



NICE Designer Self-Paced Automation Essentials

NICE Advanced Process Automation



Course Workbook
For Automation Developers





BUSINESS ENTITIES LAYER: FUNCTIONS AND EVENTS



Lesson Objectives

By the end of this lesson you will be able to:

- Create Functions and Events
- Work with data structures

NICE®

- Up until now you have learned how to capture Screen Elements, and how to create Business Entities, i.e. Types and Instances.
- This lesson will cover how to create and handle different data structures. In other words, how to populate Business Entities with data, and manipulate them for implementation purposes.



Notes from Demo

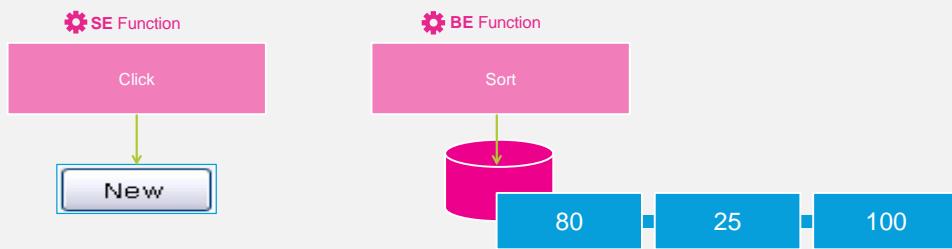
Functions Basics



NICE®

What are Functions?

- Functions perform an action on an element
- Elements in the Designer will have different Functions available, depending on their Type:
 - SE Functions operate on SE's
 - BE Functions operate on BE's



- Always select the element on which you want to operate in order to access the set of Functions available for it.

NICE®

Function's Structure



Return Type

Input Parameters

Body

Return Type and **Input Parameters** are optional.

- Function can return an element as output
- Function can get elements as input

Body is mandatory.

It contains a set of Instructions, that are executed top-down.

?



Number

Customer's birth year

Return Current year –
Customer's birth year

Example

- What is this Function's output?
- What is this Function's input?
- What does this Function do?

NICE®

- Functions are defined at the Type tab, but operate on the Type's Instances.
- They are triggered at an upper layer.
- The example function returns the customer's age.

How to Create a Function



Remember that the Function is defined at the **Type** level, but operates on the Type's **Instances**

The screenshot shows the 'Business Entities' tab selected. A tooltip indicates Step 1: 'In the Business Entities tab, click the Type you wish to add a Function to'. Step 2: 'Click New Function'. Step 3: 'Select Return Type if Function returns a value'. Step 4: 'Add parameters if Function receives any'. Step 5: 'Insert Instructions to execute at Function Body'. The 'Function Body' section contains the code: 'Return Subtract Customer's birth year from (Year_of (Current_date))'.

Create a Type

Create Properties

Create Functions

Create Events

Create an Instance

NICE®

- Some Functions do not return any Types (To set this, in Step 3, select “None” as the Return type). For example, a Function that assigns a result of a calculation into a Type.

Built-in Functions

- Library Object Functions are available in the Designer.
Here are a few examples:

Math	Date	Text	Window	Clipboard
Subtract	Year of	Concatenate	Get Active Process	Copy text
Add	Get Current Date	Get length	Set Focus to Window	Copy file





Notes from Demo

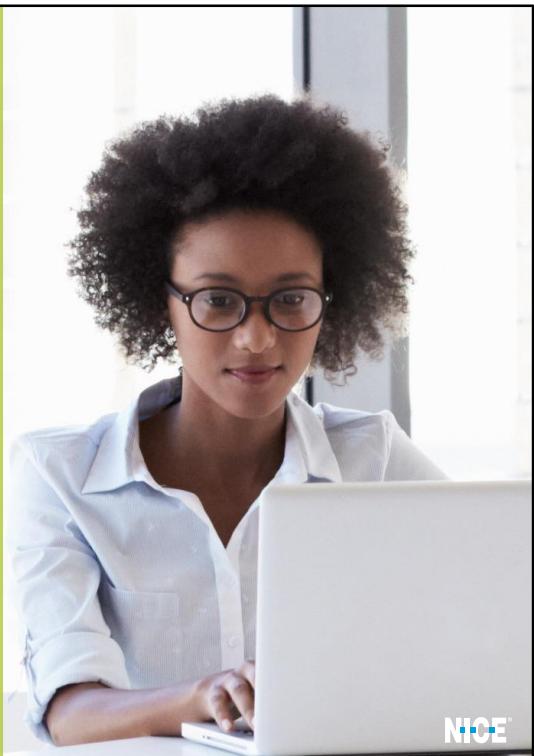
DO IT YOURSELF

Creating a Function

- Create a BE Type called 'Address' with the following Properties:
 - Street
 - City
 - State
 - Zip
- Make an Instance of this Type
- Create a Function called 'Full Name', which:
 - Returns a Text value
 - Has 2 Input Parameters: "First Name" and "Last Name"
 - Contains an Instruction to Return the free text: "Full Name" (This will be replaced with a more meaningful value later)

NICE®

Function Body (Instructions)



Function Invocation

- Function Invocation Instruction calls another Function.
This Function can be:



Function Location

Function Invocation → Library Objects →
Text →
Concatenate

Function Invocation → Screen Elements →
Button (specific SE) →
Click

Function Invocation → Business Entities →
List (specific BE) →
Sort

NICE®

- **Business Entities** – Can be either a Function you created or a built-in Function like sorting a list.
- **Library objects** – A built-in Functions library, much like Excel. Drill-down menu will display the group Functions (Math Functions / Date and time Functions etc.), and Functions you can invoke in this group (Sum, Current date etc).

How to Invoke Library Objects Functions

- The “Concatenate” Function is one of many Library Functions.

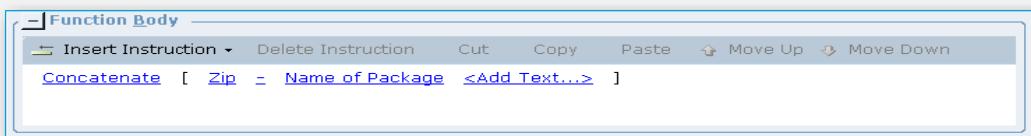
The screenshot illustrates the process of finding the "Concatenate" function within the NICE Designer Automation interface. It shows three nested windows:

- Top Window:** A context menu titled "Function Body" is open, with "Function Invocation" selected. Other options include Assignment, If, For, For Each, Return, Local Variable, and Comment.
- Middle Window:** A tree view of "Library Objects" is shown under "Regular Expression". The "Text" node is selected.
- Bottom Window:** A list of functions under "Text" is displayed, including "Concatenate", "Convert to lower case", "Convert to upper case", "Extract prefix subtext w", and "Extract suffix subtext w".

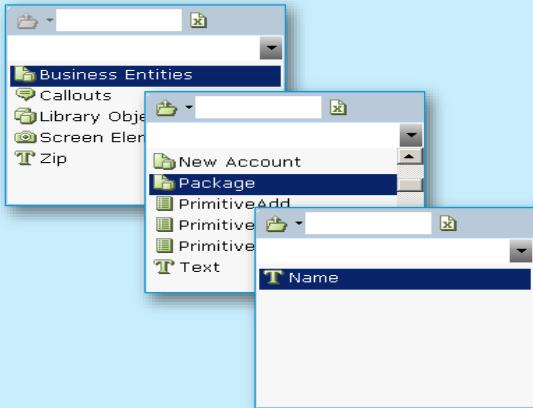
Drill down to Concatenate

NICE®

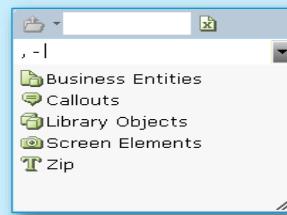
Library Objects Syntax- Concatenate Example



You can concatenate values stored in Business Entities...



... as well as free text, including spaces, commas, hyphens and all other punctuation



NICE®



Notes from Demo

DO IT YOURSELF

Invoking a Library Function: ‘Concatenate’

- Add the relevant instructions to the ‘Full Name’ function to Return the Concatenation of the “First Name” and “Last Name” parameters
- Under the ‘Address’ Type, create a function called ‘Full Address’ that returns type ‘Text’
- Add the relevant instructions to concatenate the values in the ‘Address’ type as follows:
Street, City, State Zip

NICE®

SE Functions

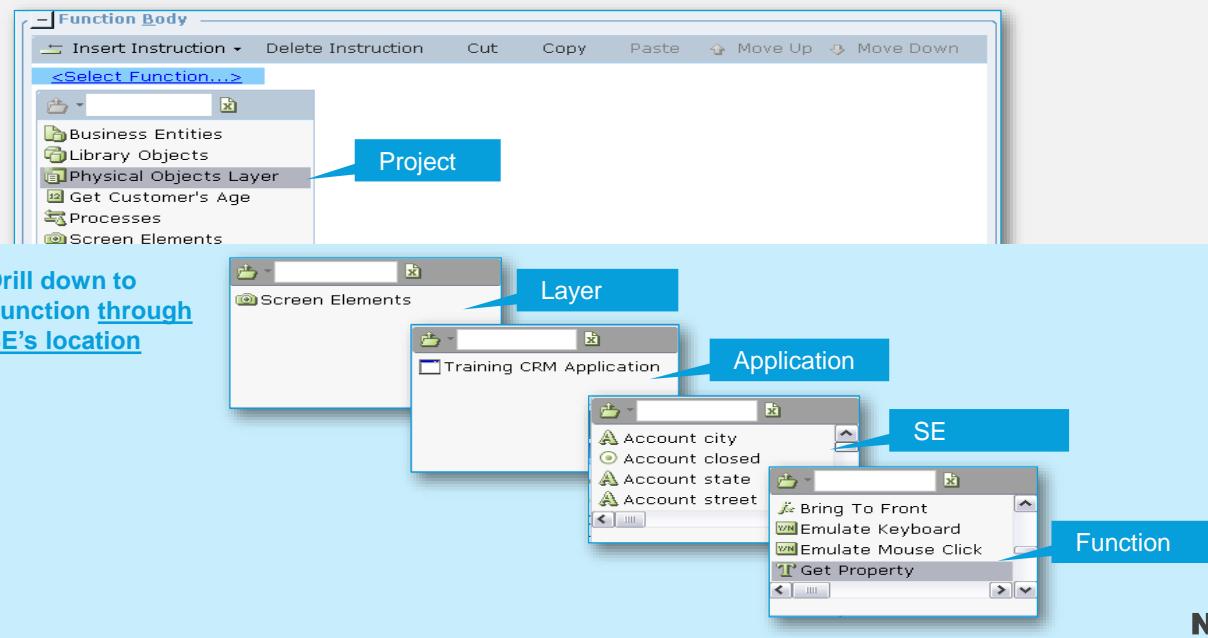
- Each Screen Element has different built-in Functions to invoke, depending on its type

Window	Text Field	Button	Combo
Bring to Front	Emulate Keyboard	Click	Make set of Items on list
Name of Window in focus	Set Focus	Get Property	Get value at X from the list

NICE®

- Before you take action on a window, it's important to first bring it to the front, so it is the active window.
- “Name of window in focus” is a Library Function.

How to Invoke SE Functions





Notes from Demo

DO IT YOURSELF

Invoking a Screen Element Function

- Create a Function called 'Click Add', which invokes the 'Click' Function associated with the 'Add' button you captured earlier.

NICE®

BE Functions – “Add Item to List” Example

- There is an Add Item Function associated with every List
- Use it to add the current contact name to My List

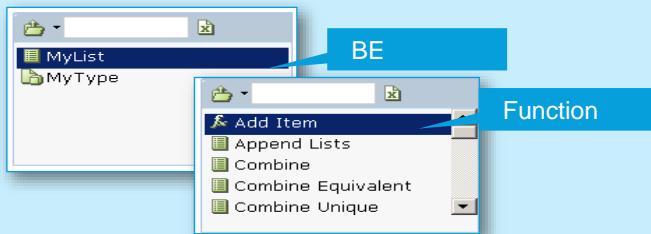


NICE®

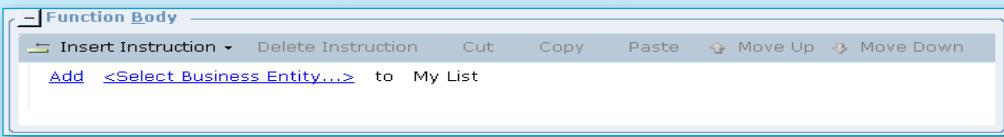
How to Invoke BE Functions



Drill down to
Function through
the BE that
contains it



Function



NICE®



Notes from Demo

DO IT YOURSELF

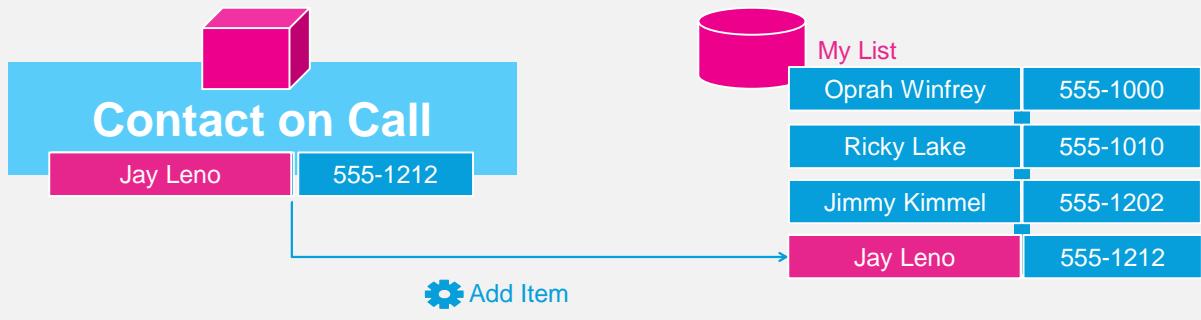
Adding to a List of Primitive Business Entities

- In the Instances tab, create a List of Text and call it 'List of Zip Codes'
- In the Types tab, create a function under the 'Address' type and call it 'Add Zip Code to List'
- In the function, add the appropriate instruction to add the 'Zip' Business Entity to the List you just created

NICE®

Adding Composite Business Entities to a List

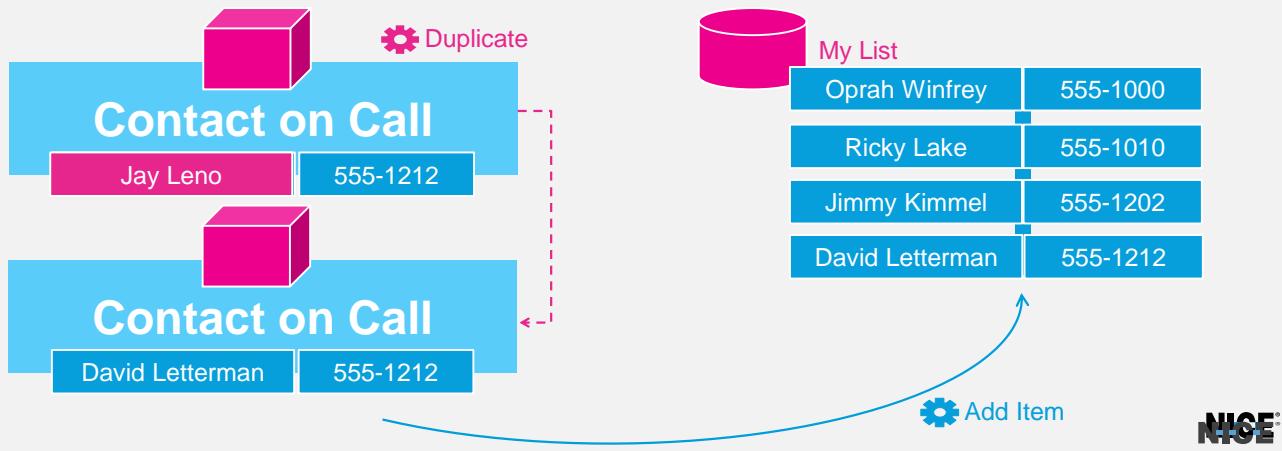
- When you add an item to a list of composite BE's, the Instance is connected to that lists' position.
- So... when its value changes, so does its corresponding value on the list.



NICE®

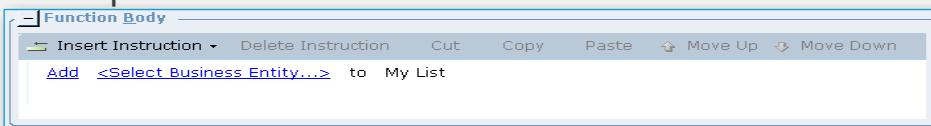
Adding Composite Business Entities to a List

- To prevent values from being overwritten, add a duplicate of the Instance (not the Instance itself) to the list.

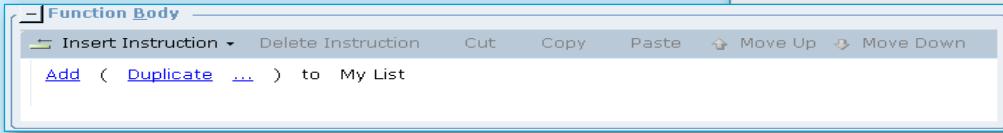
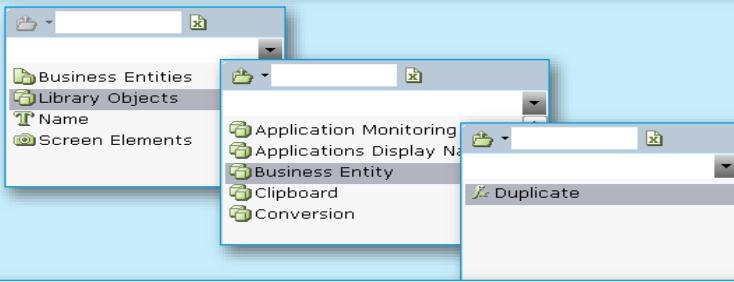


Adding Business Entities to a List

- To Add a Duplicate BE to a List:



[**Drill down to *Duplicate* Function through Library Objects → BE**](#)



NICE®



Notes from Demo

DO IT YOURSELF

Adding to a List of Composite Business Entities

- Create a List of Address Types
- In the Types tab, create a Function under the 'Address' type and call it 'Add Address to List'
- In the Function, add the appropriate Instruction to add the entire 'Address' Business Entity to the List you just created

NICE®

Simple Instructions

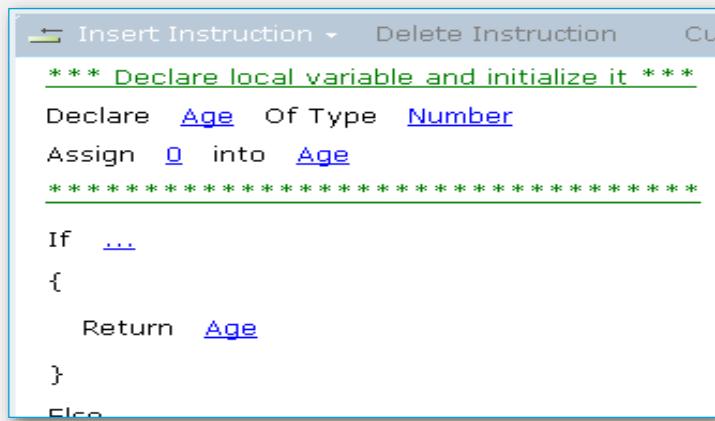
Insert Instruction →

Comment

Local Variable

Assignment

Return



NICE®

- **Comment**
 - Add documentation for implementation
- **Local Variable**
 - Declare a variable of a certain Type, to be saved and used locally within the Function, but only while the Function is running
- **Assignment**
 - Assign a value to a PO, BE or a local variable
- **Return**
 - If the Function's return type is NONE, no value is returned and the Function stops running. If the Function's return type is Number for example, define which variable or value will be returned.
- Assignment Instruction allows you to change BEs' values after their initial assignment (defined in the Data flow tab).



Notes from Demo

DO IT YOURSELF

Instructions

- Create a Type called 'Customer' and make an Instance of it
- Under this Type, create a Function called 'IsSenior', which returns a Boolean value
 - Insert Comments to your Function to describe the actions you intend to implement in the Function
 - Declare a Local variable of type 'Number' called 'Age'
 - Assign the value '60' to 'Age'
 - Insert the Instruction to return out of the Function with the value 'True'

NICE®

'IF' Instruction

```
If (Condition)
    Instruction A
Else
    Instruction B
```



What will be the value of Is_senior if the client's age is 60?

```
+ Add Condition - Delete () Log
If ... Client's age is larger than 60
{
    Assign True into Is_senior
}
Else
{
    Assign False into Is_senior
}
```

NICE®

- If the client's age is greater than 60, then Is_senior=true. Otherwise, Is_senior=false



Notes from Demo

DO IT YOURSELF

'IF' Instruction

- Modify your 'IsSenior' function as follows:
 - Add an input parameter of type 'Number' called 'Age_Input'
 - Assign this parameter to the 'Age' local variable instead of the static value assigned earlier
 - Add instructions and comments that Return the value TRUE if Age is 60 or above, and Return FALSE otherwise

NICE®

'For Each' Instruction

For Each (Item in a set)
Do this instruction

The Instruction in the loop
will be performed for each
item in the set



Example

- What is the instruction in the “For Each” Loop here?
- How many times will the loop run?
- What does this function do?

```
Declare X Of Type Number
For Each Text AKA Item In List1
{
    Assign Sum [ 1 X <Add Number...> ] into X
}
Return X
```

NICE®

- The instruction to do in this “for each” loop is to increment X by 1.
- The “for each” loop will run as many times as the number of items in List1.
- Practically speaking, the function will return the number elements in List1: Let’s say List1 has two items. When X is created, its default value is 0. Then, for each item on List1 (twice) we increment X by 1 so at the end of the loop X=2. Then X is returned.



Notes from Demo

DO IT YOURSELF

'For Each' Instruction

- Create two Lists of Text, called 'List of Phone Numbers' and 'New List of Phone Numbers' respectively
- Under the 'Customer' Type, create a Function and call it 'Add Phone Prefixes'
- Add the relevant Instructions to add '09' to the beginning of each item in the 'List of Phone Numbers' and add this new value to the 'New List of Phone Numbers'

NICE®

'FOR' Instruction

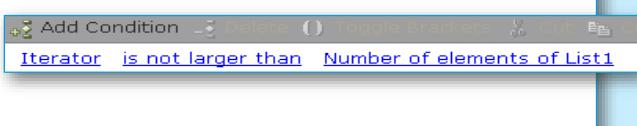
- FOR (Iterator starting value, condition to continue iterating, iterator step)
 - Do this instruction

The **FOR** loop instruction performs the instruction a set number of times. It uses a variable named Iterator to keep track of repetitions



If List1 holds 2 elements, how many clients will be added to List2?

```
For Iterator Starting With Value 0 As Long As ... Incrementing Iterator By 1
{
    Add Client to List2
}
```



NICE®

- Example explanation: Let's say **List1** has 2 items.
- **First iteration:**
 - Iterator starts with value 0.
 - The condition is TRUE (because $0 < 2$).
 - A client is added to List2.
 - Iterator is incremented to 1.
- **Second iteration:**
 - The condition is TRUE (since $1 < 2$).
 - A client is added to List2.
 - Iterator is incremented to 2.
- **Third iteration:**
 - The condition is TRUE (because 2 is indeed not larger than 2).
 - A client is added to List2.
 - Iterator is incremented to 3.
- **Fourth iteration:**
 - The condition is FALSE (because 3 is larger than 2).
- **The loop ends.**
- List2 contains 3 clients.



Notes from Demo

DO IT YOURSELF

'FOR' Instruction

- Create a List of Text, and call it 'List of Phone Numbers to Call'
- Under the 'Customer' Type, create a Function and call it 'Choose Numbers'
- Add the relevant Instructions to add every third phone number in the 'Phone Numbers' list to the 'Numbers to Call' list

NICE®

Instructions Summary

Instruction	Description	When to use?
Function Invocation	Call another Function	<ul style="list-style-type: none">▪ To act on a Screen Element▪ To invoke a Library Function▪ To call another BE Function
Assignment	Assign a value to a BE or a local variable	<ul style="list-style-type: none">▪ Assign value to local variables / Screen Elements / Business Entities
If	Apply an action according to a condition's evaluation	<ul style="list-style-type: none">▪ When the action to be done is dependant on a condition
For	Loop that iterates a set number of times, each time keeping track with an iterator	<ul style="list-style-type: none">▪ To find an item on a list

 NICE®

- The “When to use” column describes the most common examples for using this instruction.

