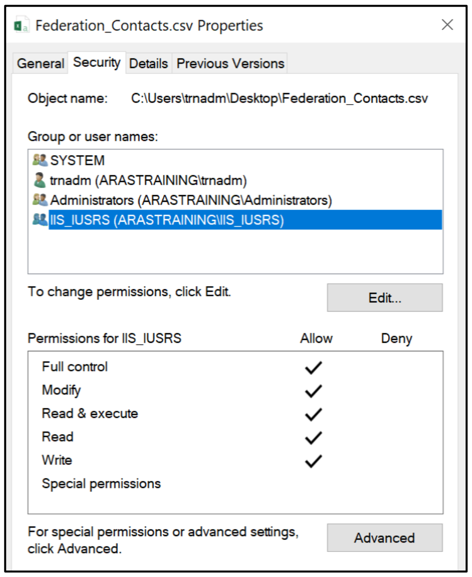
Federation_ contacts.csv

Integrating Variables in Code

Variable Items can be used to store values, which may then be accessed by custom programming code. In this example, a Variable Item is used to specify the absolute path to the location of the external repository (the file “**pc_CRM_customers_repository.csv**”). Later, all methods trying to perform CRUD operations on federated property values will obtain the path to the external repository from the value of a Variable Item named "**pc_CRM_customers_repo_location**". The advantage of using this implementation is that the path to the file can be managed by one Variable Item and does not need to be defined in multiple methods individually, in the event of a change in the repository location.

Note on user’s permissions

Proper access rights will be needed so that the external repository may be accessed by the server process. The repository location must allow IIS_IUSRS users both read and write access.



Federation_Contacts.csv Properties

General | Security | Details | Previous Versions

Object name: C:\Users\trnadm\Desktop\Federation_Contacts.csv

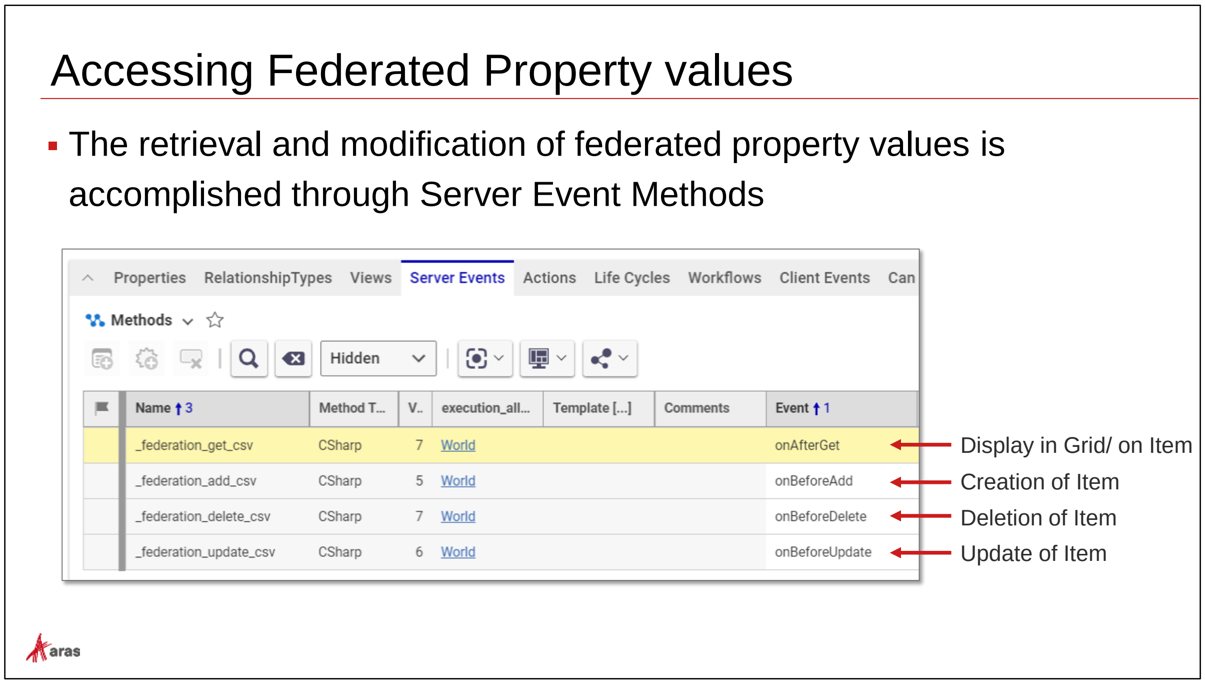
Group or user names:

SYSTEM
trnadm (ARASTRAINING\trnadm)
Administrators (ARASTRAINING\Administrators)
IIS_IUSRS (ARASTRAINING\IIS_IUSRS)

To change permissions, click Edit.

Permissions for IIS_IUSRS	Allow	Deny
Full control	✓	
Modify	✓	
Read & execute	✓	
Read	✓	
Write	✓	
Special permissions		

For special permissions or advanced settings, click Advanced.



Accessing Federated Property values

- The retrieval and modification of federated property values is accomplished through Server Event Methods

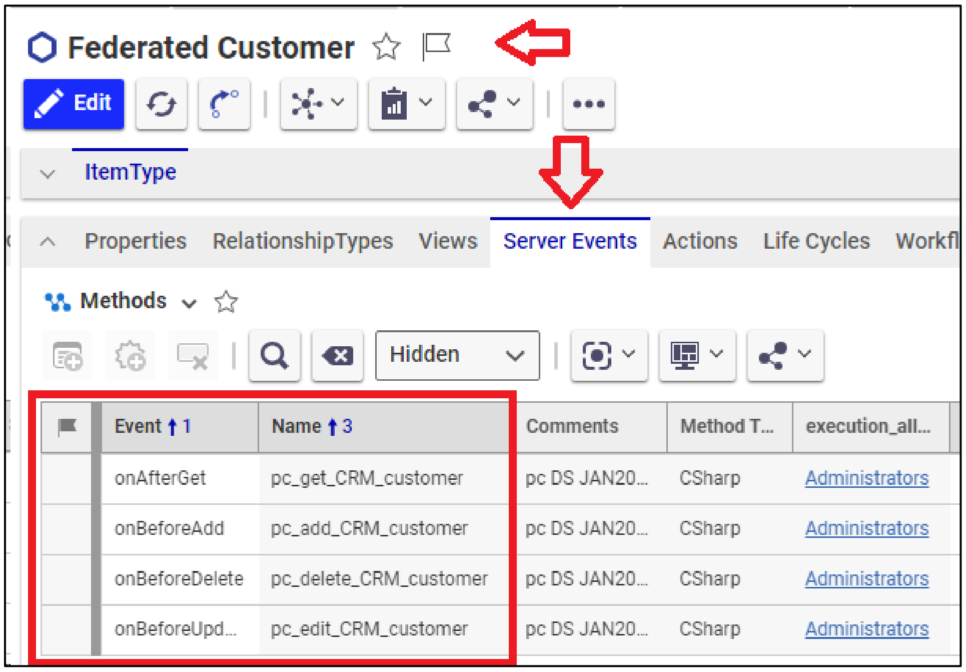
Name 3	Method T...	V..	execution_all...	Template [...]	Comments	Event 1
_federation_get_csv	CSharp	7	World			onAfterGet
_federation_add_csv	CSharp	5	World			onBeforeAdd
_federation_delete_csv	CSharp	7	World			onBeforeDelete
_federation_update_csv	CSharp	6	World			onBeforeUpdate

Accessing Federated Property Values

Server Event methods must be used when performing CRUD operations on federated property value. They must be configured at the ItemType to establish a connection to the remote system – for this training example, the remote repository is represented by the comma-separated-value file.

Try it:

1. Verify the Server Events set for the Federated Customer ItemType



Event 1	Name 3	Comments	Method T...	execution_all...
onAfterGet	pc_get_CRM_customer	pc DS JAN20...	CSharp	Administrators
onBeforeAdd	pc_add_CRM_customer	pc DS JAN20...	CSharp	Administrators
onBeforeDelete	pc_delete_CRM_customer	pc DS JAN20...	CSharp	Administrators
onBeforeUpd...	pc_edit_CRM_customer	pc DS JAN20...	CSharp	Administrators

Retrieving a Federated Property/Item

To retrieve a federated property/Item, the “onAfterGet” server event must be used.