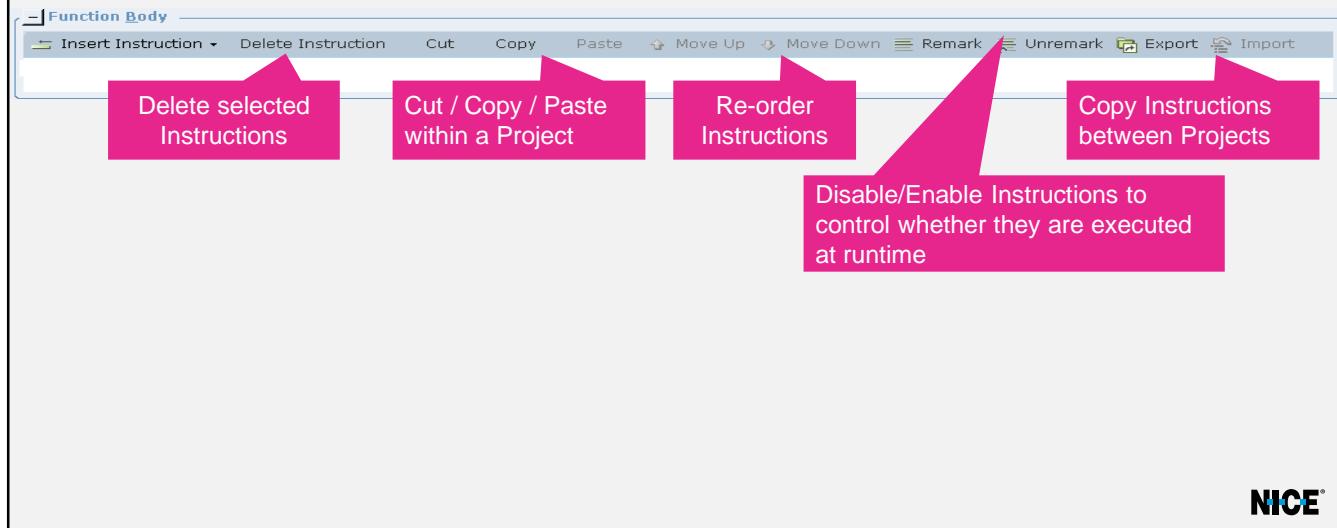
Instructions Summary

Instruction	Description	When to use?
For Each	Loop that iterates on all the items of a set	▪ To apply the same action for all items in a set
Return	End the Function from running and return a certain value	▪ Perform a calculation / query and return the result
Local Variable	Variable that exists only within the Function	▪ As needed
Comment	Add a description to instructions	▪ As needed

The NICE logo consists of the word "NICE" in a bold, sans-serif font, with a small blue square icon above the letter "I".

- The “When to use” column describes the most common examples for using this Instruction.

Other Action Editor Commands



NICE®

Events



NICE®

What are Events

- Events are raised when a change occurs. The change can be to a SE such as a button being clicked or to a BE such as a property's value being changed.



Add Button Clicked

The Event is broadcasted and any interested object can react to it



BE Event
Event Parameters

BE Events may have Parameters, whose values are populated in real time when the event occurs

Create a Type

Create Properties

Create Functions

Create Events

Create an Instance

NICE®

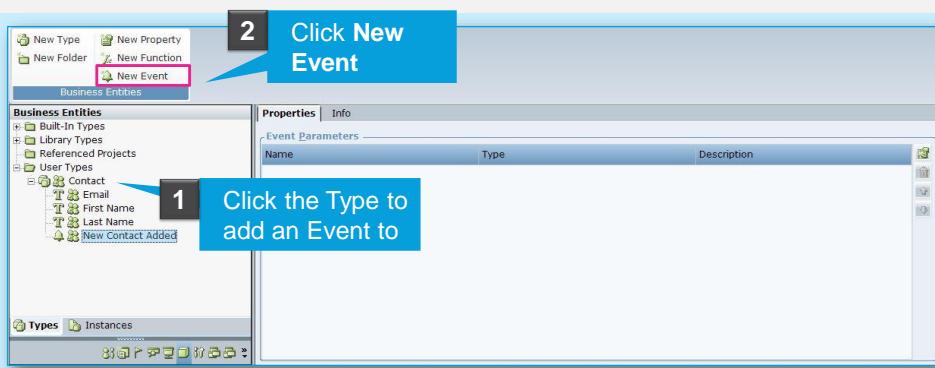
- Events are created in the Type tab, but defined in the Instance tab. This means that different Instances can have different Event parameters.
- Parameters that were assigned a value, will be used further along the Solution in upper layers.

How to Create Events

- **At the Type level:**

Create the Event

- Add Event Parameters (if any)



How to Create Events

- **At the Instance level:**

- Define when the event is raised
This is in fact, what connects the screen element's values to the BE



1 The BE Event is raised when an SE Event is raised

NICE®

Built-in Events Examples



Built-in Library Objects Events

- RT Client Shutting down
- Solution Downloaded
- Launch Bar clicked
- Multi Instance Created



Built-in SE Events

- Created / Destroyed
- Gained / Lost Focus
- Minimized / Maximized
- Clicked



Built-in BE Events

- Value Modified

Event Location

Library Objects →
Administration →
Solution Downloaded

Screen Elements →
Button (specific SE) →
Clicked

Business Entities →
BE →
Value Modified

NICE

- **Business Entities** – drill-down to the BE Instance, then click the Event you wish to raise.
- **Library objects** – A built-in Events library, to identify system Events, such as recognizing when RT Client is shutting down and when the Solution is downloaded.
- **Screen Elements** –drill-down to the SEs captured, then click the Event you wish to raise.



Notes from Demo

DO IT YOURSELF

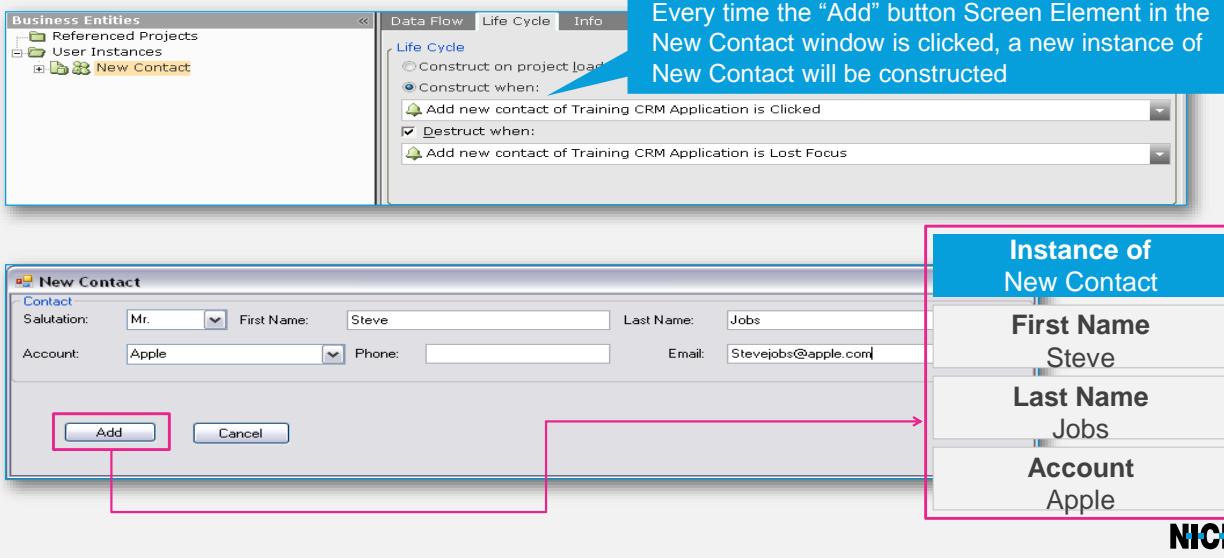
Events

- Capture the 'Add' button of the New Account Window in the Training CRM Application
- Add an Event to the 'New Account' Type called 'Account Added'
- Define the Event to be raised when the captured 'Add' button is clicked

NICE®

How to Define Life Cycle According to an Event

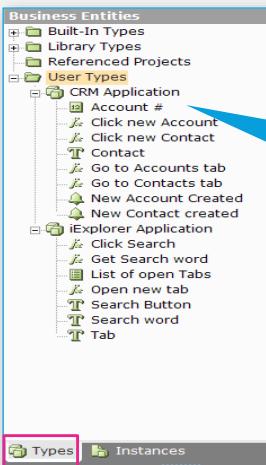
- Events can be used to define when a new BE will be constructed and destructed.



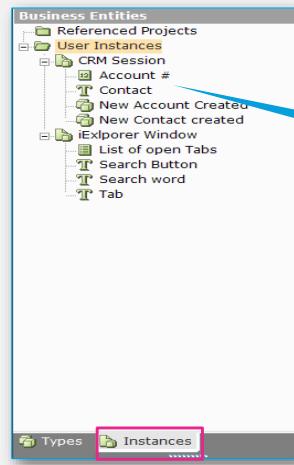
- Life cycle definition will be available only for composite BEs, not primitive. Define:
- Construct on Project load (from that point on this BE's value may change when you assign it due to an Assignment anywhere in the Solution) OR
- Define when to construct and destruct when a certain Event occurs
- Use the Life Cycle definition carefully, construct BEs and destruct them only when needed.
- Constructing unnecessary BEs can result in using up memory space
- Destructing unnecessary BEs can result in data loss

Representing Screen Elements in Business Entities

- All Screen Element Functions that will be used in the Business Solution must be constructed as BE Functions.



Create a Type for
every application
that appears in the
Business logic



Create an Instance
for every Type

NICE®

- **In the Types tab** - Define Functions' Parameters, and Return Type. Create Parameters for Events.
- **In the Instance tab** - For every Instance define Data flow, Life Cycle, and Events Parameters values.

Summary

- Creating Functions and Events
- Working with data structures



- Our Solution is built from Projects, each represent a layer. In order for them to communicate Project References are added
- Assign BE Properties to hold a pointer to the SE as a data source in real time
- **Functions** perform an action, and can return a value. They are built from a list of Instructions:
- **Function Invocation** invokes a call to another Function
- **Local Variable** allows you to use a local variable within the Function
- **Assignment** allows you to assign a value
- **Return** instruction stops running the Function. It may return a value
- **If, For and For Each** allow you to logically handle conditional and loop operations on BEs and SEs
- **SE Events** are raised when a change in the Screen Element occurs. You can create a BE Event to capture certain data from the screen when the Event occurs



Thank You





BUSINESS LOGIC LAYER



Lesson Objectives

By the end of this lesson you will be able to:

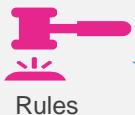
- Create Rules and nested Rules
- Create Event Handlers

NICE®

Business Logic Layer

- The Business Logic Layer enables you to define the logical entities that determine the behavior of RT Client according to your business requirements.

Business Logic



Rules

Rules define actions to take as a result of defined conditions



Event Handlers

Event Handlers define the actions to be taken as a result of Events that occur

Decision Support

Business Entities

Physical Objects



Project References Connect Layers

- Our Solution is built from Projects, each representing a Layer
- Instead of creating one big complex Project, Projects reference one another to create a structured and organized logic

Business Logic

Decision Support

Presentation

Business Entities

Physical Objects



NICE®

- In order for Project A to reference Project B, Project B must have public elements.
- Always add the lower layer reference to the upper layer Project.



Notes from Demo

DO IT YOURSELF

Project References

- Create a Business Entities Layer Project
- Create a Business Logic Layer Project
 - Reference the Business Entities Layer Project you created

NICE®

What are Rules?

- A Business Rule is comprised from a Condition and Actions.

The Condition is evaluated whenever the VIP Property of a Client is modified



VIP = True

Logical Condition

New Client is VIP

VIP = False

TIME

When the Condition is fulfilled, the Rule is turned on

ON

When the Condition is no longer fulfilled, the Rule is turned off

OFF

Action to trigger when a rule is turned on



Action



Action*

*Optional

NICE®

- When Jeff becomes a VIP client, the VIP property's value is modified, and so the RT Client will check if the condition is fulfilled. If so, It will trigger the green Action. When Jeff is no longer VIP, the rule will not be fulfilled anymore, and the red action, if defined, will be triggered.

How to Add a Condition

- A Condition is a logical expression, built from:



Item

Operator

Value

- For example:



- When you add a second (or higher) Condition, Designer will automatically add the **and** operator. By clicking it you can switch it to **or**.

Instruction Types

- An Action is a set of Instructions, executed top-down.
- **Function Invocation** – Call another Function
 - Drill down to the Function you want to invoke from the menu:

!

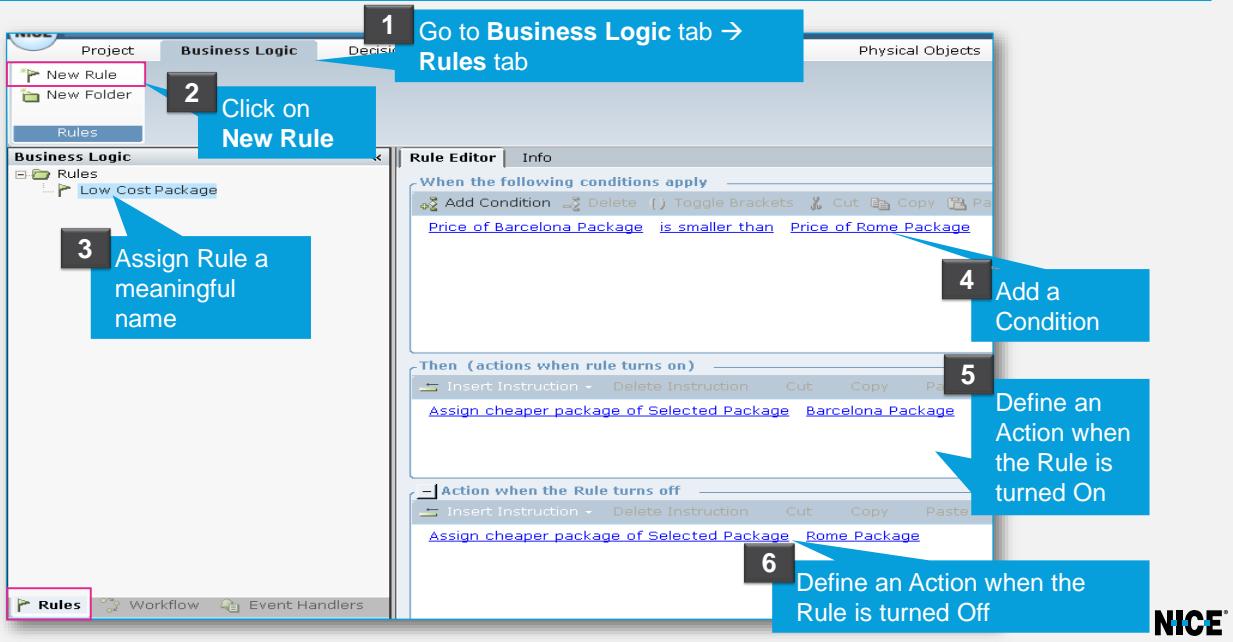
[Select Function...](#)

BE Functions	Library Objects Functions	User-Defined Functions
Add to List	Copy text to Clipboard	
Duplicate	Get Active Process	

NICE®

- **Business Entities** – drill-down to the BE Instance, then Functions you can invoke on that Instance. It can be either a Function you created or a built-in Function like sorting a list.
- **Library objects** – A built-in Functions library, much like Excel. Drill-down menu will display the group Functions (Math Functions / Date and time Functions etc.), and then Functions you can invoke in this group (Sum, Current date etc).

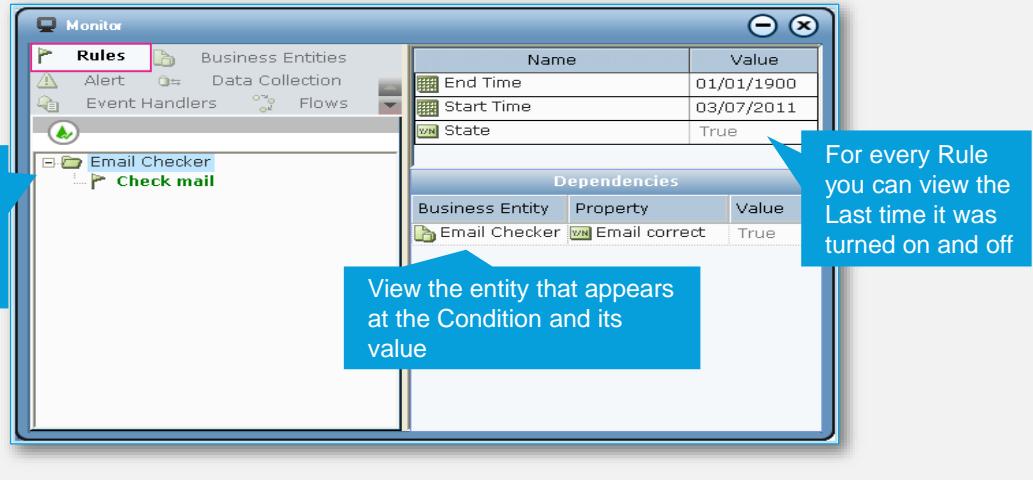
How to Create a Rule



- **It is important to differentiate between a Rule and an IF statement:**
- This example DOES NOT mean that if the Rome package's price is higher, then Rome will be assigned as the selected package.
- It DOES mean that as long as the Barcelona package's price is lower than Rome package's price, Barcelona will be assigned as the selected package. When this is no longer true, the Rome will be assigned as the selected package.

How to Monitor Rules

- The Rules tab presents all properties relevant for a Rule within a single view.



NICE®

- The Rules tab in the Monitor window displays the Rules currently defined in Designer.
- Start and End time** - Only the most recent times are shown.
- The Dependencies section (Bottom section in the right pane) lists the dependencies of the Rule, i.e. those properties on which the Rule is dependent. Property values can be changed as needed to monitor the effect on the Project.



Notes from Demo

DO IT YOURSELF

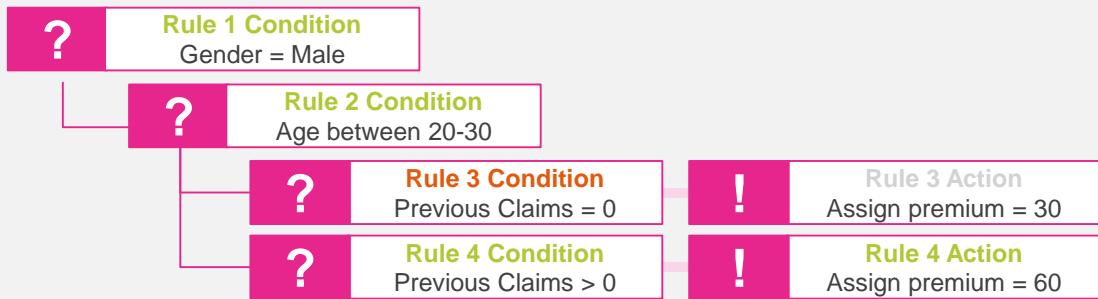
Creating and Defining a Rule

- Create a ‘Customer’ Type and Instance with the following properties:
 - ‘IsSenior’ (Boolean), initialized to FALSE
 - ‘Message to Display’ (Text)
- Create a Rule which assigns the message “Customer is not senior” to ‘Message to Display’ if ‘IsSenior’ is FALSE, and assigns “Customer is senior” otherwise
- Test your Solution using the Monitor
- Reverse your Rule to assign the message “Customer is senior” to ‘Message to Display’ if ‘IsSenior’ is TRUE, and assigns “Customer is not senior” otherwise
- Retest your Solution

NICE®

Nested Rules Example

- Rules can be nested under other Rules. A nested Rule's Condition will be evaluated only if its parent's Condition is TRUE
- Use nested Rules to define complex logic.
- An example: What is the premium value for a 25 year old male with 2 previous claims?



NICE®

- In order to nest a Rule under another Rule in the Designer, simply drag the “child” Rule to the “Father” Rule



Notes from Demo

DO IT YOURSELF

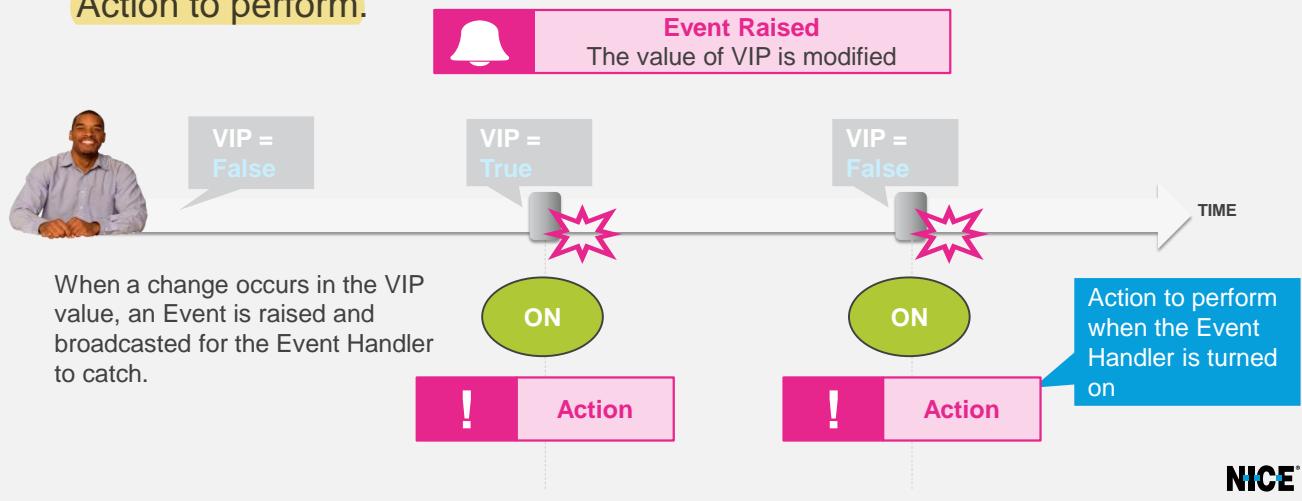
Nested Rules

- Create the following properties in the 'Customer' Type:
 - Gender (Text)
 - Age (Number)
 - Previous Claims (Number)
 - Premium (Decimal)
- Create set of nested rules that reflects the example on the previous slide
- Test your solution with Monitor

NICE®

What are Event Handlers?

- An Event Handler is comprised of an external Event that occurs in a target application, or any internal Event thrown in the Project, and an Action to perform.



- The difference between a Rule and an Event Handler is the triggering mechanism – A rule is triggered by a Condition, while an Event Handler is triggered by an Event. The use of Rules and Event Handlers depends on the Business Logic scenario and the simplicity of implementation.

Rules vs. Event Handlers - Summary

	Rules	Event Handlers
Structure	Condition and action	Event and action
Actions' are triggered when...	1. Condition is fulfilled 2. Condition is no longer fulfilled	Event is raised
?		

Select the appropriate Business Logic element (Rule or Event Handler) for each of the following scenarios:

- For smoking customers, life insurance premium is 30% higher than the regular rate.
- On their 60th birthday, life insurance premium is increased by 50%.
- Calculate the Customer's Disability insurance premium (always 1% of the salary) whenever the salary changes.

NICE®

- **1st scenario:**

- We would use a rule to implement, since customers' smoking status can change and the premium should change accordingly. If someone starts smoking, their premium is increased by 30% but if they stop smoking, then their premium should be set to the regular rate (action when the Rule is turned off).

- **2nd scenario:**

- Customers only turn 60 once, so an Event Handler would be appropriate in this case.

- **3rd scenario:**

- Since the premium is directly linked to the salary, when the salary changes so should the premium (the Action calculation is always the same 1% of something) – classic Event Handler.

Associating Event Handlers to Events

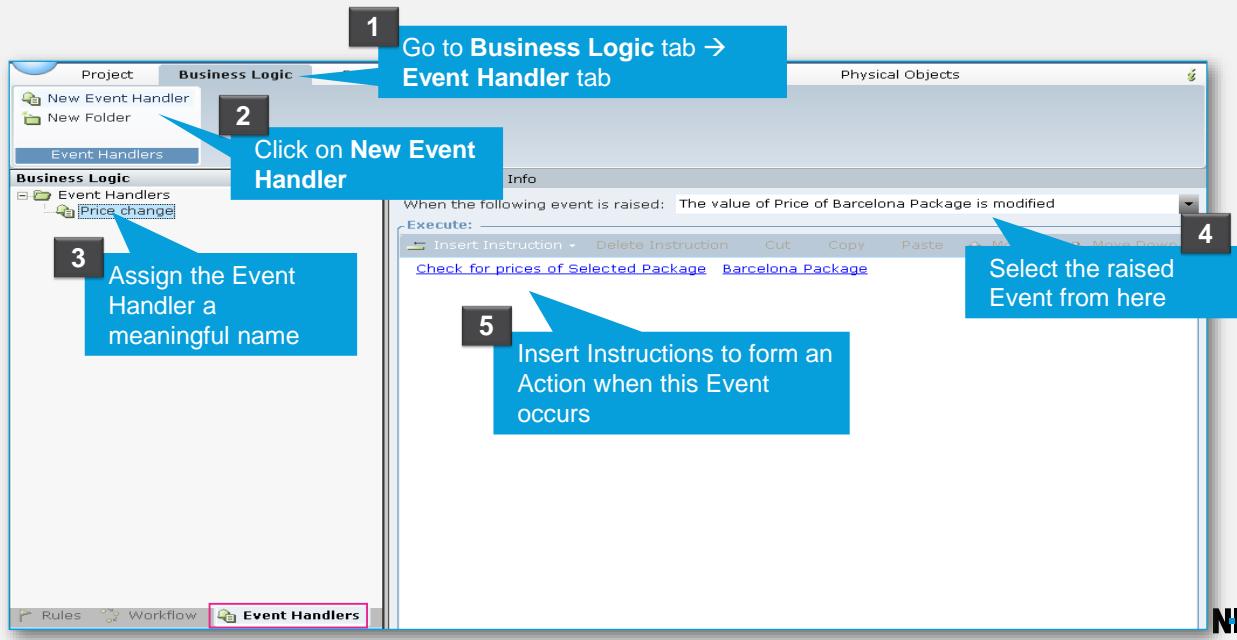
- There are various groups of Events. Here are a few examples:

BE Events	Library Objects Events	Rule Events
Value Modified	Client Shutting down	Turned off / on
	Solution Downloaded	Start time Modified

NICE®

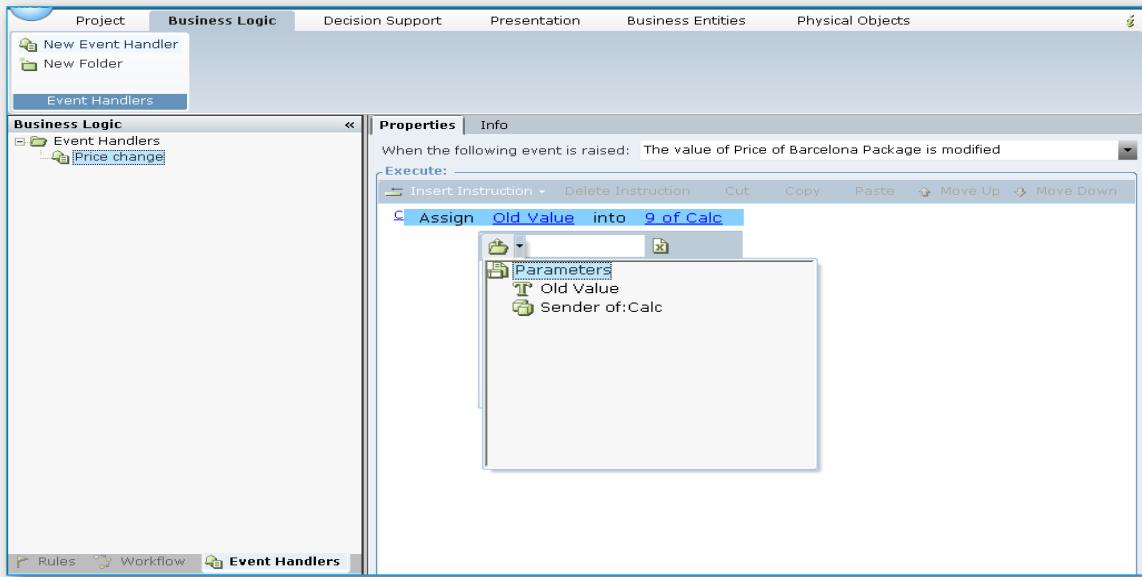
- Drill down the menu to Event group (for example BE events), then to the entity of which raised the Event, then to the Event itself.