Note that retrieving data may return a result set with more than one database row, also known as a collection, so your method code should be able to handle this scenario.

In this example, the Federated Customer Item is being populated with external data from the csv file repository. The “setProperty” method is used to assign the appropriate values to the related properties based on data retrieved from the external data source.

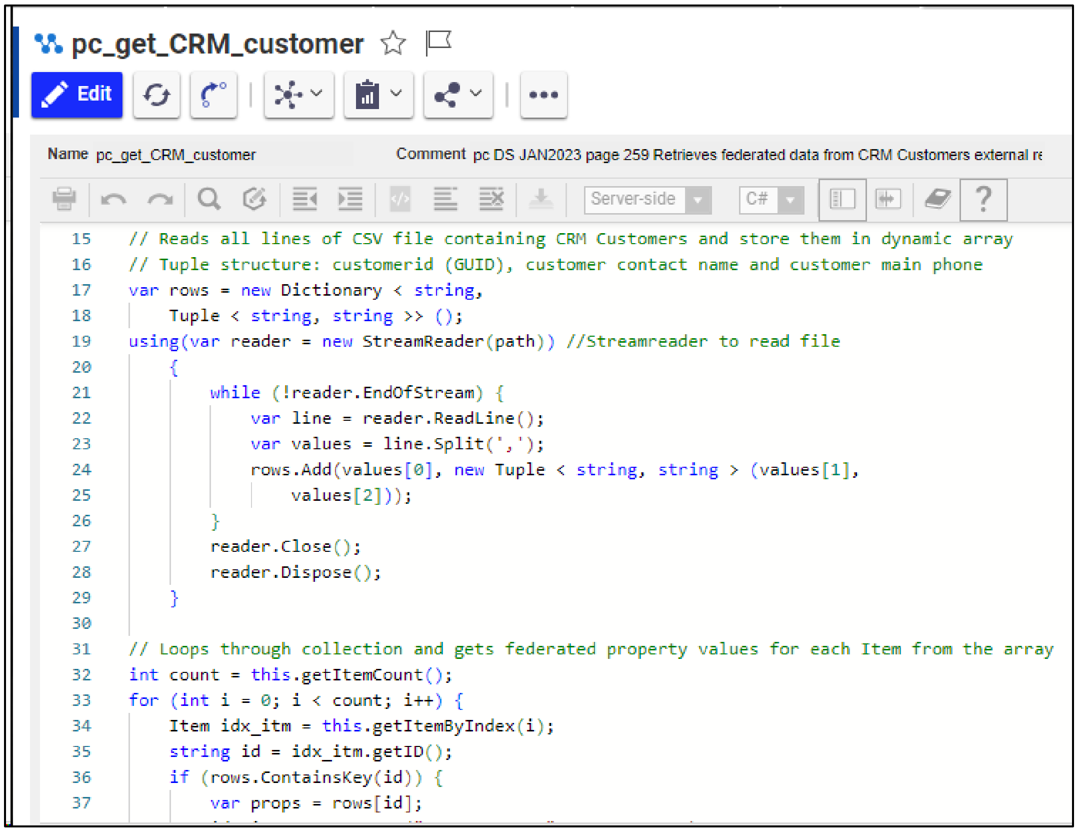
C# Server Side Method Solution for Property Retrieval

```
// PCOSTA ACE2024 - Federation Session - Federated CRM Customer Retrieval implementation
// Processes CSV repository containing federated items (CRM Customers in this example)

// Add this method as an onAfterGet Server Event in the Federated Customer ItemType

// Gets complete path and file name of CSV file from Variable named
"pc_CRM_customers_repo_location"
Item variablePath = this.newItem("Variable", "get");
variablePath.setProperty("name", "pc_CRM_customers_repo_location");
Item pathItem = variablePath.apply();
string path = pathItem.getProperty("value", "");
if (path == "") {
   return this.getInnovator().newError("Path and file name of CSV file needs to be defined on
Variable named 'pc_CRM_customers_repo_location'");
}
// Reads all lines of CSV file containing CRM Customers and store them in dynamic array
// Tuple structure: customerid (GUID), customer contact name and customer main phone
var rows = new Dictionary < string,
   Tuple < string, string >> ();
using(var reader = new StreamReader(path)) //Streamreader to read file
   {
      while (!reader.EndOfStream) {
         var line = reader.ReadLine();
         var values = line.Split(',');
         rows.Add(values[0], new Tuple < string, string > (values[1],
            values[2]));
      }
```

```
            reader.Close();
            reader.Dispose();
        }

    // Loops through collection and gets federated property values for each Item from the array
    int count = this.getItemCount();
    for (int i = 0; i < count; i++) {
        Item idx_itm = this.getItemByIndex(i);
        string id = idx_itm.getID();
        if (rows.ContainsKey(id)) {
            var props = rows[id];
            idx_itm.setProperty("contact_name", props.Item1);
            idx_itm.setProperty("main_phone", props.Item2);
        }
    }
    return this;
```