How to Create an Event Handler



- If the Event Selected has parameters, then you will be able to select its parameters in the Execute section.

Business Entity Event Parameter: Old Value

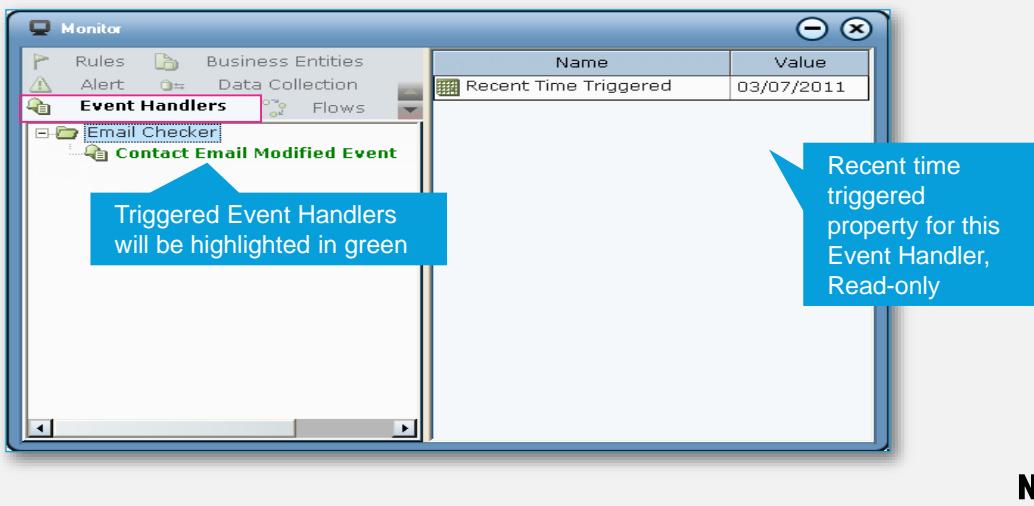


NICE®

- The Old Value parameter enables you to access the old value of a Business Entity Instance, before it was changed. This parameter is passed with the 'Is Modified' Event of a primitive Business Entity Instance.

How to Monitor Event Handlers

- The Event Handlers tab presents all Event Handlers defined in the Designer within a single view.



- If an Event is triggered while this tab of the Monitor window is open, the Event Handler in the Navigation pane **flashes green for five seconds**, in order to indicate that this Event is alive.



Notes from Demo

DO IT YOURSELF

Event Handlers

- Create an Event Handler that does the following:
 - When the ‘Age’ Property is modified:
 - If the Age > Old Value, assign “Older” to ‘Message to Display’
 - If the Age < Old Value, assign “Younger” to ‘Message to Display’

NICE®

Summary

- Creating Rules and nested Rules
- Creating Event Handlers

NICE®

- A Business Rule is comprised from a Condition and Actions. The Condition is a logical expression, evaluated every time an entity in it changes value. An Action is a set of Instructions, executed top-down.
- Rules can be nested under other Rules. A nested Rule's Condition will be evaluated only if its parent's Condition is TRUE.
- Event Handler specifies the Action to perform when an external Event occurs in a target application, or any internal Event thrown in the Project.



Thank You



The NICE logo consists of the word "NICE" in a bold, black, sans-serif font. A horizontal bar with three blue segments of increasing length is positioned above the letter "I".

NICE®

WEB SERVICES



Lesson Objectives

By the end of this lesson you will be able to:

- Describe the concept of Web Services
- Utilize SOAP services
- Utilize REST services

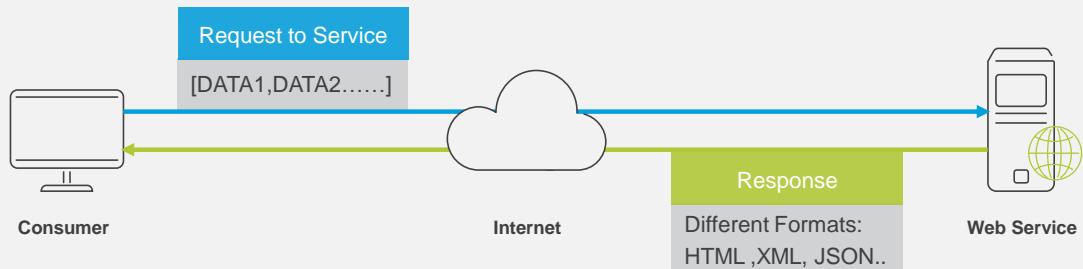


Web Services



NICE®

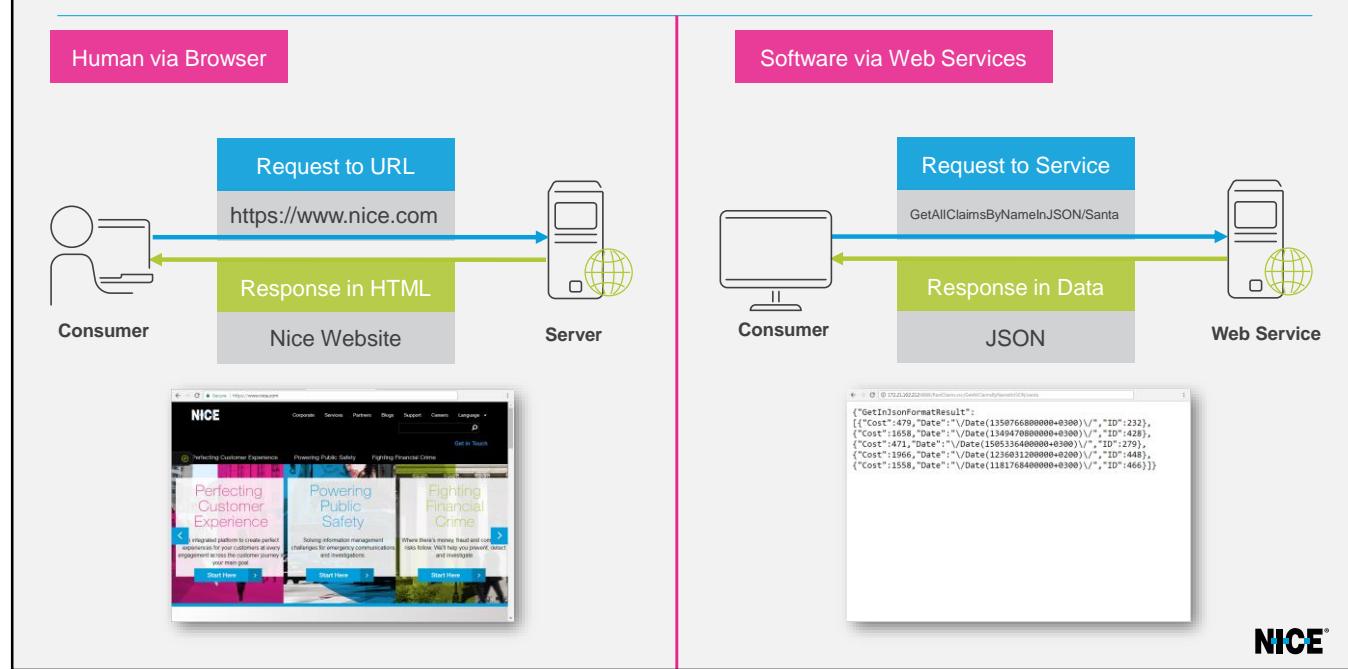
What Is Web Services?



NICE®

- Web services is a technology for transmitting data over the Internet and allowing programmatic access to that data using standard Internet protocols.
- Web services provide a way to communicate between applications running on different operating systems, with different technologies and programming languages.
- Only data is transmitted using Web services technology (web services do not have GUI).

What Is Web Services?



- Using Web service is similar to browsing- When a person wants to connect to Nice web site, he is in fact sending a request to the server just by typing the website address in the URL bar and the server response is the website(HTML).
- When we are using web services the consumer is not a person but a software that can also use the URL bar to send a request. In this example the software is requesting to get all Santa's claims in JSON format and the response is five claims, presented in JSON format(this example is using REST services)

Web Services – Usage

- Using Public services (free or paid) to consume updated data:



Stock Markets



Exchange Rates



Currencies



Locations



Schedules



Social Media

- Using third-party systems API for getting data and/or for integration:



Organizational Services



CRM's API's



Customers Authentication

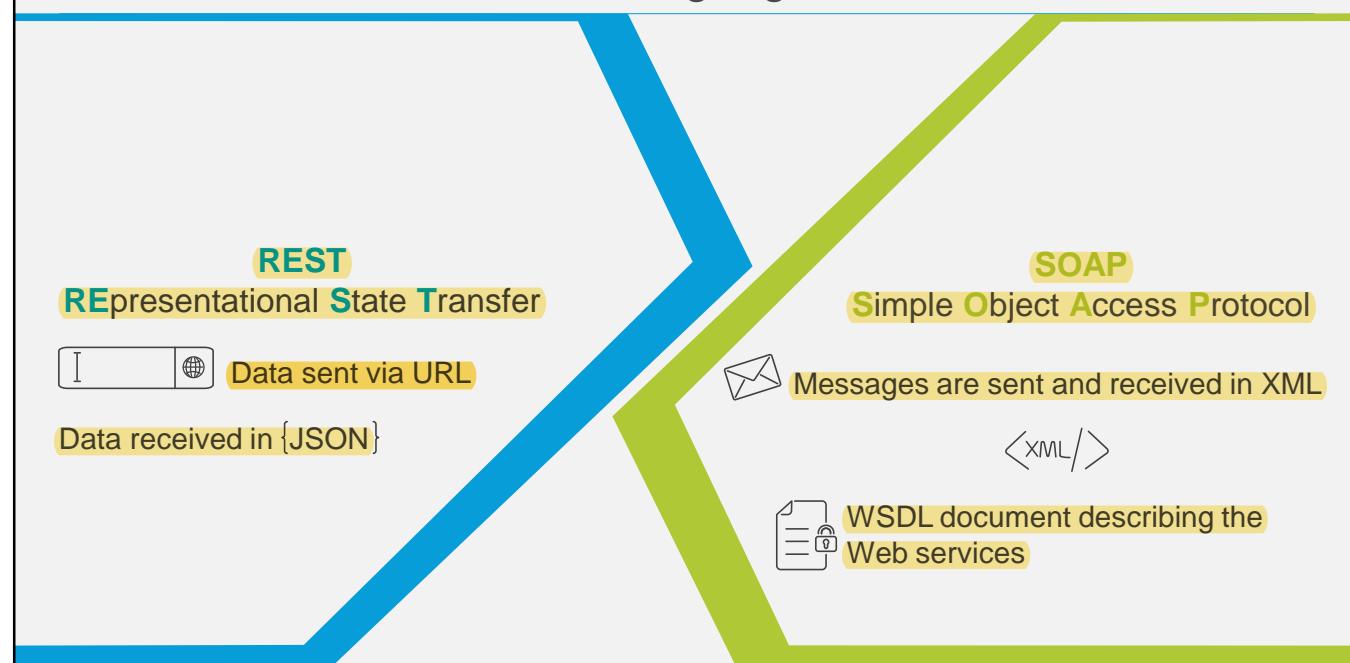


Customers Information

NICE®

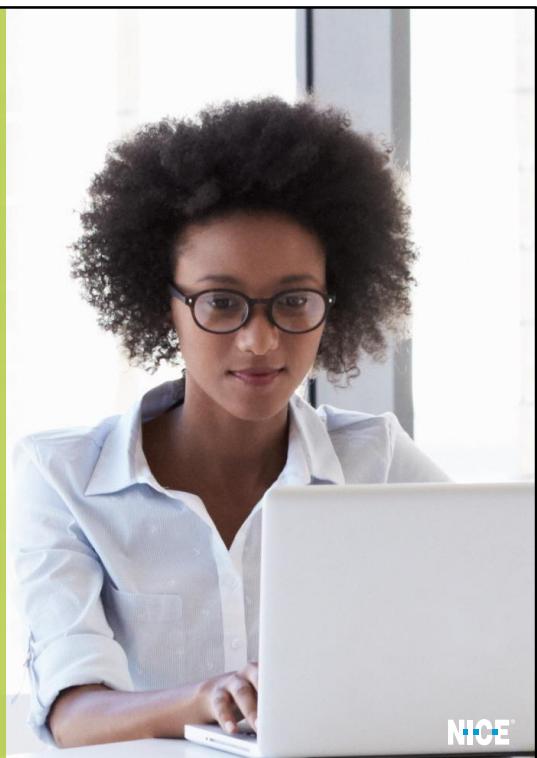
- For example, Company A provides a weather Web service and Company B uses this Web service to provide weather information to its customers. Company C may combine Company A's Web service with its own Locations Web services and offer it as a Web service to other companies.

Web Services – Common Languages



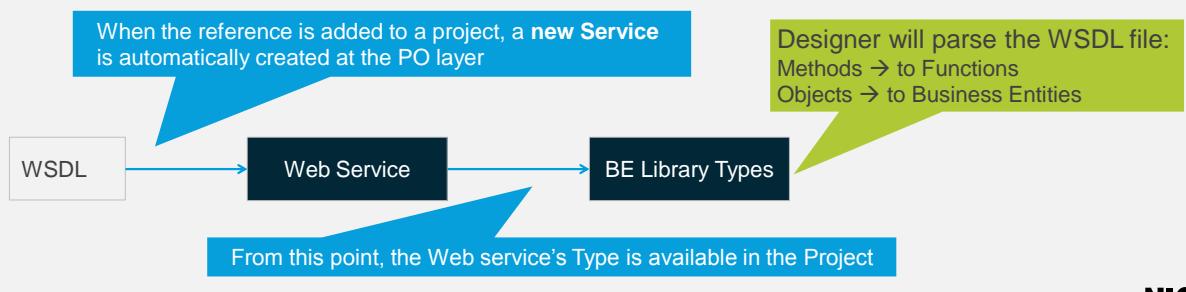
- SOAP and REST are two answers to the same question: how to access Web services
- SOAP relies exclusively on XML to provide messaging services
- REST rely almost exclusively on obtaining the needed information using the URL approach.
- Both types of services (SOAP and REST) can be used in a solution.
- A WSDL(Web Services Definition Language) is an XML document that describes a Web service. The content of the WSDL document defines the contract of the web service: what methods it has, how you must call them, what parameters it expects etc. you can use WSDL for SOAP but not for REST.

SOAP



Consuming Web Services In The Designer

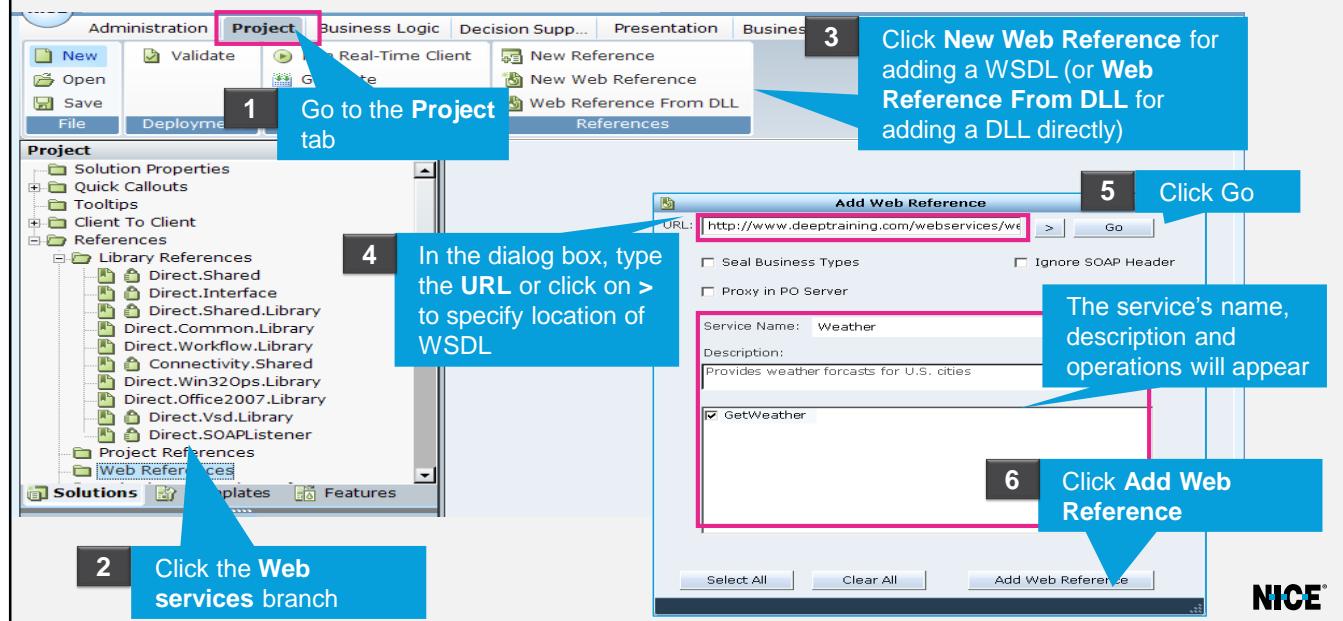
- The Real Time Designer enables you to import a web reference of a WSDL document.
- The file can be:
 - Local – specify location on server
 - On a reachable location on the net – specify URL of WSDL



NICE®

- After adding the WSDL reference, the designer will parse the XML and the relevant service functions will be available to the solution engineer in an upper layer.
- Service objects are used to enable client integration with services outside the Real-Time Designer system.
- Different Services will expose different functionalities, make sure you use them in the correct way
- The functions are available in 2 forms: Synchronous and Asynchronous (depends on the service functionality)

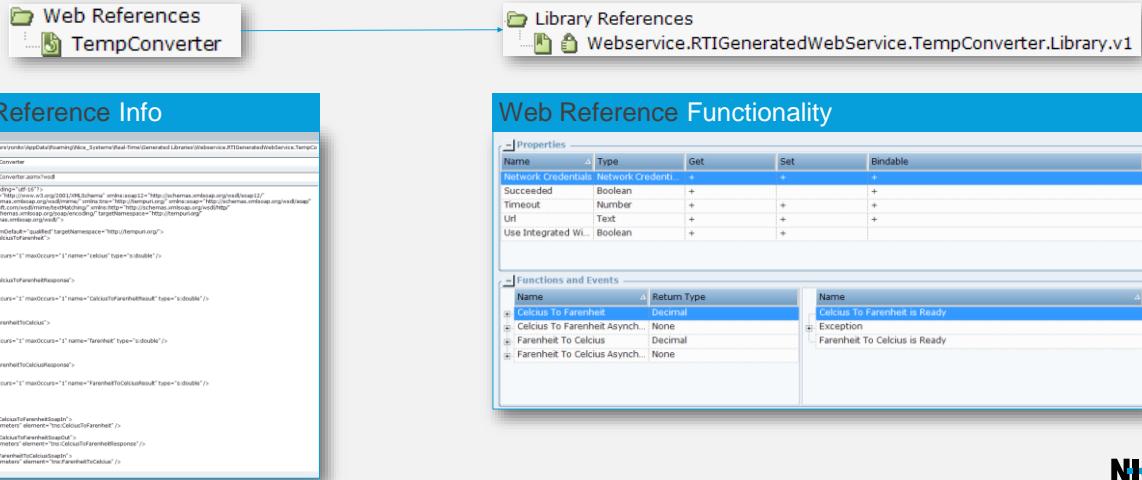
How To Add SOAP Web Service Reference



- Selecting **Seal Business Types** prohibits being able to set web-service-originating composite business entity types as a base type of another business entity type or as the type of a business entity instance.
- Some web services require composite parameters, rather than a list of textual parameters. In such cases, composite business entity types are created for the WSDL import process.
- When you select this checkbox, the web service originating composite business entity types are only available via local variable declaration within a rule/event handler/workflow step action or a business entity type function.
- Selecting **Proxy in RT Server** connects all RT Clients to the web service via the RT Server. When this checkbox is selected, the RT Server creates a proxy that is used by all RT Clients to connect to the external web service.
- Select **Ignore SOAP Header** if you do not want a generated SOAP header in the request sent to the service.
- By default, all operations are selected.
- Clear the checkboxes of the operation you do not want selected, or click the **Clear All** button to clear the checkboxes for all the operations. You can click the **Select All** button to reselect all the operations.
- Clicking the **Add Web Reference** button automatically performs the following operations transparently:
 - Generates a DLL that serves as the wrapper for accessing the web service.
 - Makes a reference to this DLL.
 - Generates a new service object in Services tab that is assigned to the specific type of service for which the DLL was created, in order to provide access to that service.
 - This means that you do not need to complete the references and then manually create the new service, as it is all done automatically for you.

Web Services Library

- After adding a Web Reference, the Designer will create a DLL file containing the types and methods provided by the Web service.

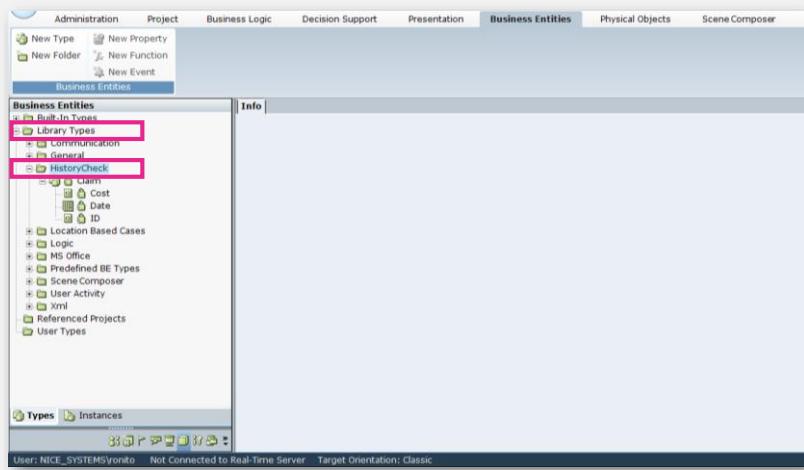


NICE®

- The Web reference Info can be found in the Project Tab → References → Web References
- The Web reference Functionality can be found in the Project Tab → References → Library References

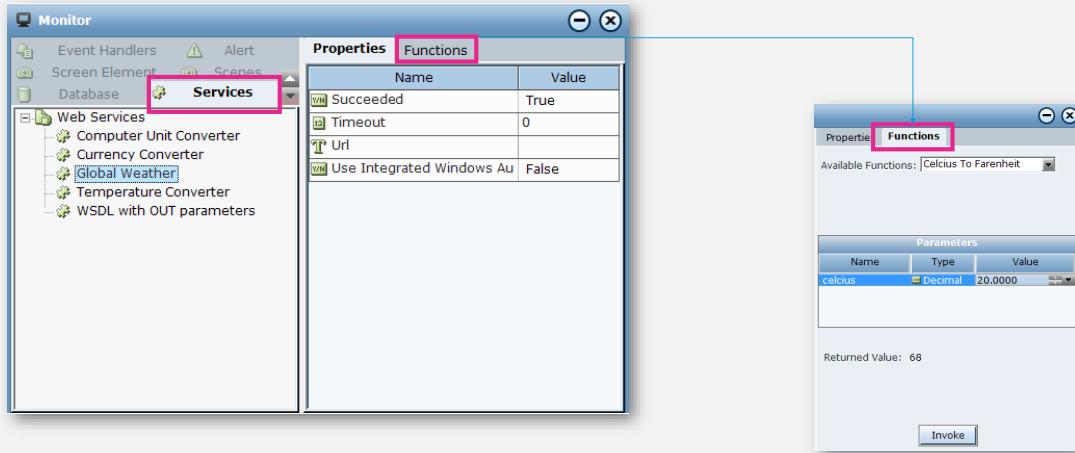
Web Services Library Types

- If the Web service that you added is using objects, you will find them in the Business Entities Tab → Library Types



Monitoring Web Services

- The **Services** tab in the Monitor window enables you to monitor and test Web services defined in a solution:



NICE®

- Succeeded:** Indicates whether or not the latest run of this service succeeded (True) or failed (False) when last called by the project.
- Timeout:** The time that Real-Time Designer waits for the web service to complete.
- Url:** The URL of the web service. By default, the URL of the imported web service definition language (WSDL) is specified.
- Use Integrated Windows Auth:** Indicates whether or not to send Windows credentials in the web service consumption.



Notes from Demo

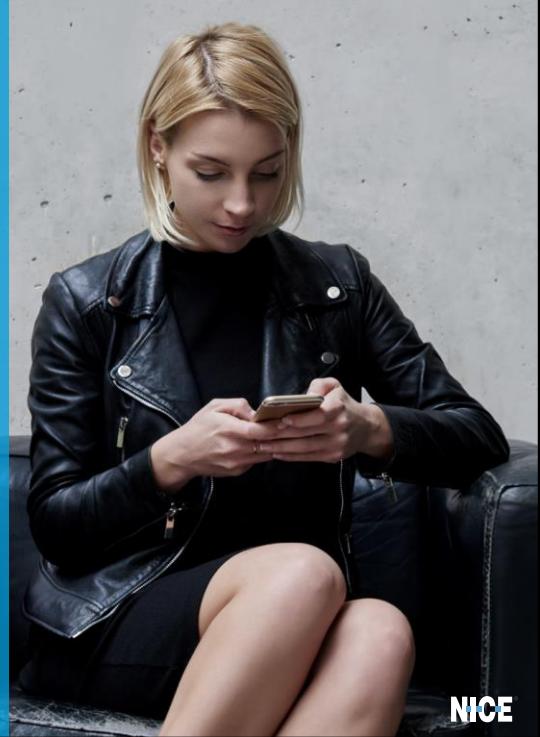
DO IT YOURSELF

SOAP Services

- In RTD1 machine open Web Browser go to following URL:
<http://172.21.102.212:80/soap/Converter.asmx?wsdl>
- Check Service is reachable and WSDL received.
- Add new Web Reference to the same URL.
- In the Physical Object tab rename the service “TempConvert”
- Create a BE with 2 properties:
 - TempCelsius
 - TempFahr
- Create logic to assign the “Fahrenheit To Celsius” Function to TempCelsius of the “TempConvert” Web Service, whenever the “TempFahr” property is modified.
- Perform similar actions for the “TempFahr” property.
- Open the Monitor tool and enter different temperature values in each of the properties to see how the data changes in the other property.

NICE®

REST



REST Web Services

- REST services parameters are usually transferred as strings via URL



- Response is usually plain text in JSON format

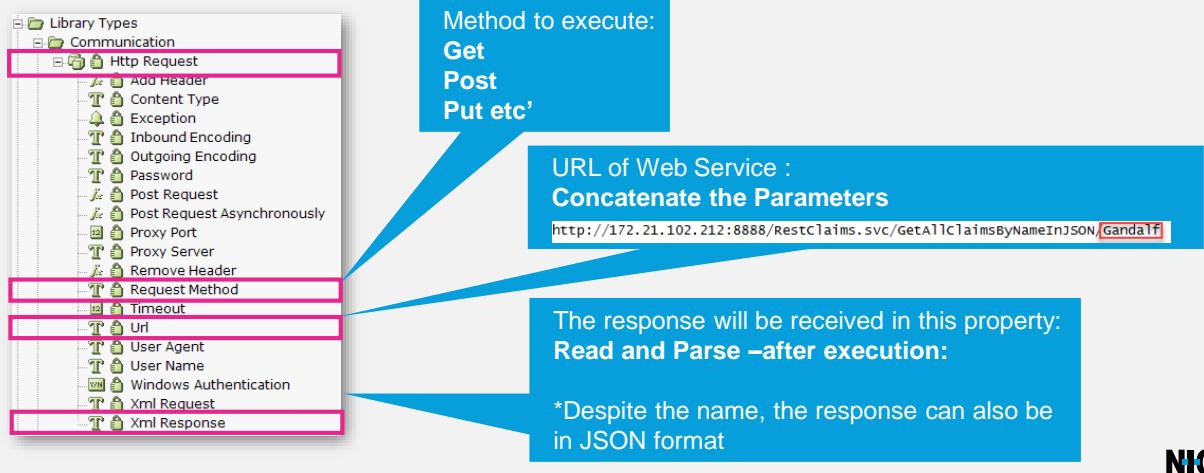
```
{"GetInJsonFormatResult": [{"Cost":2468,"Date":"\\Date(152572680000+0300)\\","ID":415}, {"Cost":2518,"Date":"\\Date(126030960000+0200)\\","ID":439}, {"Cost":2840,"Date":"\\Date(1193000400000+0300)\\","ID":191}, {"Cost":2135,"Date":"\\Date(1209416400000+0300)\\","ID":882}, {"Cost":2233,"Date":"\\Date(1270933200000+0300)\\","ID":991}]} 
```

- If Additional parsing is required, you may use Regular Expressions library

NICE®

Invoking Web Services Using HTTP Request – REST Example

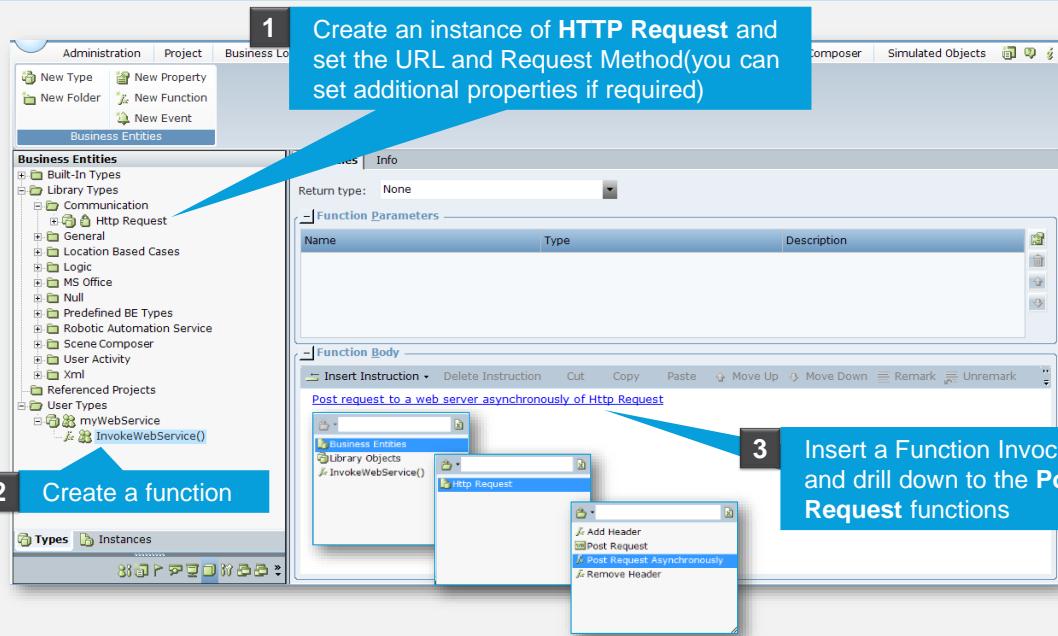
- To execute a web request, we usually use 3 main properties of the “HTTP Request” type.



NICE®

- REST can use four different methods (GET, POST, PUT, and DELETE) to perform tasks.
- If you need to use parameters in your request, you need to concatenate them in the URL property

How to Invoke Web Services Using HTTP Request



NICE®

HTTP Request Functions And Monitoring

- All functions can be executed and tested through the Monitor

The screenshot shows the 'Monitor' window with the 'Functions' tab selected. Under 'Available Functions', the 'Add Header' function is highlighted. Below it, the 'Parameters' table shows two entries: 'Header' (Type: Text) and 'Value' (Type: Text). A tooltip for 'Add Header' states: 'Adds a header to the request'. Another tooltip for 'Post Request Asynchronous' states: 'Execute the request asynchronously, for time consuming requests'. A note below says: 'You can use the HTTP Request event to verify if the request succeeded:'. A small icon indicates: 'The value of Xml Response of Http Request was modified'.

Adding or Removing Header information

Executes the request and waits for results
Returns True or False

Execute the request asynchronously,
for time consuming requests

You can use the HTTP Request event to verify if the request succeeded:

The value of Xml Response of Http Request was modified



REST Services - Online Lesson

For more details on invoking REST Services, please refer to the [Online Lesson Available in Nice University](#)

NICE®

- Header is an optional element that can contain some extra information to be passed to the web service.
- In some cases, you may need to pass some additional information in the Header element of the HTTP request.
- For example: pass authentication information to the web service or pass information on how the body contents should be processed by the web service



Notes from Demo

DO IT YOURSELF

REST Services

- In RTD1 machine open Web Browser go to following URL:
• <http://172.21.102.212:80/rest/getclaims/bylastName?lastName=>
Check if Service is reachable and results received.
- Create a BE RequestProvider of type HTTP Request:
 - Add new property and name it baseURL
 - Assign the Web Service URL to baseURL property
 - Add new property: searchFor
 - Provide Initial value for Request Method: GET
 - Create a function: GetRequestURL-it will return concatenation: baseURL+searchFor
- Create EH(on searchFor modified):
 - Assign the GetRequestURL return value to URL of RequestProvider
 - Execute the Request
- Run Project -> Open Monitor -> check your Logic

NICE®

Summary

- The concept of Web Services
- Utilizing SOAP services
- Utilizing REST services

NICE®

- You can import a working Web Services Description Language (WSDL) file into the Designer, to be used in the project for accessing and retrieving information in a non-GUI way
- Available services :
- **SOAP**
- 1) TempConversion service:
- <http://172.21.102.212:80/soap/Converter.asmx?wsdl>
- 2) HistoryCheckService
- <http://172.21.102.212:80/soap/HistoryCheck.asmx?wsdl>
- **REST**
- <http://172.21.102.212/rest/getclaims/bylastName?lastName=John>



Thank You





PRESENTATION LAYER



Lesson Objectives

By the end of this lesson you will be able to:

- Launch documents in context of an application
- Describe the use of Callouts
- Create, design and preview a Callout
- Add content to a Callout



Presentation Layer

Defines the items that are displayed on the agent's screen.

Business Logic

Decision Support

Presentation



Callouts



Launch in Context

Business Entities

Physical Objects

NICE®

- Presentation layer is the first operative layer in your solution.
- It allows to define elements that are displayed on the agent's screen. In upper layers you will define what triggers the Presentation to display.
For example:
 - When a certain event in the system occurs.
 - If a certain condition is TRUE.

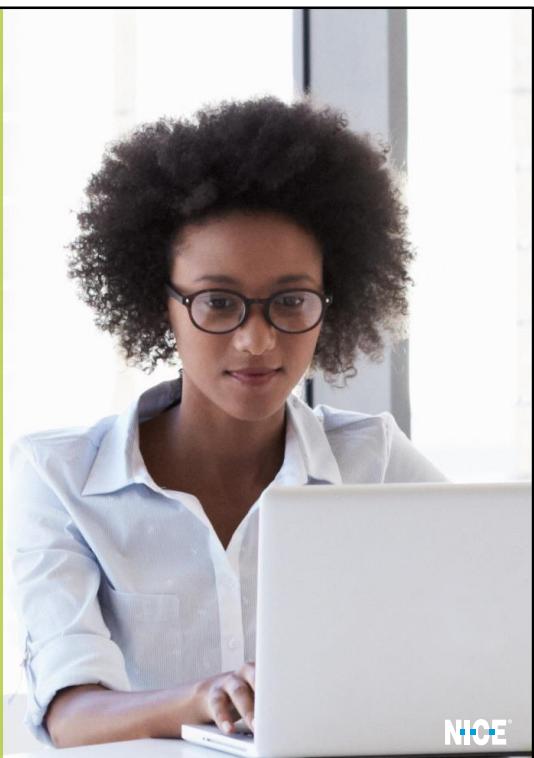
LIVE DEMO

Project References

- Open a new project for your Business Entities Layer
- Open a new project for your Presentation Layer and reference the BE Layer project

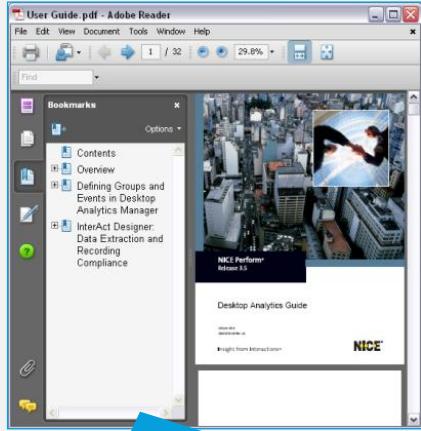
NICE®

Launch In Context (LIC)

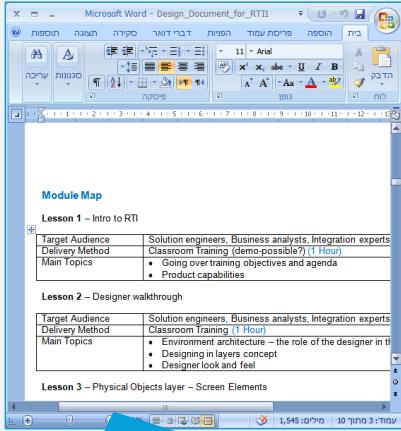


Launch In Context

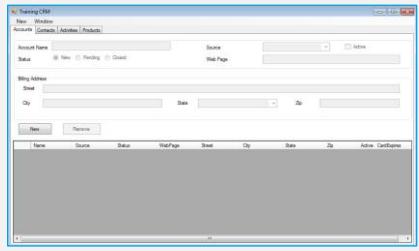
- Launch in Context enables to display a document in the context of its standard application



.pdf file in
Adobe Reader



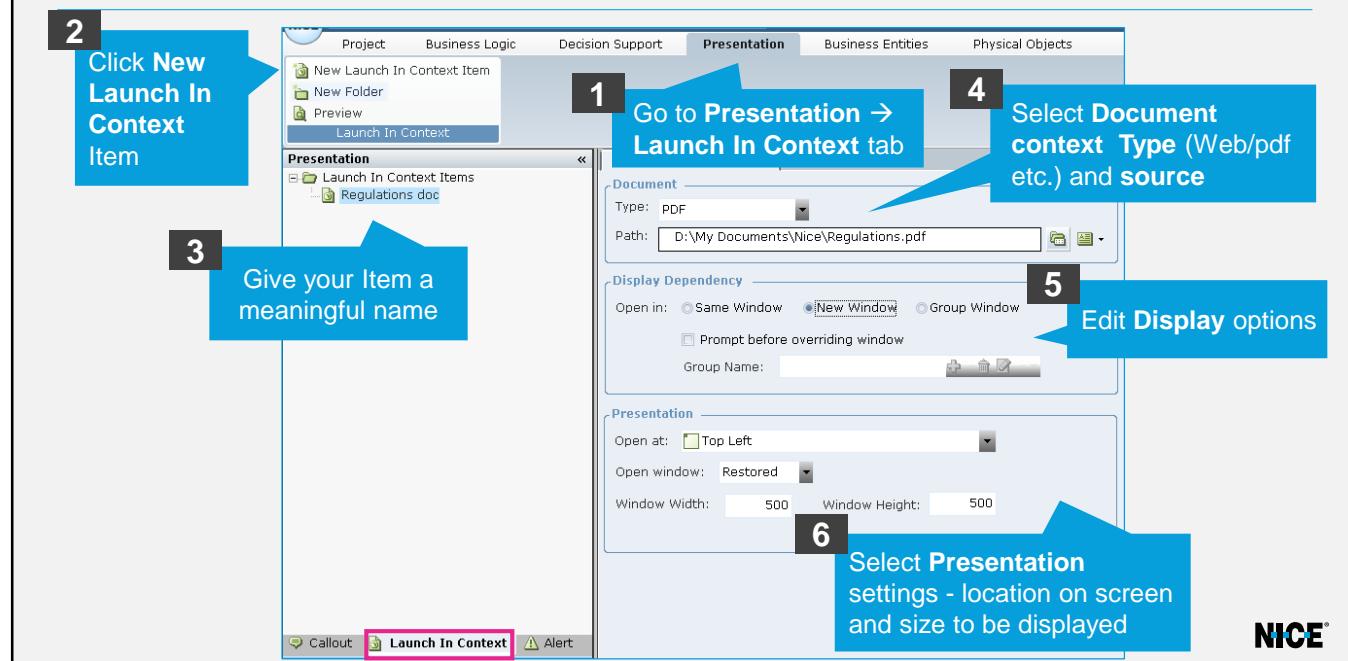
.doc file in
Microsoft WORD



CRM application

NICE®

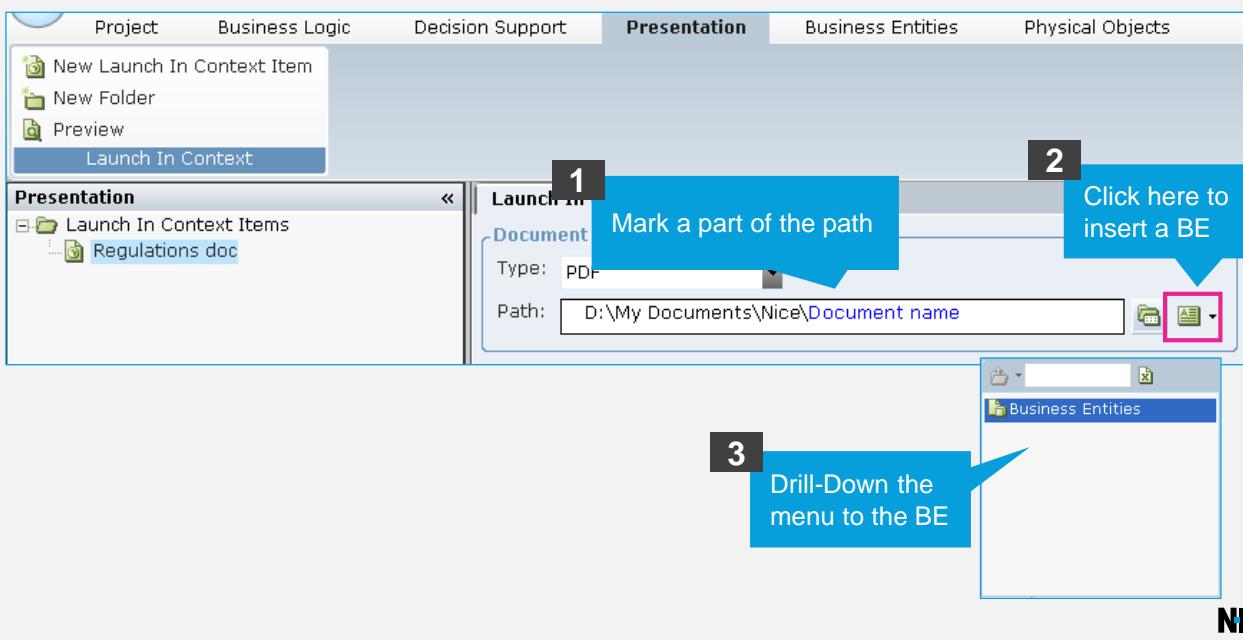
How to Create LIC Items



Step 5 notes:

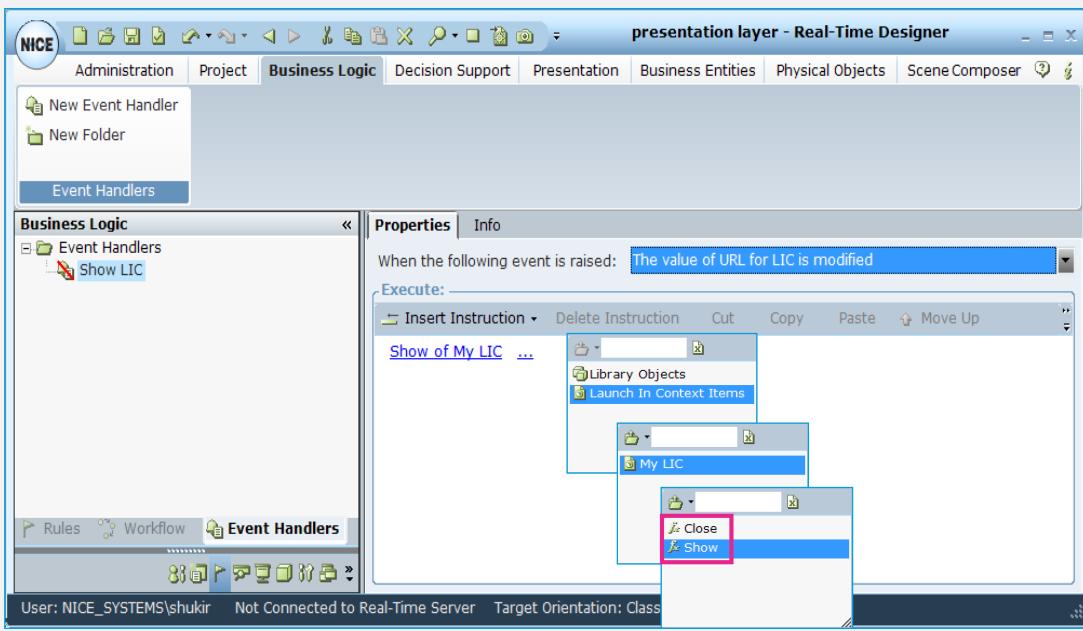
- **Open in** options allows to define the presentation of an LIC item, if the same item is opened more than once:
 - **Same window** - Open the LIC in the same window
 - **Open in New window** - Open the LIC in a new window
 - **Open in Group window** - Open a few LIC items in the same window if they belong to the same group
- **Prompt before overriding window** - relevant if the Same window option or Group window were selected. Checking this box will force an Agent to confirm the display of this new information in the same window before overriding the window's content.
- **Group Name** - enabled only if the Group window option is selected. You have options to add\delete\select an existing group.

How to Create LIC Items



- An LIC item's path (or part of one) can hold a dynamic value so that the file launched can change in real time.
- This can be defined by selecting the path (or part of it), clicking the button and drilling down to a Business Entity, so the Business Entity's value will be taken according to the logic implemented in the solution.

Launching



- There are two functions associated with LIC items: Show and Close.
- These functions can be used to open and close an LIC item according to Business Logic.