Try it:

1. Open the method associated with "OnAfterGet" Server event



Name pc_get_CRM_customer Comment pc DS JAN2023 page 259 Retrieves federated data from CRM Customers external

```
// Reads all lines of CSV file containing CRM Customers and store them in dynamic array
// Tuple structure: customerid (GUID), customer contact name and customer main phone
var rows = new Dictionary < string,
    Tuple < string, string >> ();
using(var reader = new StreamReader(path)) //Streamreader to read file
    {
        while (!reader.EndOfStream) {
            var line = reader.ReadLine();
            var values = line.Split(',');
            rows.Add(values[0], new Tuple < string, string > (values[1],
                values[2]));
        }
        reader.Close();
        reader.Dispose();
    }

// Loops through collection and gets federated property values for each Item from the array
int count = this.getItemCount();
for (int i = 0; i < count; i++) {
    Item idx_itm = this.getItemByIndex(i);
    string id = idx_itm.getID();
    if (rows.ContainsKey(id)) {
        var props = rows[id];
```

Adding New Item to a Federated Dataset

To add a new data row to the external csv-file repository, you can use either the “onBeforeAdd” or “onAfterAdd” server events.

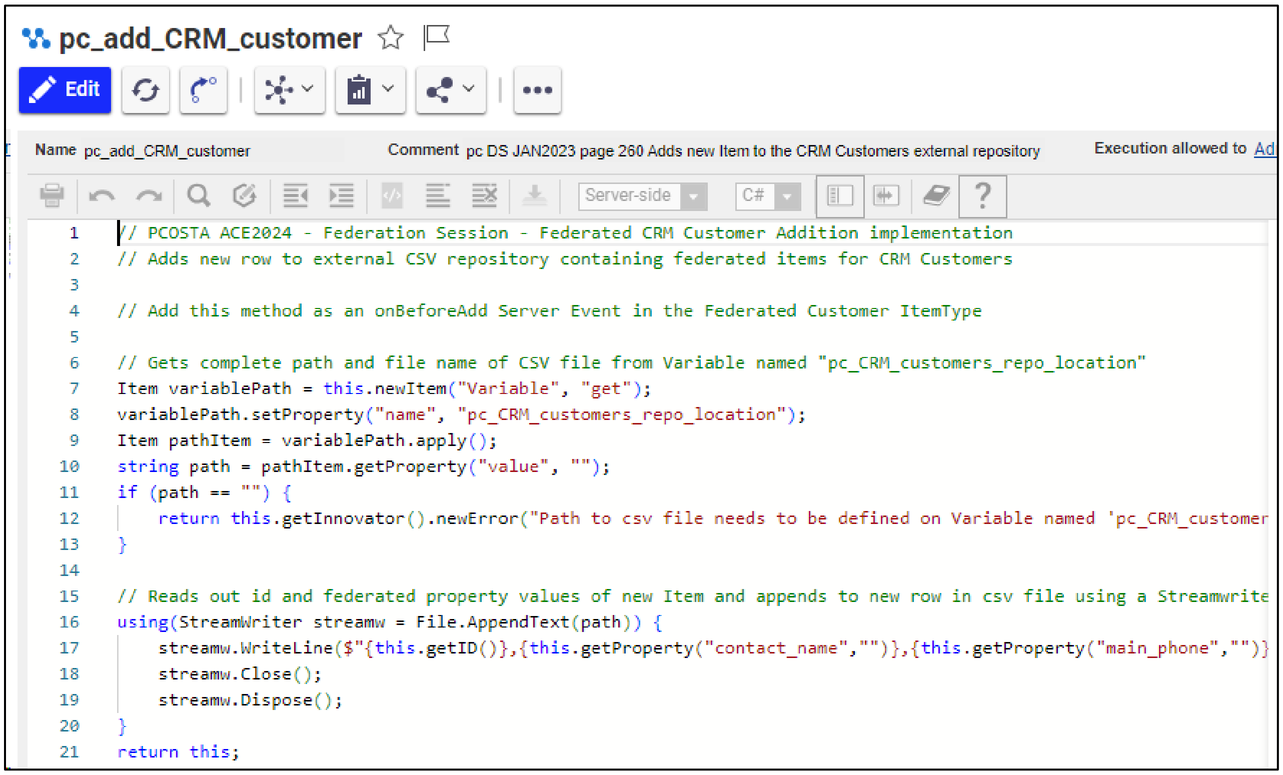
In this example, a new row will be created in the csv-file (pc_CRM_customer_repository.csv).