LIVE DEMO

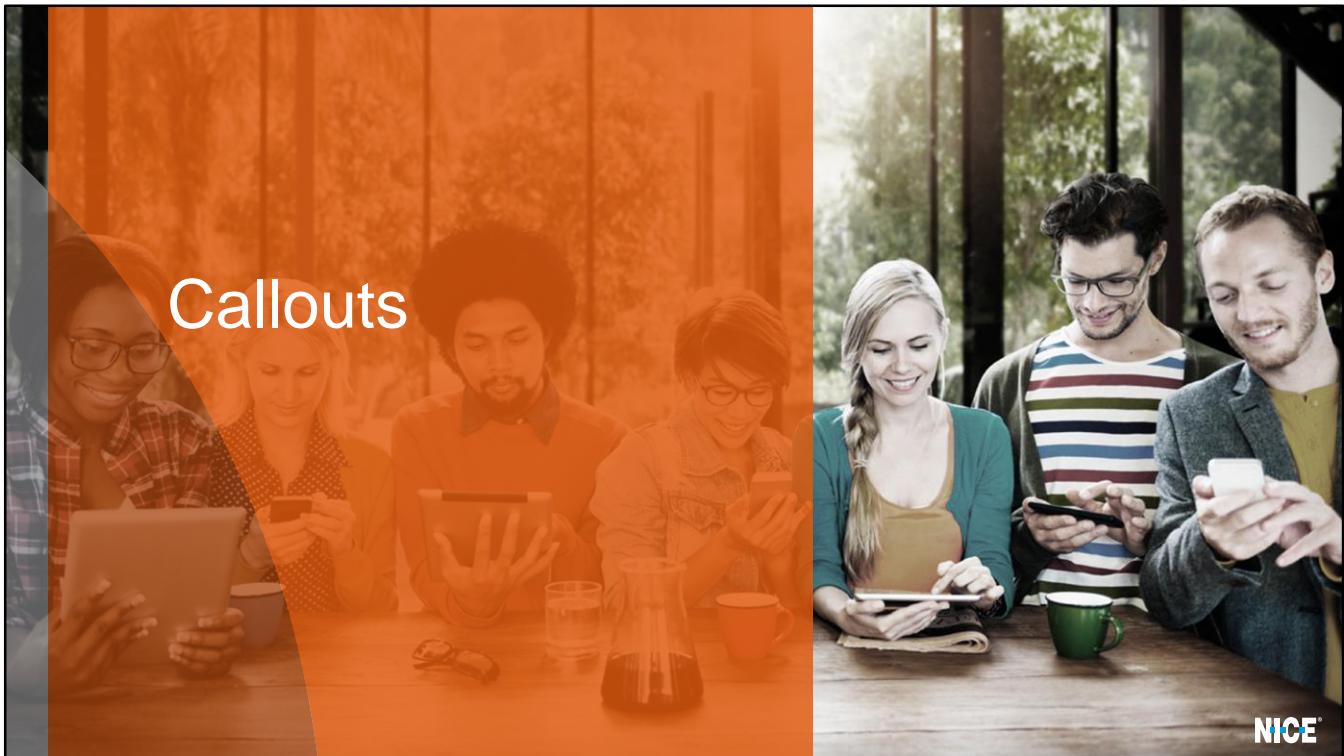
Creating Launch-In-Context Items

Create a new LIC item for www.nice.com
to be opened as follows:

- Open in a New Window
- In the Top-Right of the screen
- Restored with Width: 500 and Height: 400
- Show WITHOUT Toolbars and Menus

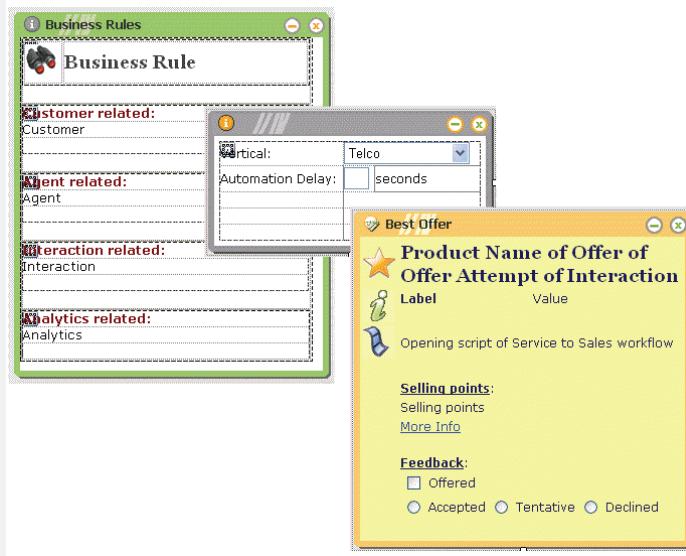


Callouts



Use of Callouts

- Best offer, reminder, survey, data completion, item description



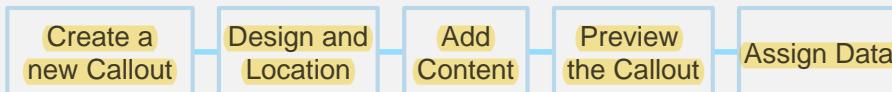
NICE®

- A Callout is a popup window that is displayed on an Agent's screen under certain conditions.
- A Callout can contain predefined text, images and dynamic data.
- RT Designer enables you to design the look, behavior and content of each Callout.

How to Design Callouts – Step by Step

- RT Designer provides a variety of tools to make the graphical and content design of a Callout simple

- We will follow this process flow:



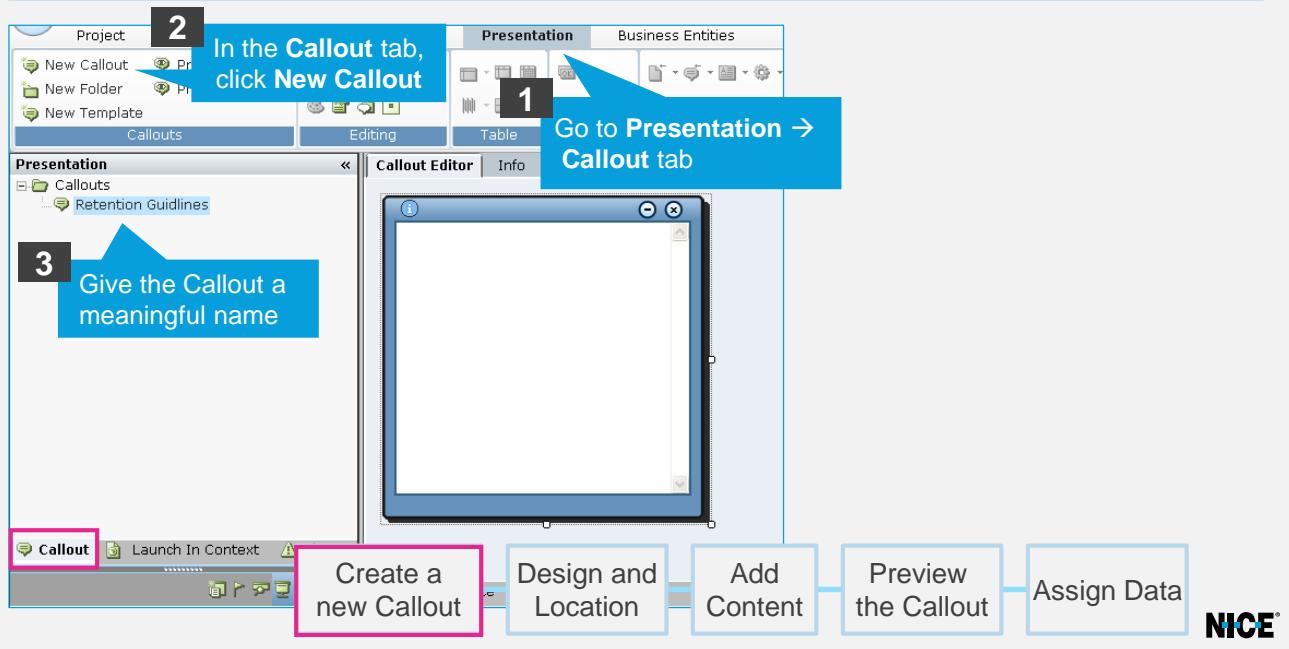
- The Callout should deliver a message in a short and targeted manner.
Keep it simple!

NICE®

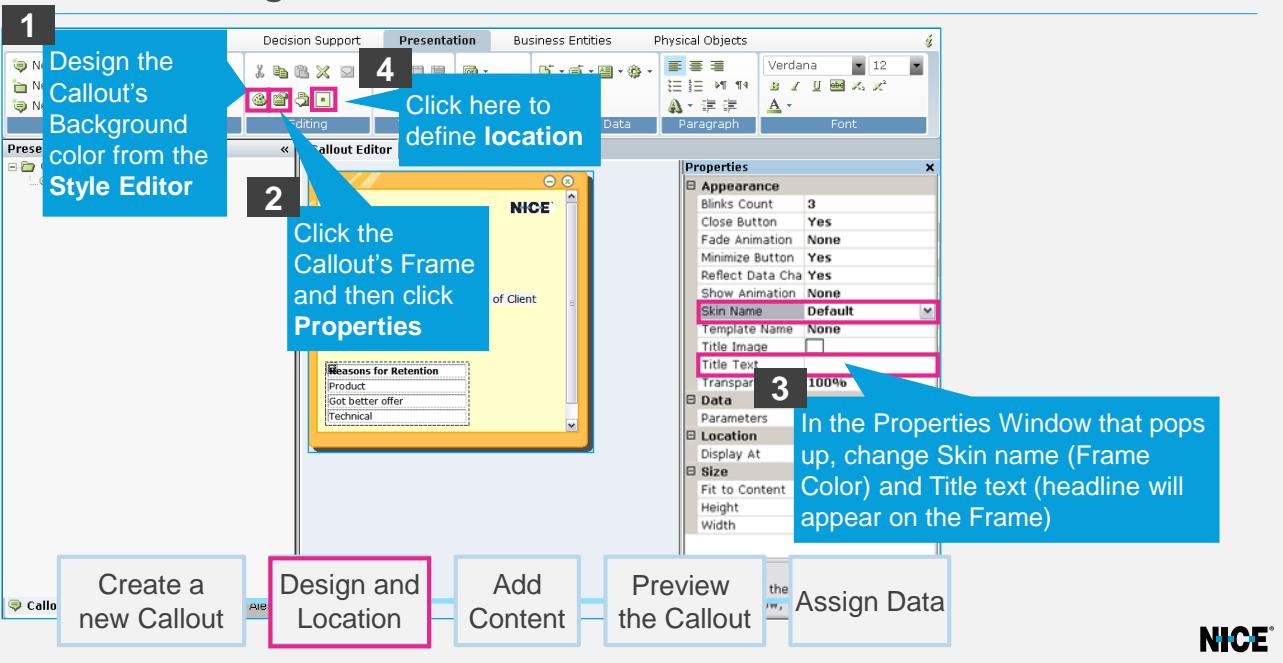
There are a few things to be aware when creating a Callout:

- Hiding essential fields / information by accident.
- Preventing the agent perform the task by covering buttons with a callout that would prevent the agent from clicking on it.
- Considering different habits / settings: full screen, resolution, Citrix, etc.

How to Create a Blank Callout



How to Design and Locate a Callout



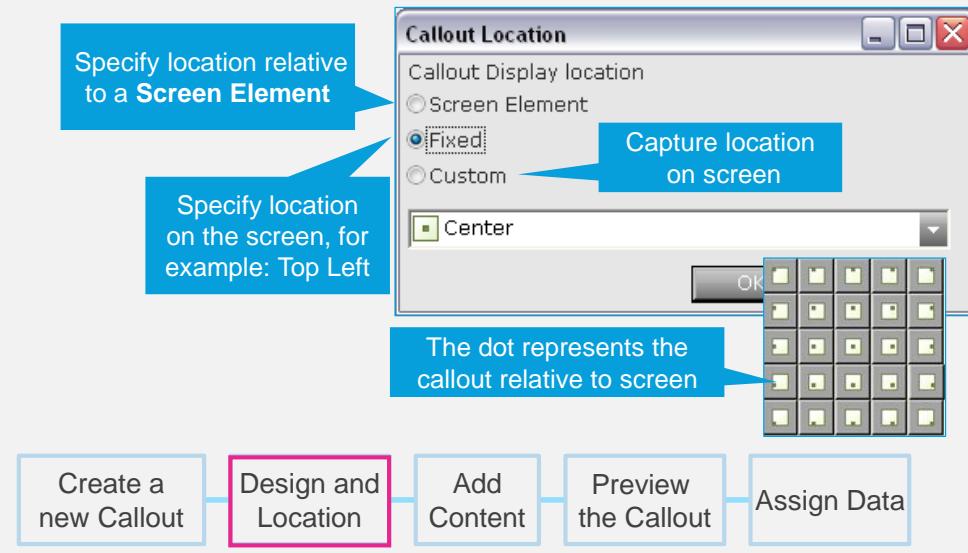
Different properties will be displayed for every element selected: body, frame, HTML object etc.

Editing tools:

- At Callout Properties define **bgcolor** = background color, can be found under Layout list
- At the Callout's frame Properties define **Skin name** = frame color, can be found under Appearance list

Callouts Design – Location on Screen

- Location defines the Callout's position on the screen



NICE®

Callout location should be in line with the Callout's message:

- If the message points to correcting a certain field, you should locate it next to the field (Screen Element).
- Locate all messages following a process or regulation in the same location, so the Agent will instinctively know where to go for action.

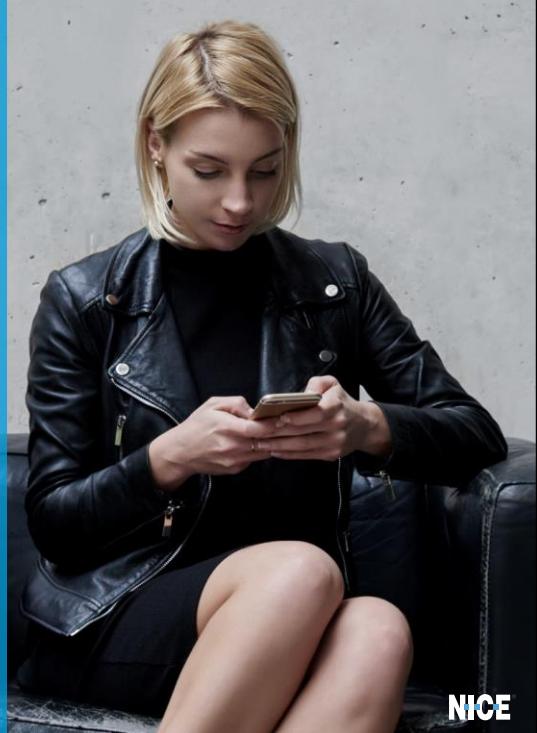
LIVE DEMO

Creating a Callout

- Create a New Callout, and name it “New Account”
- Define the “New Account” callout as follows:
 - Skin Name: Ocean Skin
 - Title Text: “Add New Account”
 - Location: Fixed: Center – Right

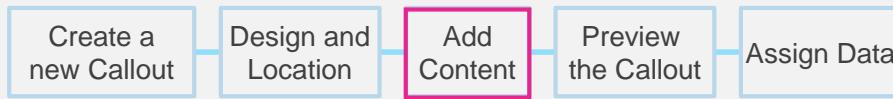
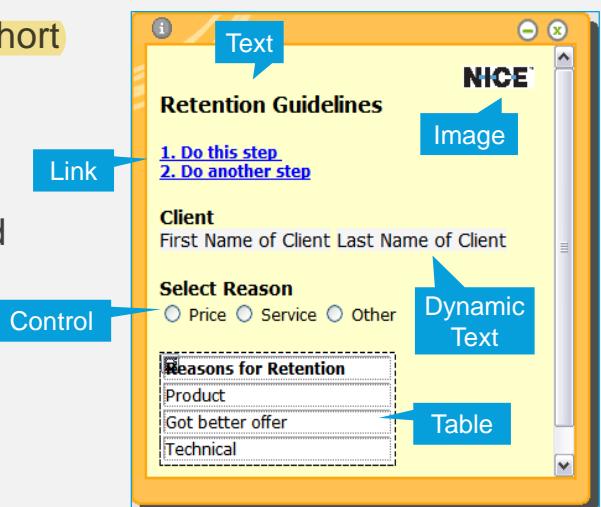
NICE®

Add Content to a Callout



Callouts Content

- The content of a Callout should be short and clear so that it can be read and understood quickly by an Agent.
- The following objects can be entered in a Callout:



NICE®

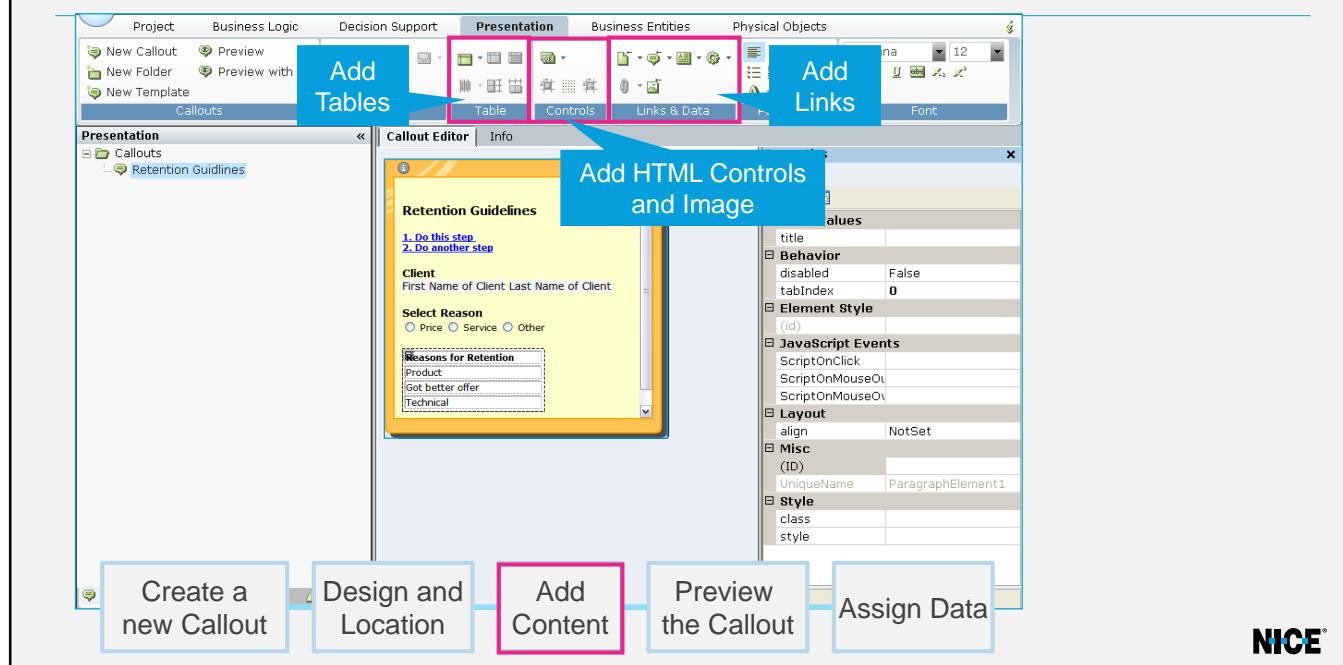
Link – RT Designer enables implementer to create various kinds of links that can be displayed by clicking on text, an image or a button in a Callout:

- Link to another Callout
- Link to a Launch-In-Context item
- Link to invoke a Function

Control – various HTML GUI elements that can be inserted into a Callout, such as a button, text field and dropdown menu.

Note: Adding an object to a Callout's body is done through the right-click menu, or Tool menu in Presentation layer.

How to Add Content to the Callout



All Controls can be designed using the Style Editor from the Editing Menu.

When adding links, you will be asked to select the object you want the control to be linked to.

For example, adding a link to another callout will require you to drill down to the other Callout.

HTML Controls

- RT Designer enables you to create various GUI objects that can be inserted into a Callout



Check box



Input Text area

Radio
Button

Button



Combo Box

Date
Picker

IFrame

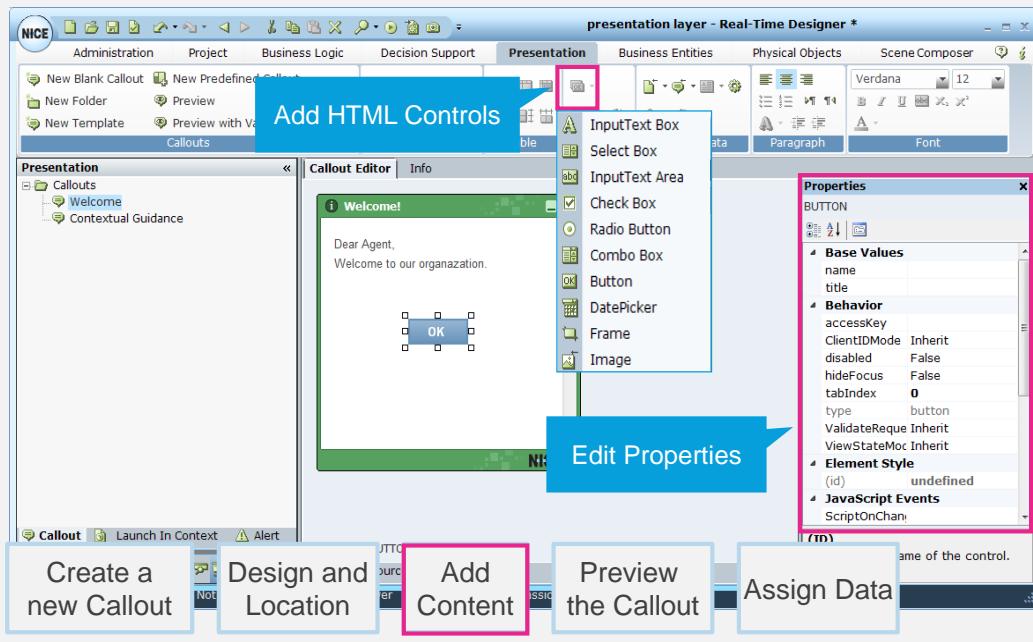
Create a
new CalloutDesign and
LocationAdd
ContentPreview
the Callout

Assign Data

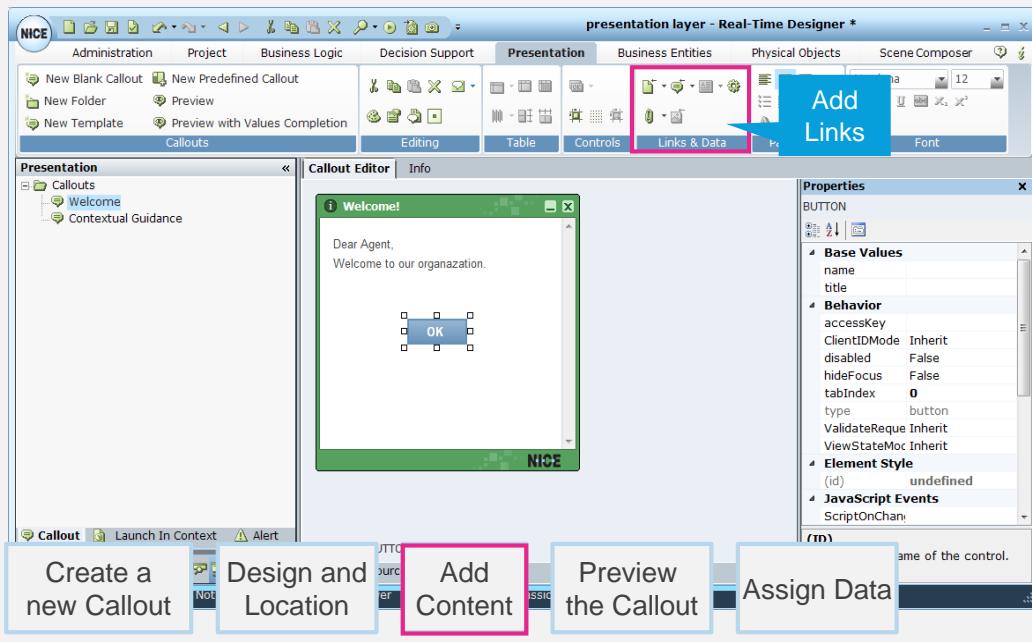
NICE

- GUI Controls are simply HTML Objects.
- These objects are built-in HTML elements.
- It is possible to view and modify an object's HTML source from the Source tab.

Callout Content – HTML Controls



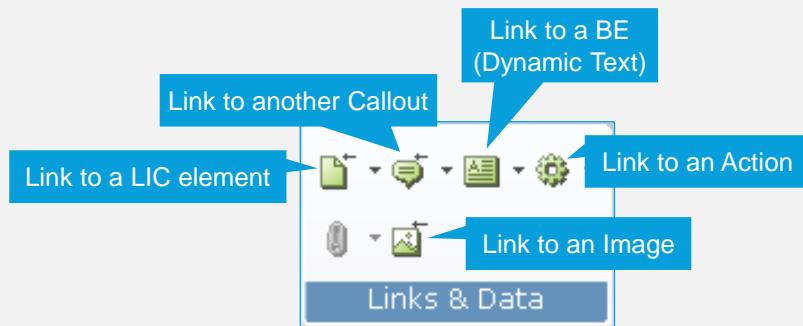
Callout Content – Links



- RT Designer enables you to create various kinds of links that can be displayed by clicking on text, an image or a button in a Callout such as a Link to another Callout, to a Launch-In-Context item or a Link to invoke a Function.
- When you add links, you will be asked to select the object you want the control to be linked to.
- For example, adding a link to another callout will require you to drill down to the Callout you wish to link to.

Links Menu

- Various Links are available to be added to a Callout:

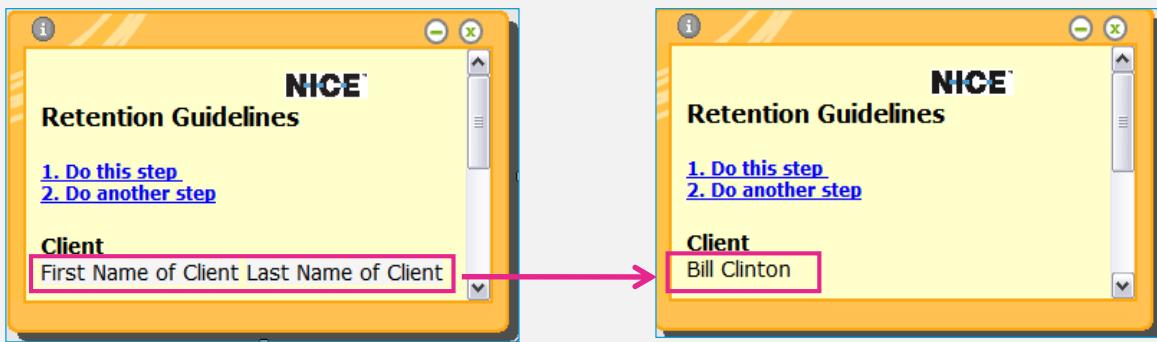


NICE®

- When you add links, you will be asked to select the object you want this control to be linked to.
For example, adding a link to another callout will require you to drill down to the other Callout.

Link to a BE

- **Dynamic Text** control enables you to insert text in a Callout that reflects the value of a BE's Property
- That value may change to reflect the new value of that Property each time it changes



NICE®

- In the Callout's design assign dynamic text to BE properties.
- In real time the Callout will reflect that property's value.

Link to an Action

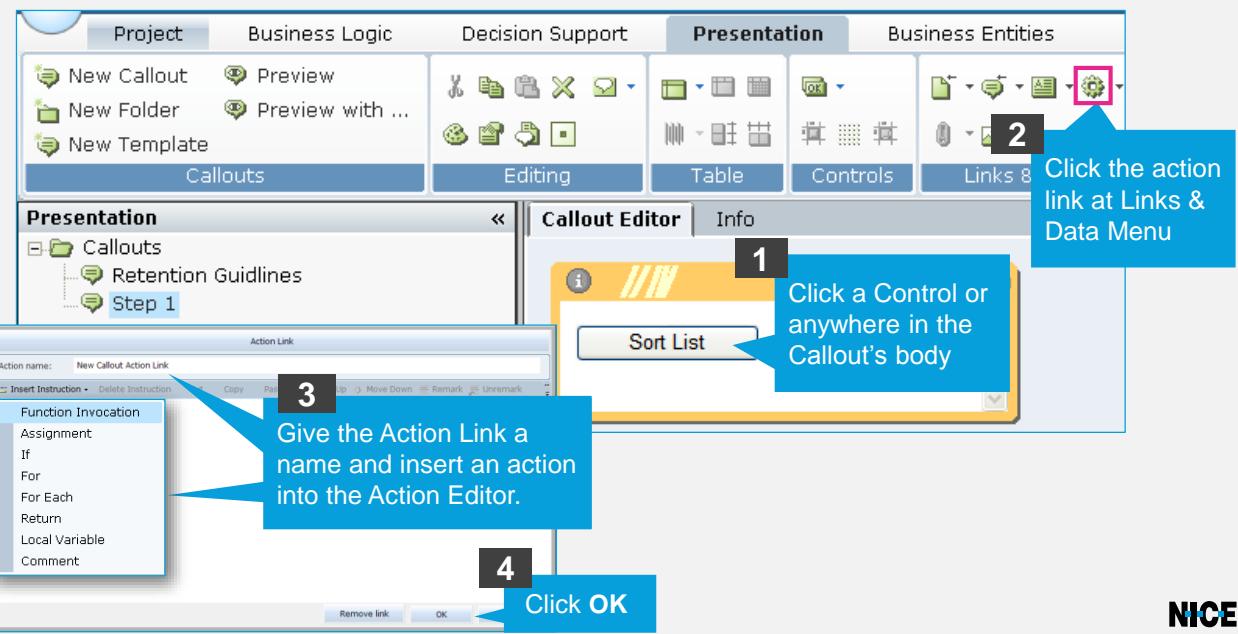
- **Functions** perform an action, and can return a value

				
SE Functions	BE Functions	Library Objects Functions	Callout Functions	LIC Functions
Get Text	Add Item to List	Add Numbers	Show / Close	Show / Close
Click	User created functions	Get Active Process	Set Body	

NICE®

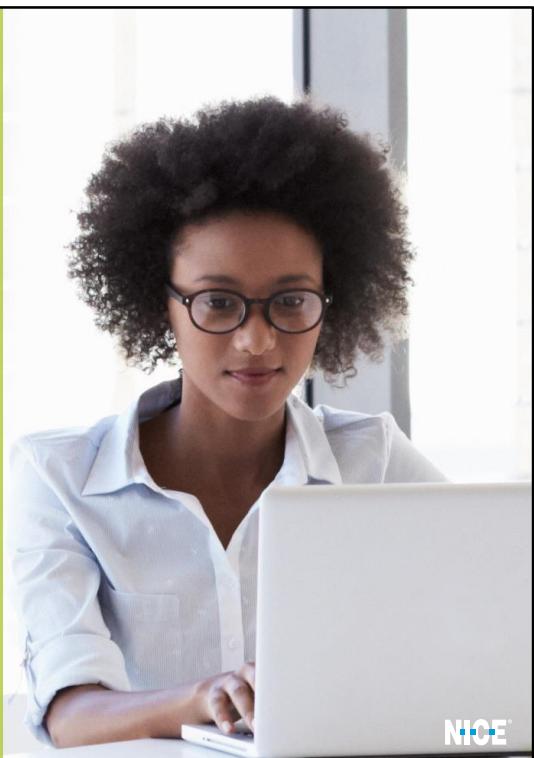
- **Business Entities Functions** – Can be either a function the Integration Engineer created or a built-in function like sorting a list.
- **Library Objects Functions** – A built-in functions library, much like Excel. A drill-down menu will display the group functions (Math functions / Date and time functions etc.), and then displays the functions you can use in this group (Sum, Current date etc.).

How to Link an Action to a Control

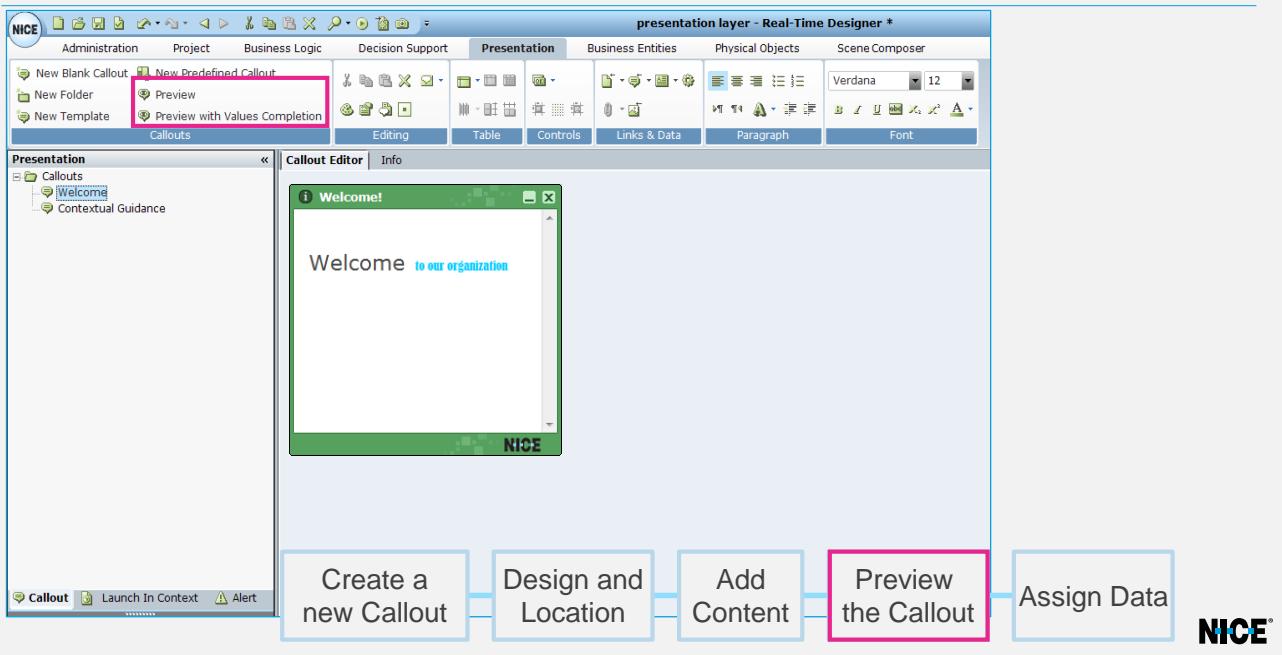


- To add an Action as a hyperlink, perform steps 2 and 3. The Action's name will be added as a hyperlink to the Callout.

Preview the
Callout



Callout Content - Text



Typically, when designing a callout, the callout is previewed during the design process to view how it will appear at run time

There are two ways to Preview a Callout:

1. **Preview** displays the callout AS IS (with no values assigned to objects), mostly used for checking appearance and location on the screen.
2. **Preview with...** enables you to preview a callout with dynamic data

Quick Callouts- Callouts On Demand

- The RT Client provides Agents with a window of options from which they can quickly access the most frequently used functions or information that they require



- A Quick Callout window is displayed when an Agent clicks the Quick button in the Agent's Launch Bar

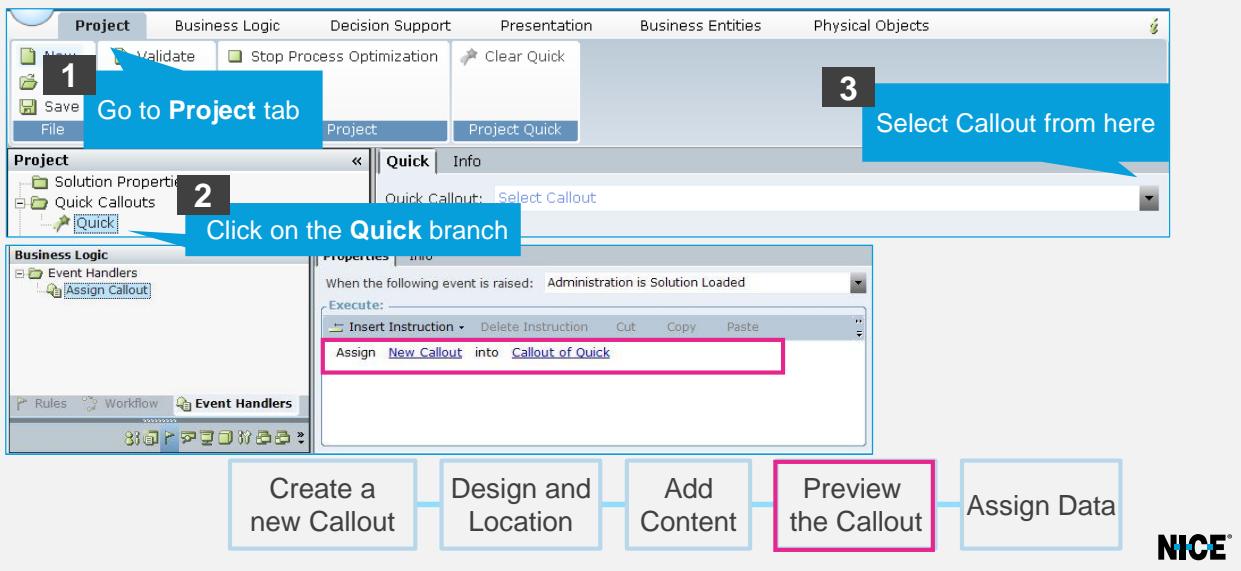


NICE®

- Only one Quick Callout window can be displayed on an Agent's screen, so create a Callout with all the Functions and information the Agent will need to reference quickly, and add that Callout to the Quick Menu.

How to add a Quick Callout

- Add a Callout to the Quick Menu

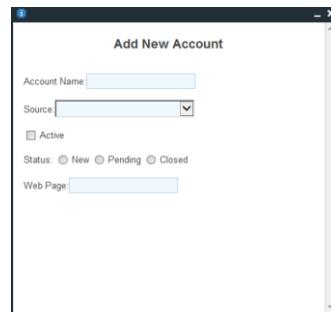
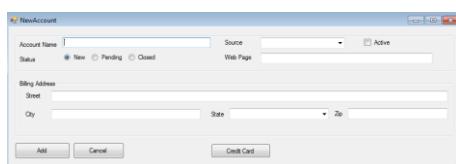


- The Quick Callout is selected from the project tab, and only one callout can be selected as the Quick Callout. So, to create a Quick Callout, in the Project tab click on the Quick branch, and then select a specific callout you have created.
- While only one callout can be selected as the Quick Callout in the Project tab, you can define Business Logic that changes the callout to be shown as a Quick Callout window, using a business rule, workflow or event handler.
- After the definitions are completed, run the RT Client to view it from the Quick or Help buttons in the Launch bar.
- Assignments will be displayed with real time values.

LIVE DEMO

Adding Content to a Callout

- Add the fields from the new account window to the callout:



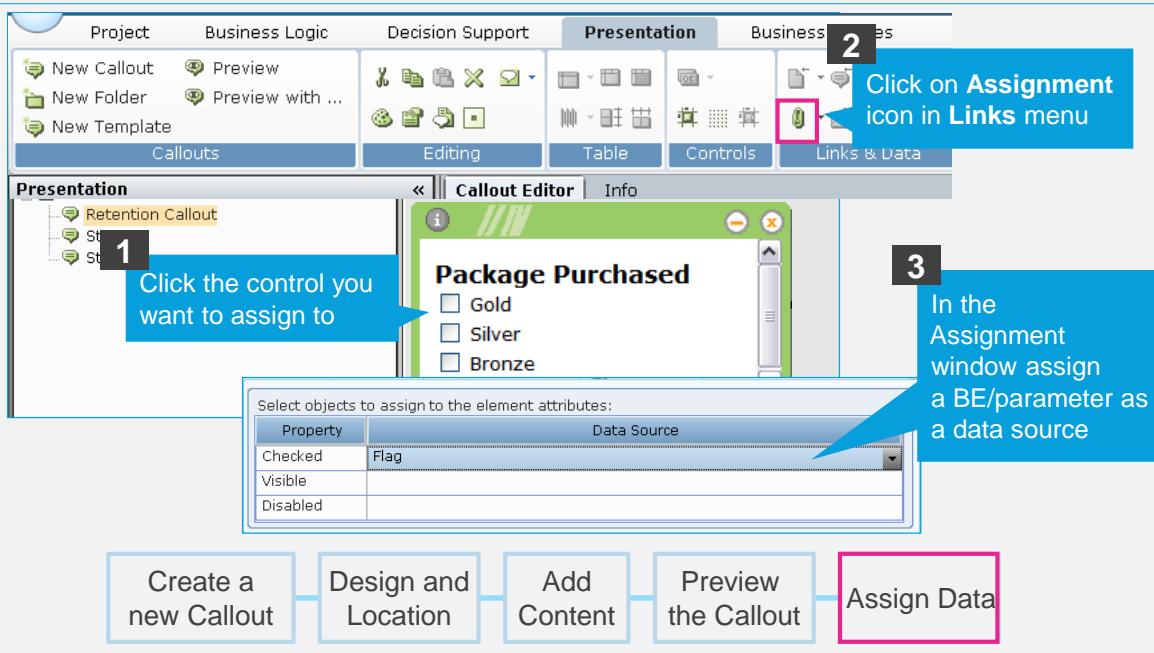
- Set this callout to be a Quick Callout

NICE®

Assign Data to a Callout

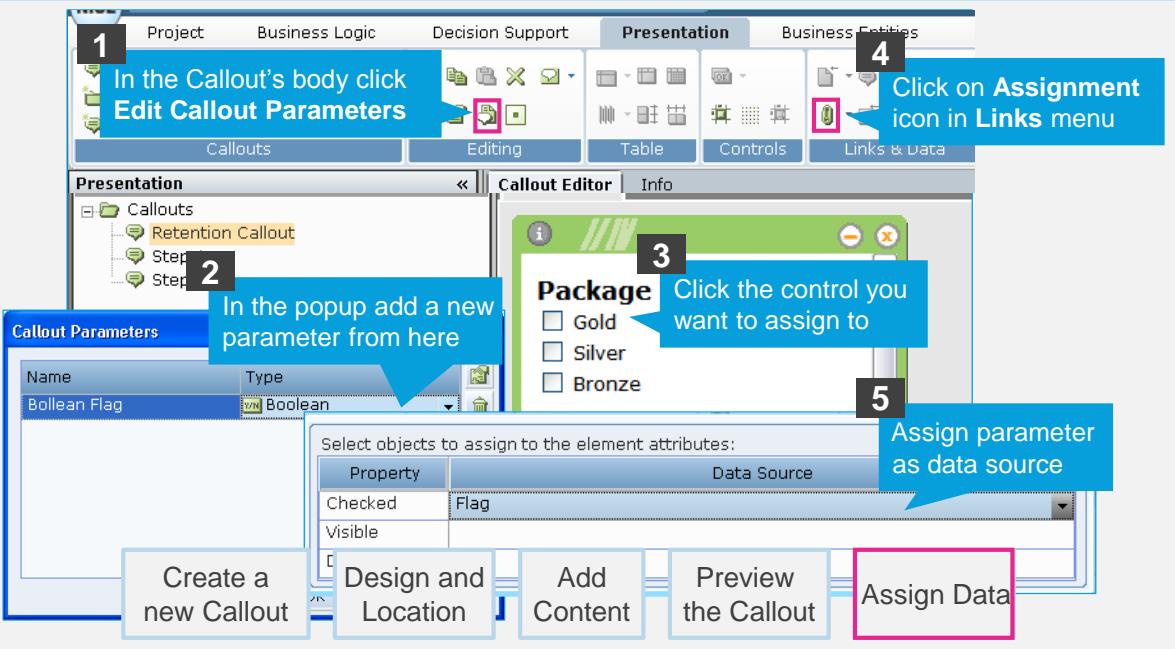


How to Assign a Control to a BE



- FLAG's value will be determined according to the BE's value in real time.

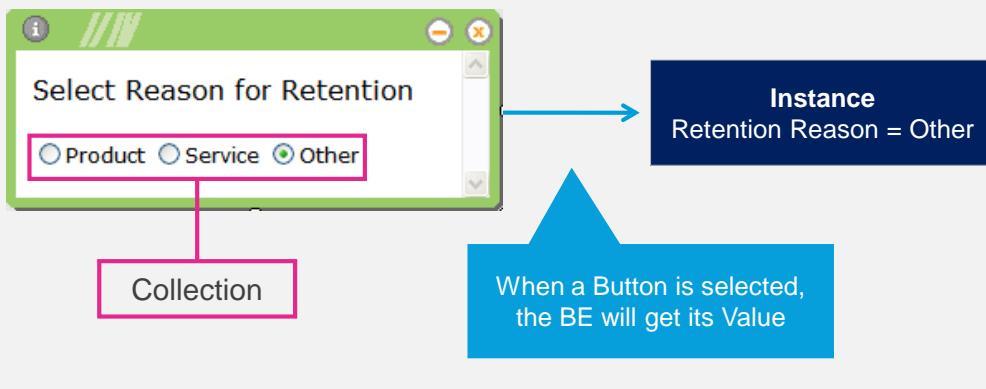
How to Assign a Control to a Parameter



- The element that will trigger the Callout to present, will have to deliver a value for the FLAG.

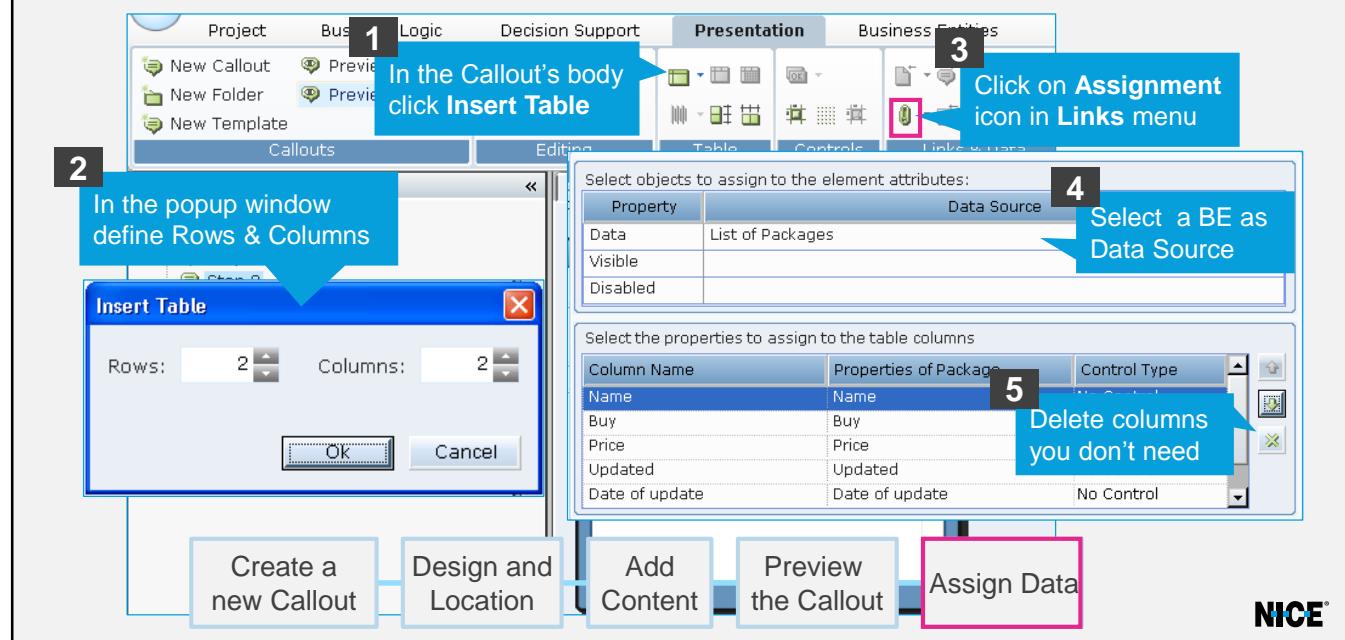
How to Use Radio Buttons

- Radio Buttons have to be a part of a collection, since they are interdependent
- If we assign one of the Radio Buttons to a BE, its value will be assigned to the BE and vice versa



- The Collection is defined in the Radio Buttons' properties as Name.
- When several Radio Buttons are given **the same name**, they are in a collection.

How to Assign a Table to a BE



- In order to display a table in the Callout that contains information from a BE, all you have to do is create the table as described in this slide.
- There is no need to add the columns' names in the callout itself, RT Designer arranges the table so that the properties' names are automatically displayed as column names.

LIVE DEMO

Assigning Data to a Callout

- In your BE Layer, create the required Business Entities to assign to the Controls in the callout
- Assign your Business Entities to the controls in the “New Account” Callout
- Test your solution

NICE®

Summary

- Launching documents in context of an application
- Describing the use of Callouts
- Creating, designing and previewing a Callout
- Adding content to a Callout

NICE®

- Presentation Layer defines the items that are displayed on an agent's/supervisor's screen
- **Launch in Context** enables you to display a document in the context of its standard application
- A **Callout** is a popup window that is displayed on an Agent's screen under certain conditions
- A Callout can contain predefined text, HTML controls, images and dynamic data
- After creating a Callout, define its location and design, add Controls and links and preview your Callout



Thank You





PHYSICAL OBJECTS LAYER: DATABASES



Lesson Objectives

By the end of this lesson you will be able to:

- Add a Database connection
- Create Database elements
- Stream data from Database elements to Business Entities and vice versa



Physical Objects Layer

- This is the first layer on which all other layers are based.
It represents real physical entities in the environment in which the Real-Time solution will operate.

Business Logic

Decision Support

Presentation

Business Entities

Physical Objects



Screen elements



DB elements



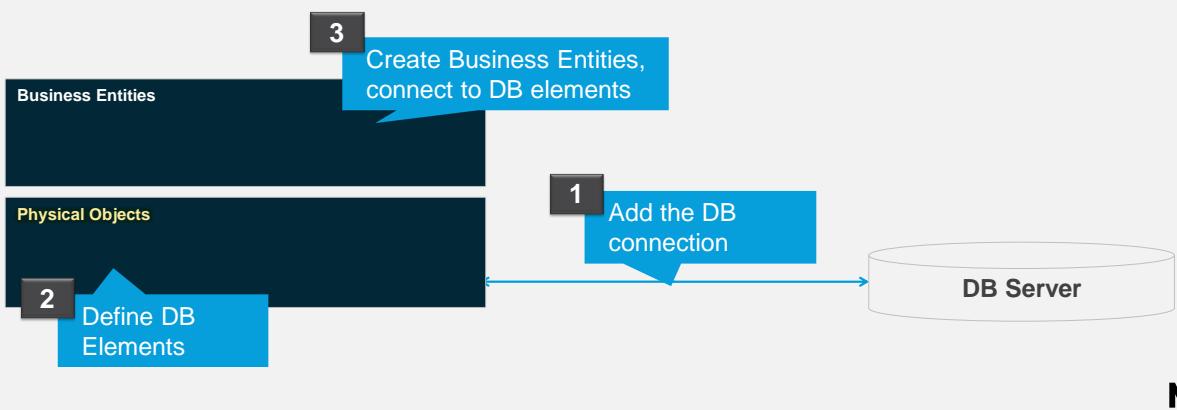
Web Services

Each Physical Object is a connection to a source of information

- DB Elements and Web Services are part of the Physical Objects Layer.

DB Connections Layers Module

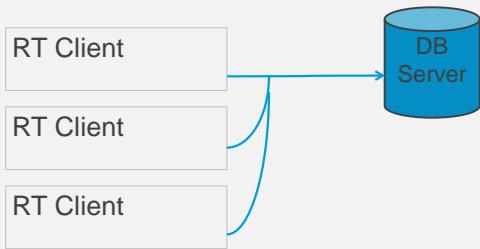
- DB connections are part of the Physical Objects layer
- They are used in order to allow data flow from the database to data structures called Business Entities, and vice versa



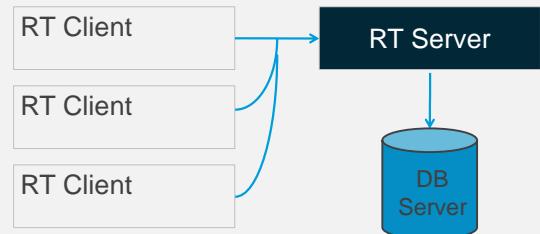
- When you begin to design a solution, you must get a list of what data is required from the database, from the business analyst
- Then, you can follow the step-by-step flow described here.