C# Server Side Method Solution for Item Addition to the Dataset

```
// PCOSTA ACE2024 - Federation Session - Federated CRM Customer Addition implementation
// Adds new row to external CSV repository containing federated items for CRM Customers


// Add this method as an onBeforeAdd Server Event in the Federated Customer ItemType


// Gets complete path and file name of CSV file from Variable named
"pc_CRM_customers_repo_location"
Item variablePath = this.newItem("Variable", "get");
variablePath.setProperty("name", "pc_CRM_customers_repo_location");
Item pathItem = variablePath.apply();
string path = pathItem.getProperty("value", "");
if (path == "") {
   return this.getInnovator().newError("Path to csv file needs to be defined on Variable named
'pc_CRM_customers_repo_location'");
}


// Reads out id and federated property values of new Item and appends to new row in csv file
using a Streamwriter
using(StreamWriter streamw = File.AppendText(path)) {

streamw.WriteLine($"{this.getID()},{this.getProperty("contact_name","")},{this.getProperty("m
ain_phone","")}");
   streamw.Close();
   streamw.Dispose();
}
return this;
```

Try it:

1. Open the method associated with “OnBeforeAdd” Server event



pc_add_CRM_customer

Name pc_add_CRM_customer   Comment pc DS JAN2023 page 260 Adds new Item to the CRM Customers external repository   Execution allowed to Ad

Server-side   C#

```
// PCOSTA ACE2024 - Federation Session - Federated CRM Customer Addition implementation
// Adds new row to external CSV repository containing federated items for CRM Customers

// Add this method as an onBeforeAdd Server Event in the Federated Customer ItemType

// Gets complete path and file name of CSV file from Variable named "pc_CRM_customers_repo_location"
Item variablePath = this.newItem("Variable", "get");
variablePath.setProperty("name", "pc_CRM_customers_repo_location");
Item pathItem = variablePath.apply();
string path = pathItem.getProperty("value", "");
if (path == "") {
    return this.getInnovator().newError("Path to csv file needs to be defined on Variable named 'pc_CRM_customer
}

// Reads out id and federated property values of new Item and appends to new row in csv file using a Streamwrite
using(StreamWriter streamw = File.AppendText(path)) {
    streamw.WriteLine($"{this.getID()},{this.getProperty("contact_name","")},{this.getProperty("main_phone","")}
    streamw.Close();
    streamw.Dispose();
}
return this;
```

Updating Changes to the Remote System

To update an existing row in the external csv-file repository, either the “**onBeforeUpdate”** or “**onAfterUpdate”** server events can be used to trigger the update method below, when an existing Item is saved.

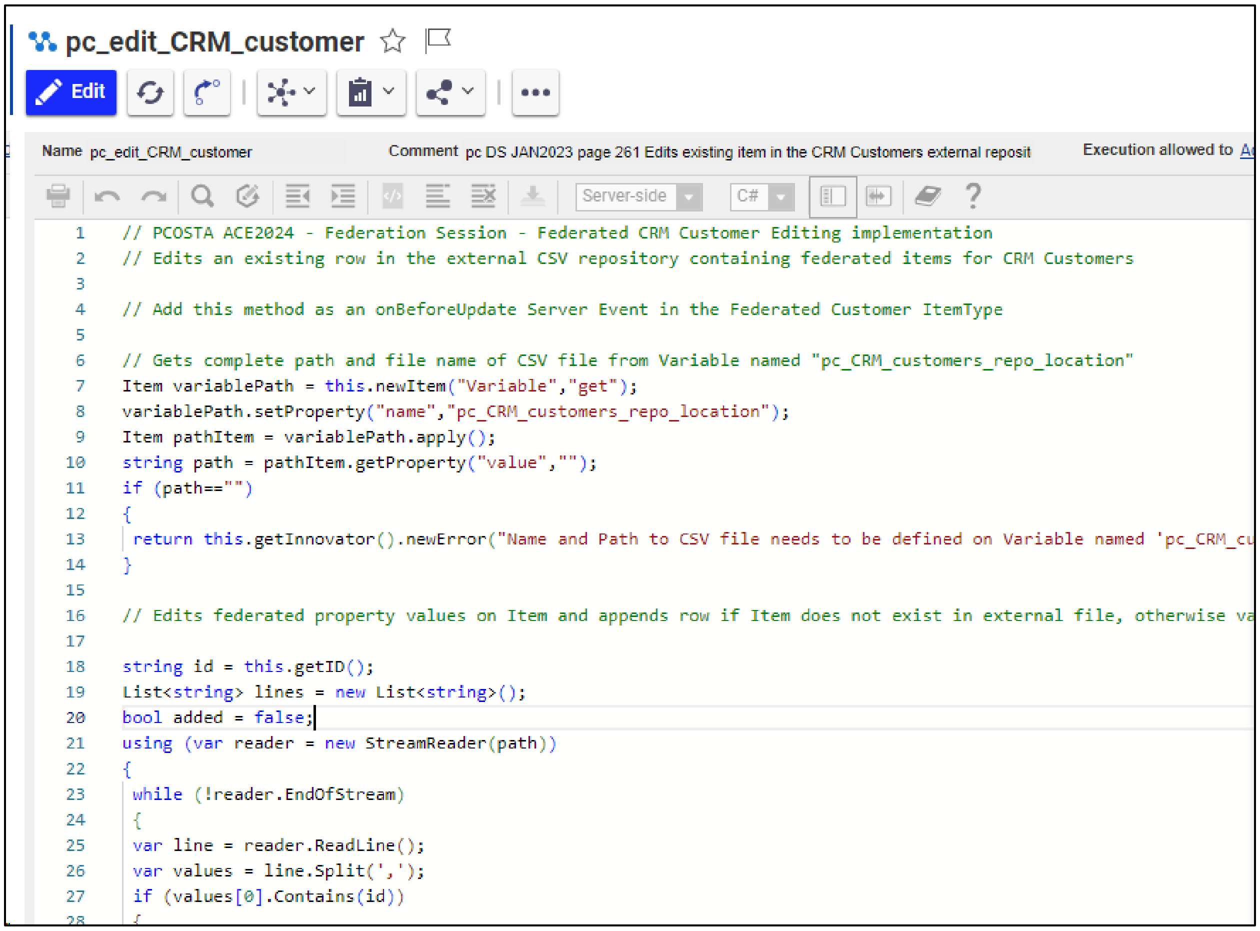
C# Server Side Method Solution for Federated Property Update

```
// PCOSTA ACE2024 - Federation Session - Federated CRM Customer Editing implementation
// Edits an existing row in the external CSV repository containing federated items for CRM Customers

// Add this method as an onBeforeUpdate Server Event in the Federated Customer ItemType

// Gets complete path and file name of CSV file from Variable named "pc_CRM_customers_repo_location"
Item variablePath = this.newItem("Variable","get");
variablePath.setProperty("name","pc_CRM_customers_repo_location");
Item pathItem = variablePath.apply();
```

```
string path = pathItem.getProperty("value","");
if (path=="")
{
 return this.getInnovator().newError("Name and Path to CSV file needs to be defined on
Variable named 'pc_CRM_customers_repo_location'");
}

// Edits federated property values on Item and appends row if Item does not exist in external
file, otherwise values are replaced

string id = this.getID();
List<string> lines = new List<string>();
bool added = false;
using (var reader = new StreamReader(path))
{
 while (!reader.EndOfStream)
 {
 var line = reader.ReadLine();
 var values = line.Split(',');
 if (values[0].Contains(id))
 {
 values[1]=this.getProperty("contact_name","");
 values[2]=this.getProperty("main_phone","");
 line = string.Join(",",values);
 added = true;
 }
 lines.Add(line);
 }
}
if (!added)            // Adds a row if Item does not exist in the external CSV file
{
lines.Add($"{id},{this.getProperty("contact_name","")},{this.getProperty("main_phone","")}");
}
File.WriteAllLines(path, lines);
return this;
```