DB Connections

- Before you can define a database element, you must first create a connection to the external database
- Local connection



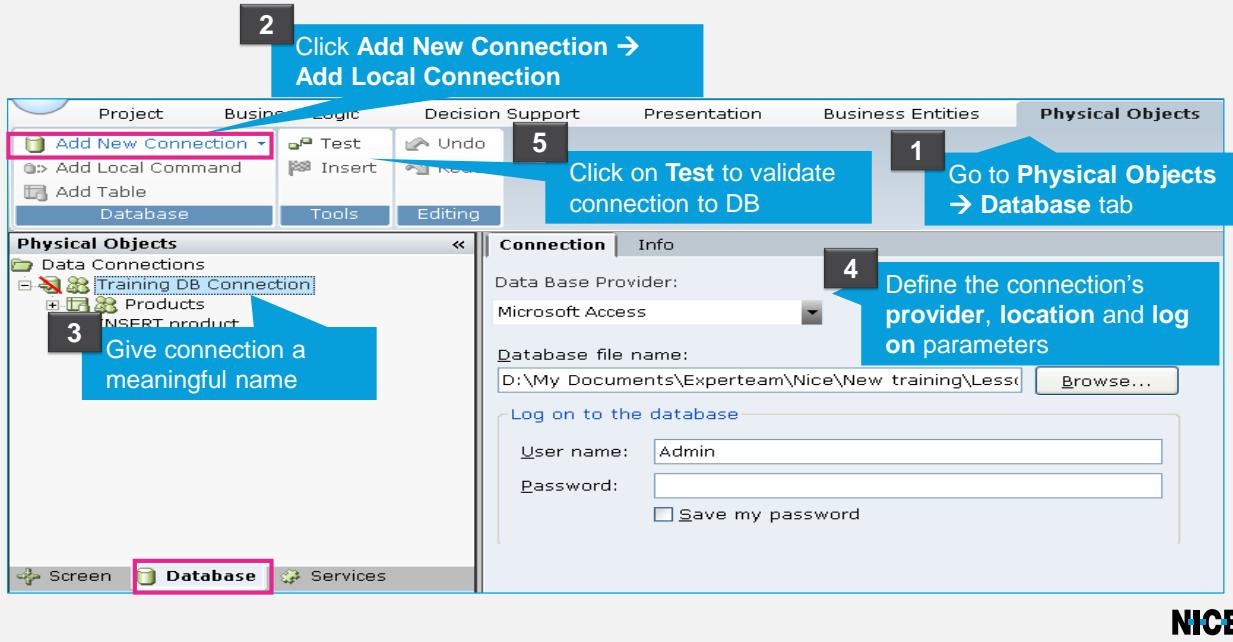
Server connection



NICE®

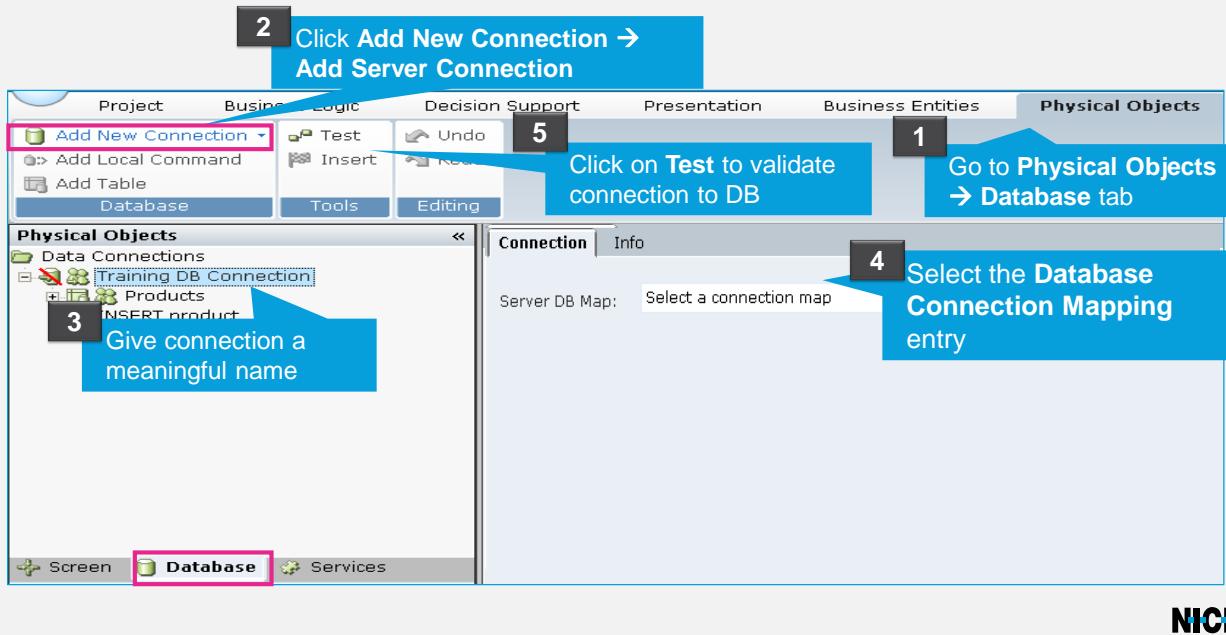
- **Local connection** - each Agent has their own direct connection to the database.
- **Server connection** - a single connection is made to the database, which is used by all Agents to obtain data from the connected database. This type of connection is advantageous when a large number of Agents need to be connected to the database at the same time. Because there is only one connection to the database, the maximum number of network connections allowed is not exceeded.

How to Add a Local DB Connection



- **Supported providers**
- Microsoft Access
- Microsoft SQL
- ODBC
- OleDB
- Oracle
- **Provider Log on parameters** are standard connection parameters to each database and differ for each database type. This slide displays an example for Microsoft Access parameters.
- The Test step validates connection to the database. If no connection was established between RT Designer and the database, check for errors that appear in the Messages pane.

How to Add a Server DB Connection



- **DB Connection map** – Defined by the System Administrator, to assign a correct DB Server URL for each environment – development, test and production.



Notes from Demo

DO IT YOURSELF

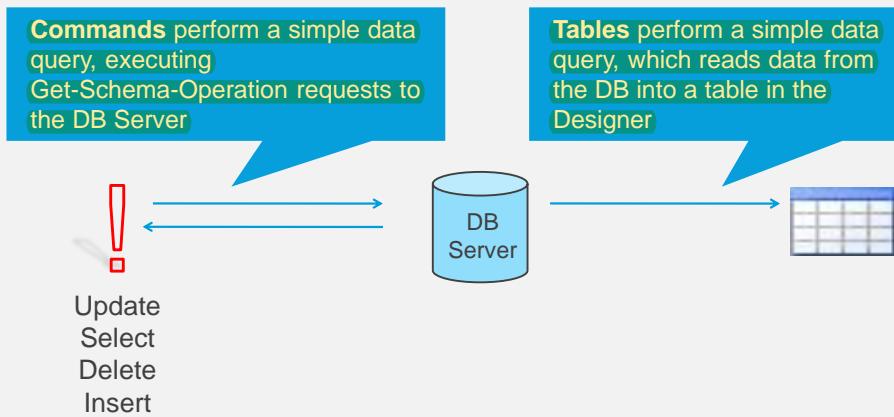
Adding a DB Connection

- Add a new local DB Connection called 'My DB'
- Define the Database Provider to be Microsoft Access
- Enter the path of the MS Access 'Database' file provided
- Leave the default Log On parameters
- Test the connection to validate it

NICE®

DB Connector Elements

- Two options are available for creating database connector elements:
Commands and Tables



NICE®

- DB elements will be used in upper layers. Therefore, you should create all commands that involve connection to the database, and all tables you will need information from

How to Load a Table from the DB

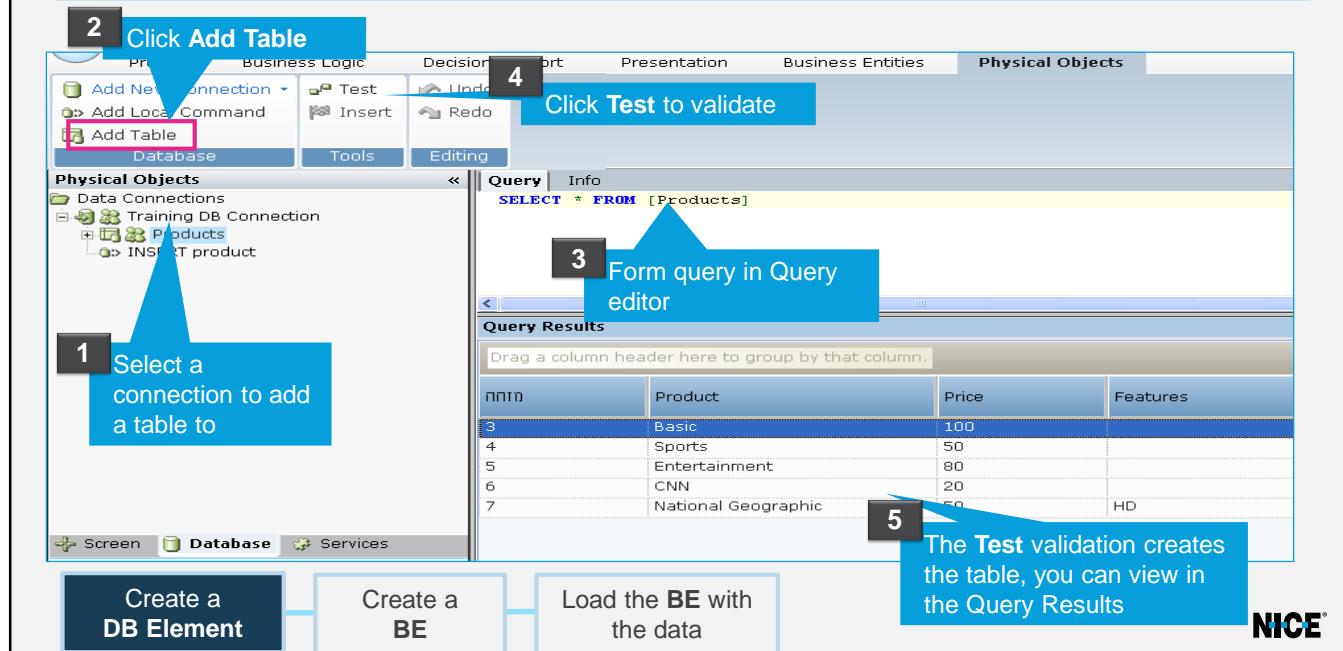
1. A Table DB Element is created at the Physical Objects Layer
2. A BE is created at the Business Entities Layer, in order to hold the Table's data
3. A Load Function is added to the BE in order to activate the DB Element



NICE

- Functions are defined at the Type tab, but operate on the Type's Instances
- Built-in Function on DB tables:
 - **Update** Function updates a table in the DB
- Built-in Functions on DB commands:
 - **Execute Non Query Function** executes a DB command against the DB connection. and returns number of affected rows. For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command. For all other types of statements, the return value is -1
 - **Execute Scalar** Function executes a DB command against the DB connection. It returns the first column of first row in the result set

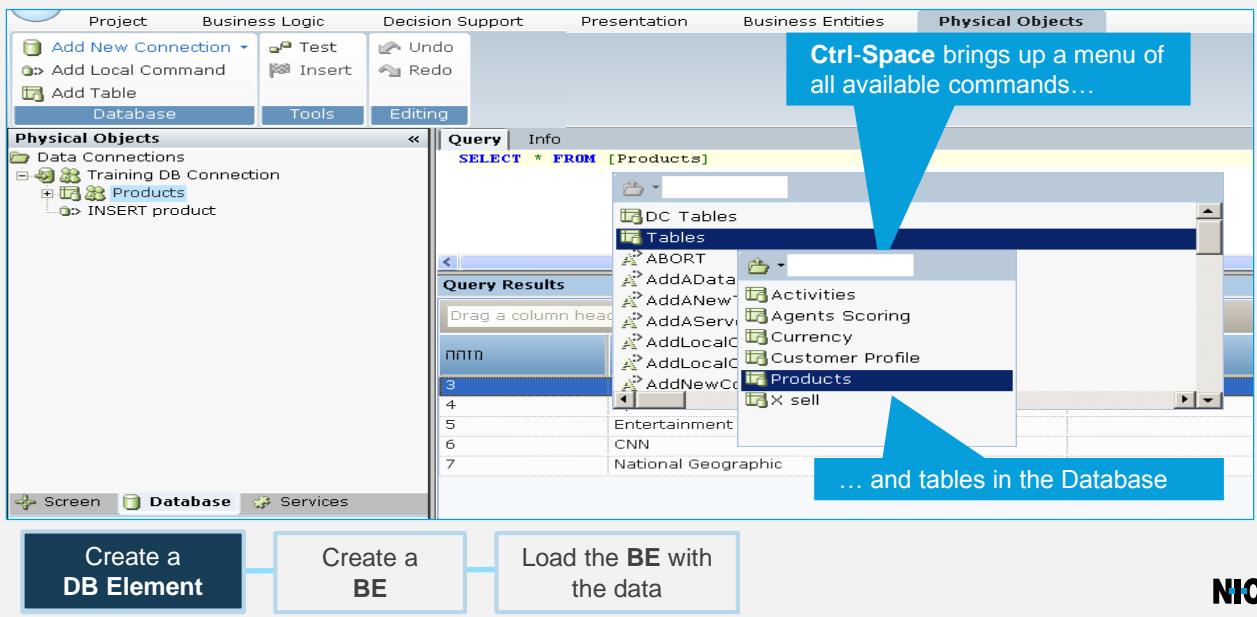
How to Create a DB Table Element



- Tables are only enabled in a Local DB connection.
- Defining a Table in RT Designer enables you to define a SELECT SQL command, which reads information from a database into a table in RT Designer.
- The following command enables you to access all data in the table: **SELECT * FROM**

Note: Query Editor may enable you to type another first word apart from SELECT, but it will not be valid.

How to Create a DB Table Element



- Query editor automatically completes the SQL syntax for you, all you need to do is click it.
- There are two ways to write a command in the Query Editor:
 1. Click **Insert** at Tools menu to open a list of all the options of a standard SQL editor to select from.
 2. Start typing the command. The RT Designer uses Intellisense, which automatically completes the command with every letter you type.
- **NOTE:** The command's parameters can be of a fixed value, or BE used as parameters. For example, Select **ColumnName** FROM **tableName**
- **Ctrl + spacebar** opens a list of commands, BEs and tables for you to select from.
- Adding a dot after a table will display a list of the table's fields to select from.



Notes from Demo

DO IT YOURSELF

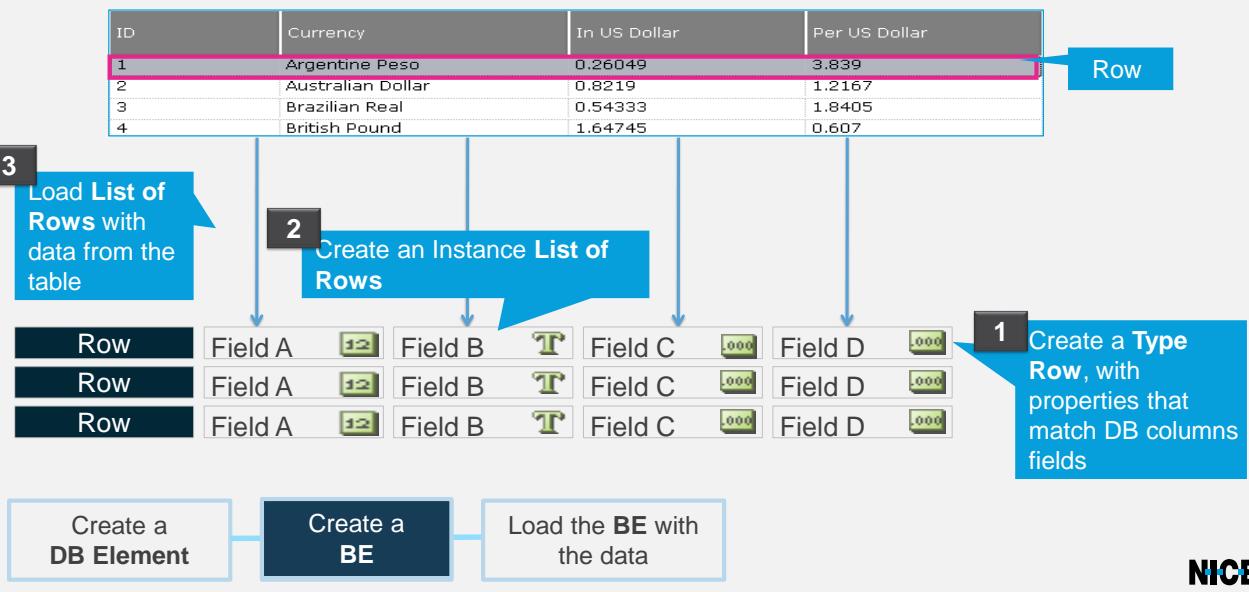
Creating a DB Table Element

- Add a 'My Products' table element to your 'My DB' connection
- Enter the query to select all columns from the 'Products' table in the MS Access 'Database' to be part of your table element
- Validate the query by testing it

NICE®

How to Load Tables into BEs

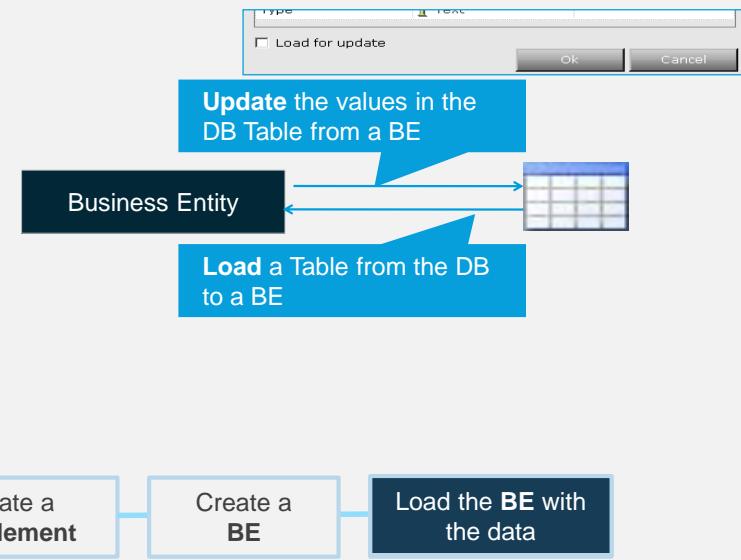
- A table in the database is actually a list of rows



- All DB tables that are used in the solution must be loaded to BEs. In order for the BE to hold the table's data, it must be structured as a list of rows. Each row will hold the columns Types.
- Follow these steps:
- Create a User Type “Row”. Add “Row” properties according to the table's columns Types.
- Create an instance “List of Rows”.

DB Table's Functions

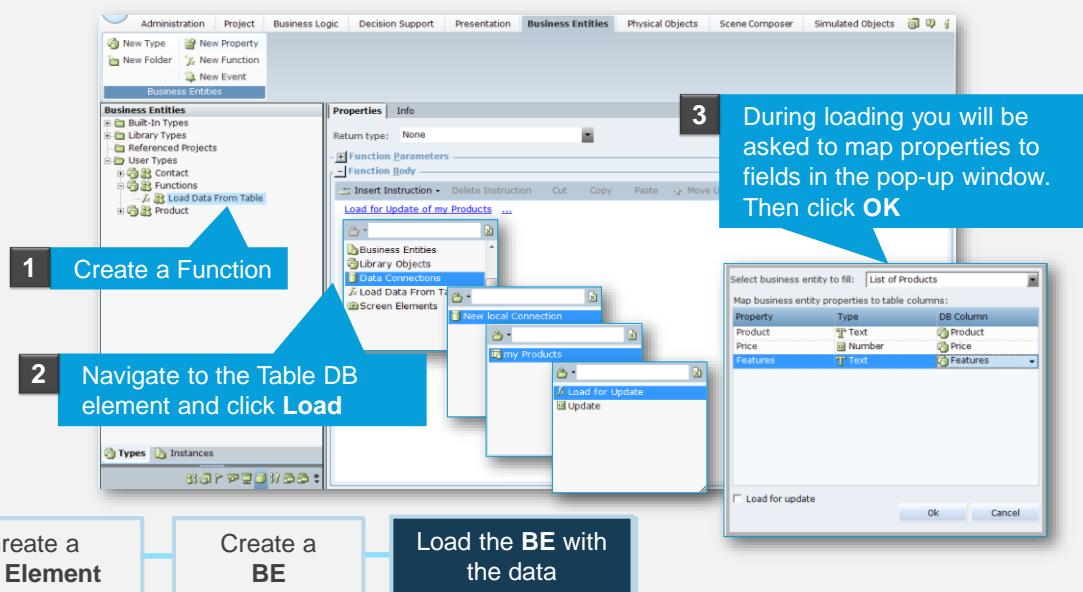
- There are 2 built-in Functions available for Tables:



NICE®

- When loading, there will be a **Load for update** checkbox. It should be checked only if you intend to update the table in the DB during the solution. Until now, you have loaded data from DB field to property. If **Load for update** is checked, the reverse option is enabled: the DB field will be open for update from the property's data if it was changed.

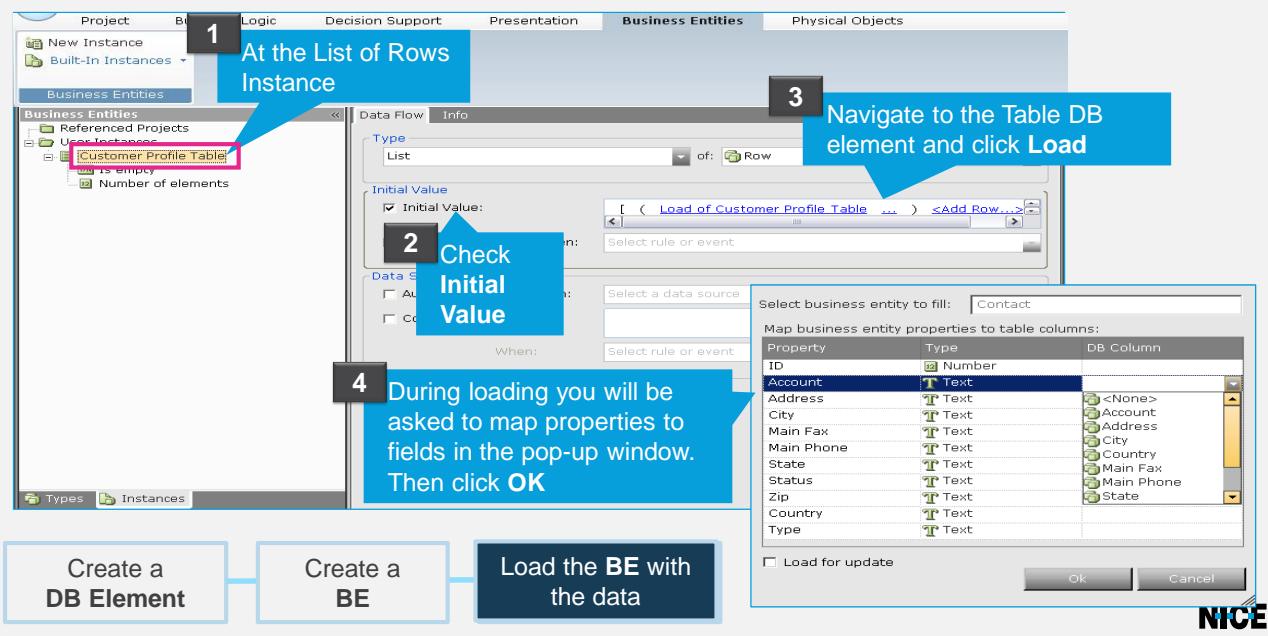
How to Load Data From the Table into the BE



NICE®

- Only DB columns from the same type as the property mapped will be available. For example, a property from the type Number cannot be mapped to a Text field in the DB

How to Load Data From the Table into the BE- Initial Value





Notes from Demo

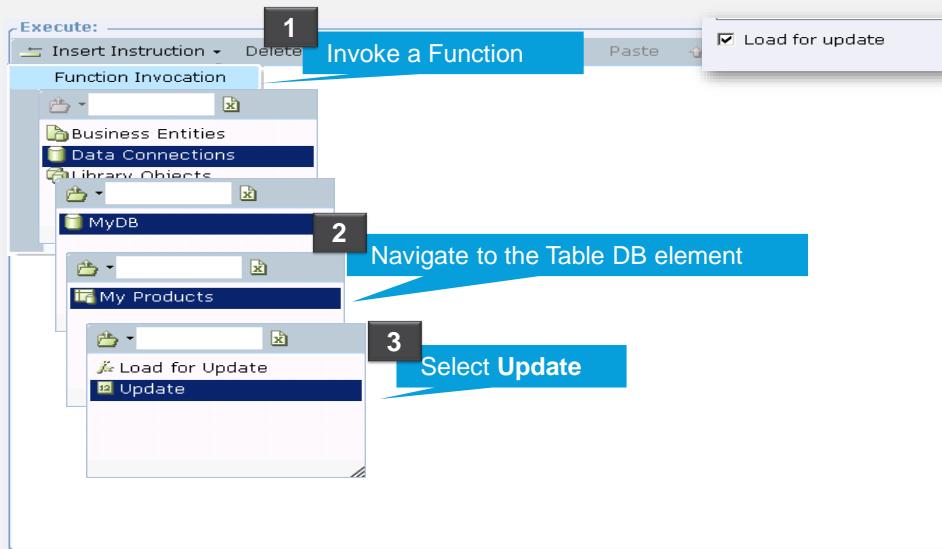
DO IT YOURSELF

Loading a Table into a Business Entity

- Create a Business Entities project
- Create a Type called 'Product Row'
- Add properties to it that match the names and types of the DB elements:
 - ID
 - Product
 - Price
 - Features
- Create a BE Table by creating a List of Product Rows
- Initialize the List by Loading the 'My Products' table
- Map the fields in the 'My Products' table entity to the properties of the Product Row