Try it:

1. Open the method associated with “OnBeforeUpdate” Server event



pc_edit_CRM_customer

Name pc_edit_CRM_customer Comment pc DS JAN2023 page 261 Edits existing item in the CRM Customers external reposit Execution allowed to A

```
// PCOSTA ACE2024 - Federation Session - Federated CRM Customer Editing implementation
// Edits an existing row in the external CSV repository containing federated items for CRM Customers

// Add this method as an onBeforeUpdate Server Event in the Federated Customer ItemType

// Gets complete path and file name of CSV file from Variable named "pc_CRM_customers_repo_location"
Item variablePath = this.newItem("Variable","get");
variablePath.setProperty("name","pc_CRM_customers_repo_location");
Item pathItem = variablePath.apply();
string path = pathItem.getProperty("value","");
if (path=="")
{
 return this.getInnovator().newError("Name and Path to CSV file needs to be defined on Variable named 'pc_CRM_cu
}

// Edits federated property values on Item and appends row if Item does not exist in external file, otherwise va

string id = this.getID();
List<string> lines = new List<string>();
bool added = false;
using (var reader = new StreamReader(path))
{
 while (!reader.EndOfStream)
 {
 var line = reader.ReadLine();
 var values = line.Split(',');
 if (values[0].Contains(id))
```

Removing Data from a Federated Dataset

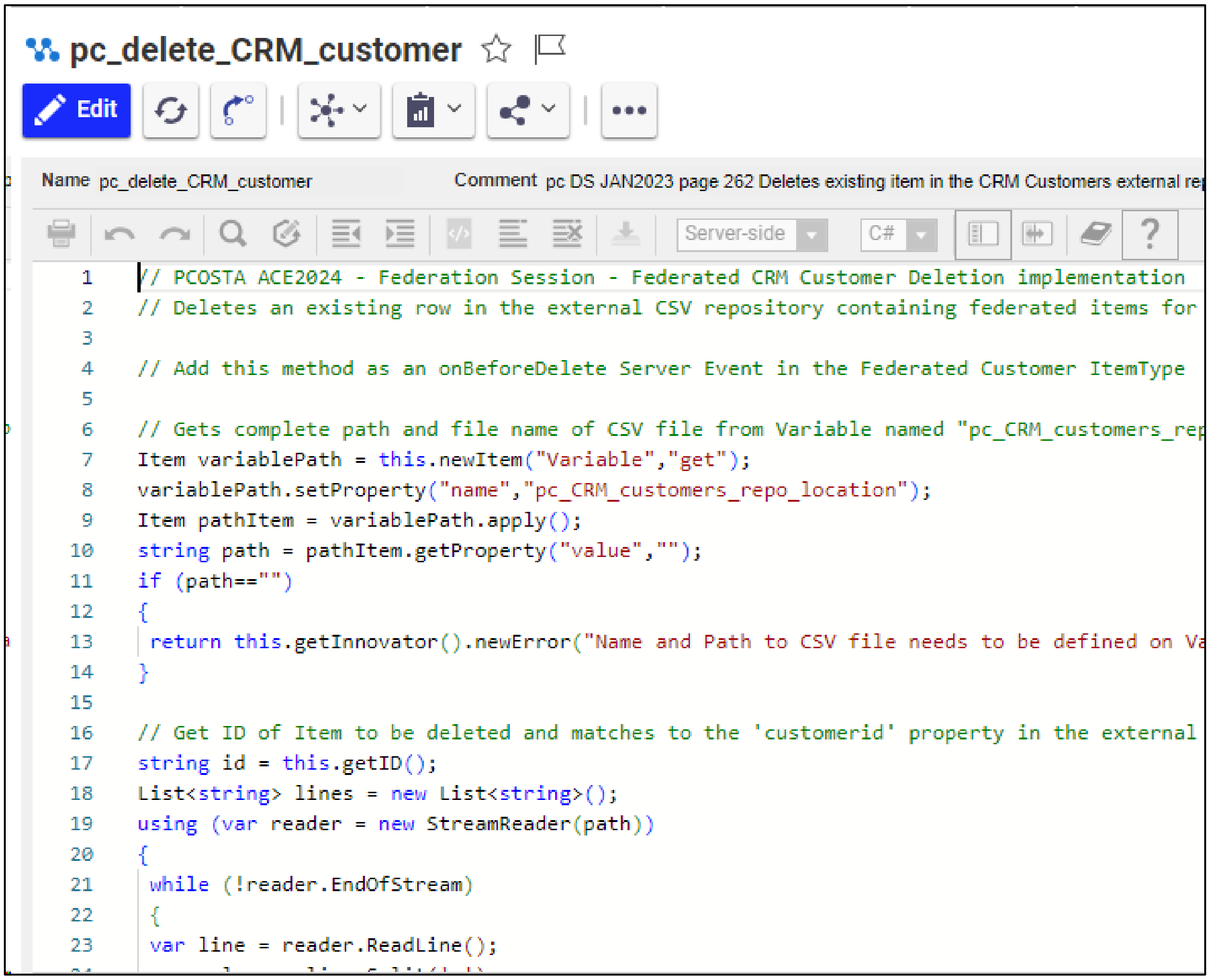
Either the “onBeforeDelete” or “onAfterDelete” server events can be used to allow deletion of data from an external repository.

C# Server Side Method Solution for Federated Item Deletion

```
// PCOSTA ACE2024 - Federation Session - Federated CRM Customer Deletion implementation
// Deletes an existing row in the external CSV repository containing federated items for CRM
Customers
// Add this method as an onBeforeDelete Server Event in the Federated Customer ItemType
// Gets complete path and file name of CSV file from Variable named
"pc_CRM_customers_repo_location"
Item variablePath = this.newItem("Variable","get");
variablePath.setProperty("name","pc_CRM_customers_repo_location");
Item pathItem = variablePath.apply();
string path = pathItem.getProperty("value","");
if (path=="")
{
 return this.getInnovator().newError("Name and Path to CSV file needs to be defined on Variable
named 'pc_CRM_customers_repo_location'");
}
// Get ID of Item to be deleted and matches to the 'customerid' property in the external CSV file
string id = this.getID();
List<string> lines = new List<string>();
using (var reader = new StreamReader(path))
{
 while (!reader.EndOfStream)
 {
 var line = reader.ReadLine();
 var values = line.Split(',');
 if (!(values[0].Contains(id)))
 {
 lines.Add(line);
 }
 }
}
File.WriteAllLines(path, lines);
return this;
```

Try it:

1. Open the method associated with “OnBeforeDelete” Server event



Name pc_delete_CRM_customer Comment pc DS JAN2023 page 262 Deletes existing item in the CRM Customers external rep

```
// PCOSTA ACE2024 - Federation Session - Federated CRM Customer Deletion implementation
// Deletes an existing row in the external CSV repository containing federated items for

// Add this method as an onBeforeDelete Server Event in the Federated Customer ItemType

// Gets complete path and file name of CSV file from Variable named "pc_CRM_customers_rep
Item variablePath = this.newItem("Variable","get");
variablePath.setProperty("name","pc_CRM_customers_repo_location");
Item pathItem = variablePath.apply();
string path = pathItem.getProperty("value","");
if (path=="")
{
 return this.getInnovator().newError("Name and Path to CSV file needs to be defined on Va
}

// Get ID of Item to be deleted and matches to the 'customerid' property in the external
string id = this.getID();
List<string> lines = new List<string>();
using (var reader = new StreamReader(path))
{
 while (!reader.EndOfStream)
 {
 var line = reader.ReadLine();
```

Summary

The following topics and activities were covered in this session:

- Review of Federated Items and Properties.
- Discussion of Federated Data Sources.
- Configuration of Federated Property or ItemType.
- Execution of CRUD operations with Federated Property values.