NICE®

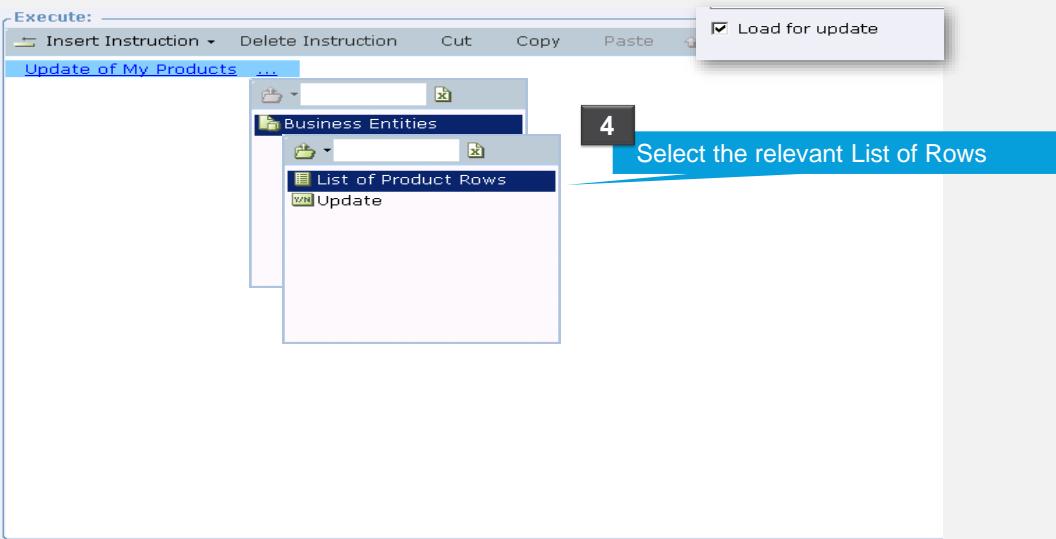
How to Load Data from the BE into the Table



NICE®

- The table will only be successfully updated by Lists that have already been mapped to it

How to Load Data from the BE into the Table

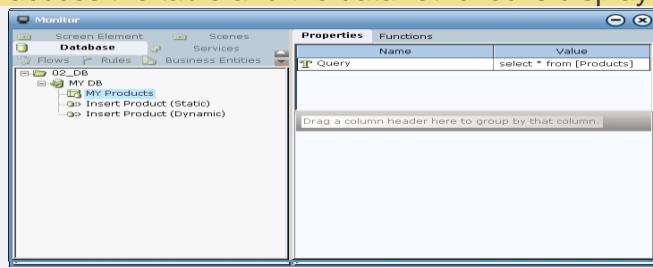


NICE®

- The table will only be successfully updated by Lists that have already been mapped to it

How to Monitor Databases and their Connections

- The Database tab presents the state of the database connections and their Data Tables
- Database Connection:
 - Connection String shows the value of the connection string used to establish the Database connection
 - Recent Activation time shows the date and time the database was recently accessed by the Client
- Data Table:
 - The query used to access the table and the data retrieved is displayed



NICE®

- The table will only be successfully updated by Lists that have already been mapped to it



Notes from Demo

DO IT YOURSELF

Loading a Business Entity into a Table

- Create a Type (and Instance): ‘DB Commands’:
 - Function: Updates the ‘MyProducts’ table with the values in the ‘List of Product Rows’

NICE®

Static DB commands

- DB Commands perform a simple data query, executing Get-Schema-Operation requests to the DB Server
- Commands' parameters can use Static or Dynamic values
- `SELECT [Customer Profile].Age FROM Customer Profile`

SQL command

DB Table

DB Column



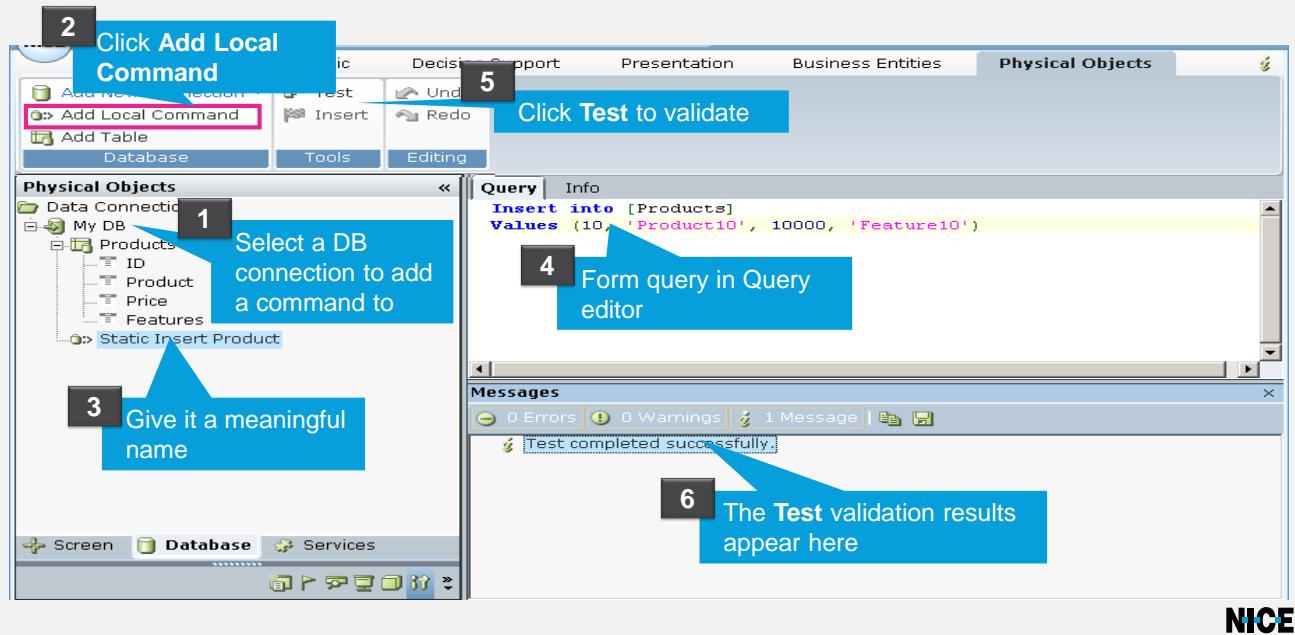
DB commands

- Common DB Commands
- **SELECT** [Column(s)] FROM [TABLE]
WHERE COLUMN1 [operator] Value
- **INSERT INTO** [TABLE]
VALUES (Col1Value, Col2Value, ...)
OR
- **INSERT INTO** [TABLE] (COLUMN3, COLUMN4...)
VALUES (Col3Value, Col4Value, ...)
- **UPDATE** [TABLE]
SET COLUMN2=NewValue, COLUMN3=NewValue...
WHERE COLUMN1 [operator] Value
- **DELETE** FROM [TABLE]
WHERE COLUMN1 [operator] Value

NICE®

- Commands examples:
- SELECT command extracts data from the database
- INSERT command inserts data into the database
- UPDATE command updates existing data in the database
- DELETE command deletes data from the database
- Built-in Functions on DB commands:
 - **Execute Non Query** Function executes a DB command against the DB connection. and returns the number of affected rows. For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command. For all other types of statements, the return value is -1
 - **Execute Scalar** Function executes a DB command against the DB connection. It returns the first column of first row in the result set

How to Add a DB Command Element



- The values entered into an INSERT command must be in the same order as the fields in the DB table appear. In this example, the order is ID, Product, Price, and then Features



Notes from Demo

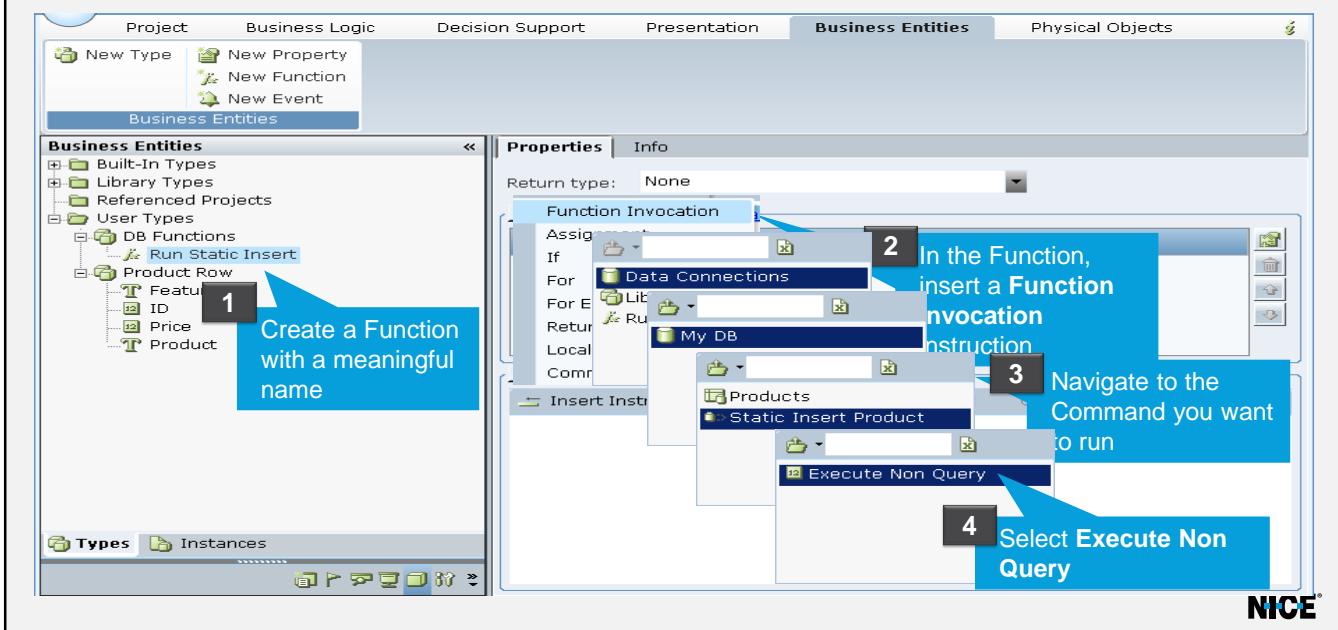
DO IT YOURSELF

Static DB Commands

- Create a DB Command Element called 'Insert Product (Static)'
- In the Query Editor of the Command, write a query that inserts a new entry in the Products table in the DB, specifying the following values for the following fields:
 - Product = 'Microwave'
 - Price = 275
- Validate your Command by Testing it

NICE®

How to Run a DB Command



- The values entered into an INSERT command must be in the same order as the fields in the DB table appear.
- In this example, the order is ID, Product, Price, and then Features



Notes from Demo

DO IT YOURSELF

Running DB Commands

- Add elements to the 'DB Commands' Type:
 - Function: Run the 'Insert Product (Static)' DB command
 - Function: Loads the values from the 'MyProducts' table into the 'List of Product Rows'

NICE®

Dynamic DB commands

- DB Commands perform a simple data query, executing Get-Schema-Operation requests to the DB Server
- Commands' parameters can use Static or Dynamic values

SELECT [Customer Profile].Age FROM Customer Profile

SQL command

DB Table

DB Column

SELECT [table].column FROM table

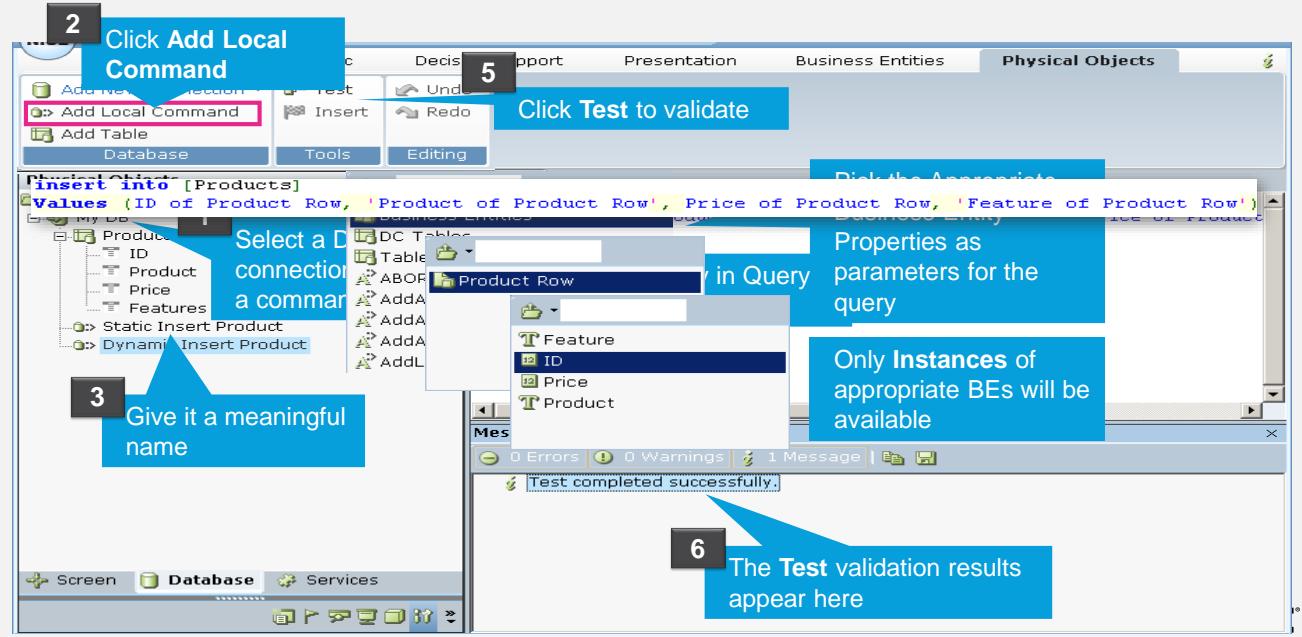
BE Property

BE Property

NICE®

- Commands examples:
- SELECT command extracts data from the database
- UPDATE command updates data in the database
- DELETE command deletes data in the database
- INSERT command inserts data to the database
- Built-in Functions on DB commands:
 - **Execute Non Query** Function executes a DB command against the DB connection. and returns the number of affected rows. For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command. For all other types of statements, the return value is -1
 - **Execute Scalar** Function executes a DB command against the DB connection. It returns the value in the first column of first row in the result set

How to Add a Dynamic DB Command Element



- The values entered into an INSERT command must be in the same order as the fields in the DB table appear. In this example, the order is ID, Product, Price, and then Features



Notes from Demo

Layered Projects with Databases

- In general, only higher layers should reference the lower layers. The reverse is not encouraged.
- Creating Dynamic Queries also requires the Physical Objects Layer to receive information from the BE Layer.



NICE®

Layered Projects with Databases

- In general, only higher layers should reference the lower layers. The reverse is not encouraged.
- Creating Dynamic Queries also requires the Physical Objects Layer to receive information from the BE Layer
- So how do we accommodate for this?
 - Create a dedicated project for DBs, including all related BEs
 - Reference this project from the regular BE project

Business Entities Project

Execute Non Query of Command

DB Project

Physical Objects:

- DB Table
- Field1
- Field2
- Field3

Command

Business Entities:

- Row
- Field1
- Field2
- Field3

Physical Objects Project

Screen Elements

NICE®



Notes from Demo

DO IT YOURSELF

Dynamic DB Commands

- Create an Instance of Product Row
- Create a DB Command Element called 'Insert Product (Dynamic)'
- In the Query Editor of the Command, write a query that inserts a new entry in the Products table in the DB, specifying the properties of the Product Row type as the values for their corresponding fields.
- Validate your Command by Testing it
- Create a Function that executes the 'Insert Product (Dynamic)' Non-Query

NICE

Summary

- Adding a Database connection
- Creating Database elements
- Streaming data from Database elements to Business Entities and vice versa

NICE®

- DB connections are part of the Physical Objects layer
- They are used in order to allow data flow from the database to data structures called Business Entities
- Step by step design:
- Add a DB connection to RT Designer
- Create DB connector elements: commands and tables
- Create Business Entities to hold data and invoke DB elements



Thank You





BUSINESS LOGIC LAYER: WORKFLOWS



Lesson Objectives

By the end of this lesson you will be able to:

- Create Workflows – Steps and Transitions
- Test Workflows



Business Logic Layer

- The Business Logic Layer enables you to define the logical entities that determine the behavior of RT Client according to your business requirements

Business Logic



Rules



Event Handlers



Workflows

Decision Support



Presentation



Business Entities



Physical Objects



- Workflows provide a structured way to define a business process by Steps and Transitions.

What are Workflows?

- In life, processes are typically comprised of a series of functions to be performed in a specific order and dependent upon specific conditions

Steps define the Actions to be done

Transitions define conditions to move on to the next Step

- Workflows are used to:
 - Guide Agents through a process
 - Force Agents to perform a flow in a certain order



NICE®

- Workflows are a series of steps performed in a certain order, and transitions that define when to move between steps.
- There is a certain order to the Workflow's steps – You do not brush your teeth before getting up from bed.
- Transitions are like this flow: you get out of bed >you brush your teeth >you get dressed > you eat breakfast> you arrive at work.

When are Workflows Triggered?

- Workflows can be triggered in 3 ways:

Start

- On demand

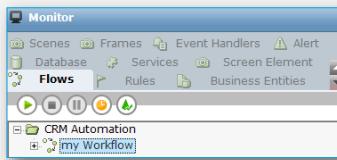
Users can click the Flows button on the Launch Bar. The Launch bar appears on the user's desktop to display a list of available workflows



- Push Mode

Triggered by a Business Entity or a Rule

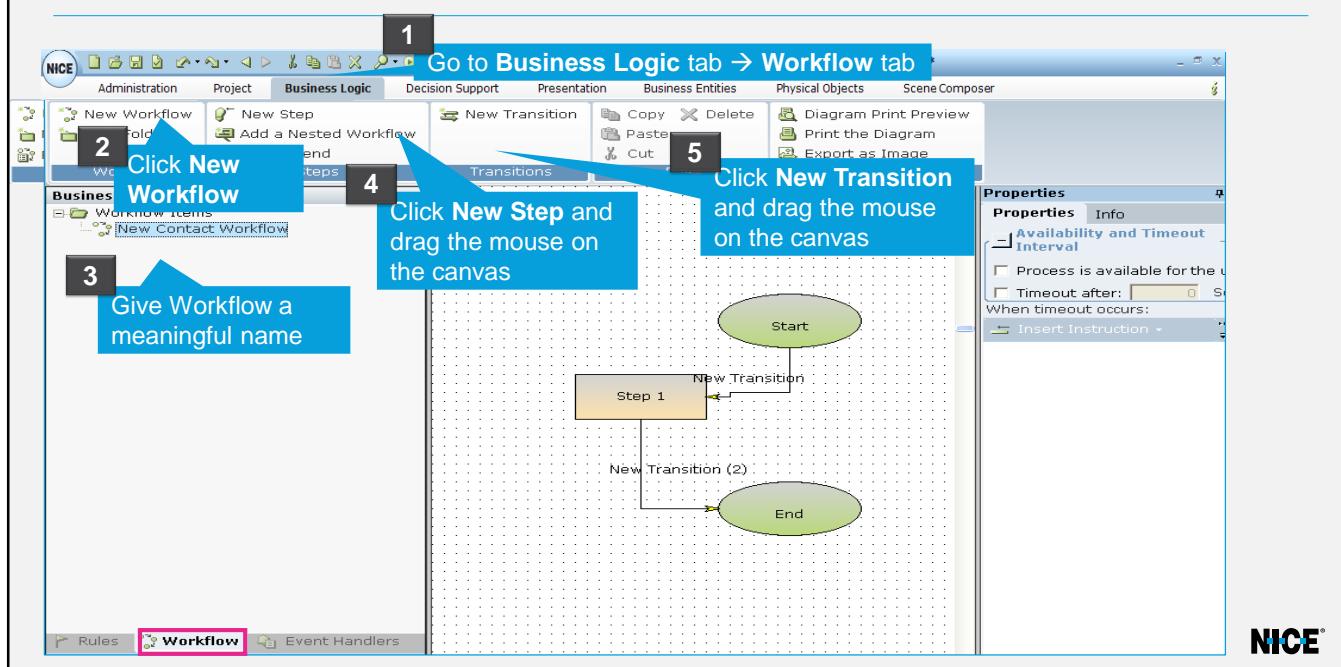
- Through the Monitor



NICE®

- Push Mode examples:
- Convert a lead to a new Account
- Create a service request
- Navigate to the billing info screen
- Get and rank available offers

How to Create a Workflow



- Transition condition - When a certain Event or a Rule occurs, transition to the next step.
- If you do define 2 transitions to a single step, you must define their priorities. If both conditions are met, the one with the higher priority will be executed, and the other will be discarded.

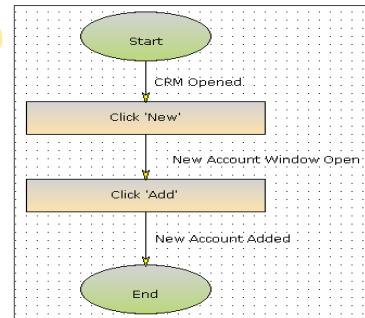


Notes from Demo

DO IT YOURSELF

Creating a Workflow

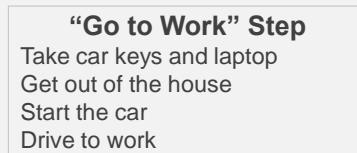
- Create a workflow called 'Guide', which will guide an agent through the process of creating a new account, and set it up as follows:
 - Create the necessary layers to accommodate this task (capture SEs, BE, callouts....)
 - Create a BE Boolean instance which is automatically assigned with the "Exists" property of the "New" button SE value and name it "CRM Open"



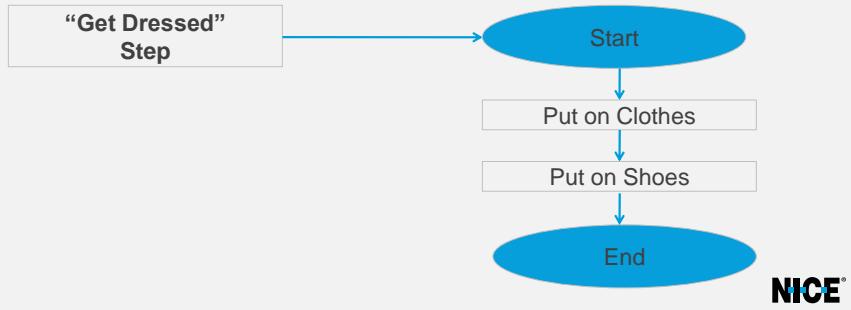
NICE®

Step Structure

- Steps hold Actions to perform



- Workflows can be nested under a certain Step, like a sub-procedure



- **To define a nested workflow for a workflow step:**

1. Define the workflow in the standard fashion
2. Select the step in the workflow that is to have a nested workflow assigned to it
3. In the Action tab of the Properties pane, select the name of the workflow to be assigned to the step in the Nested Workflow dropdown list. When the workflow reaches this step, it will run the nested workflow like a sub-routine.

Define Step Properties

- **Visibility**

Whether the Agent will see the step in the flow

An Agent can go to this step whenever he wants



- **Interactivity**

Whether the Agent can move back and forth between steps

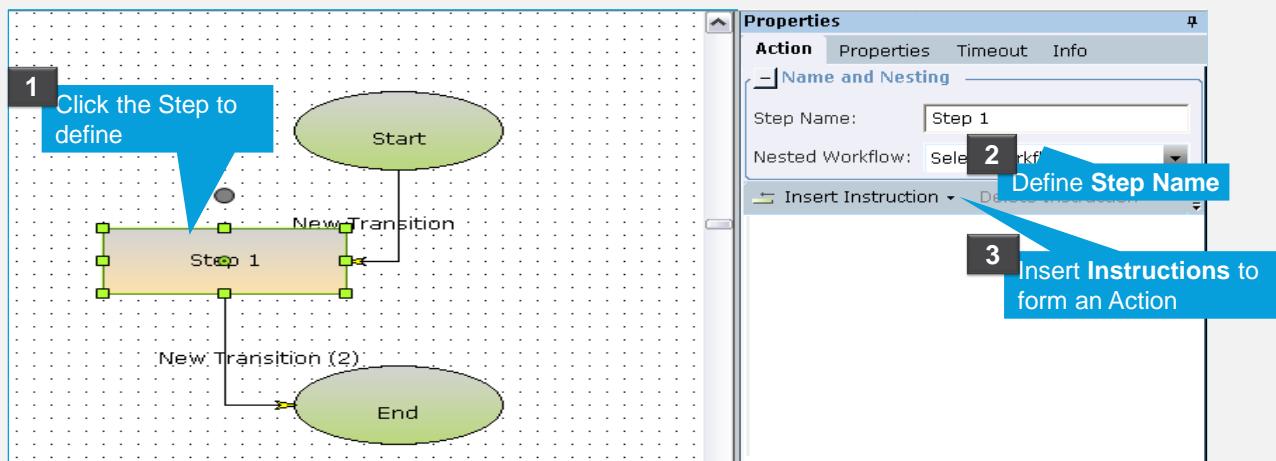
- **Timeout**

Set a limit of time to wait for any of the transitions to occur. Can be used for delay, error handling, and timed loops

NICE®

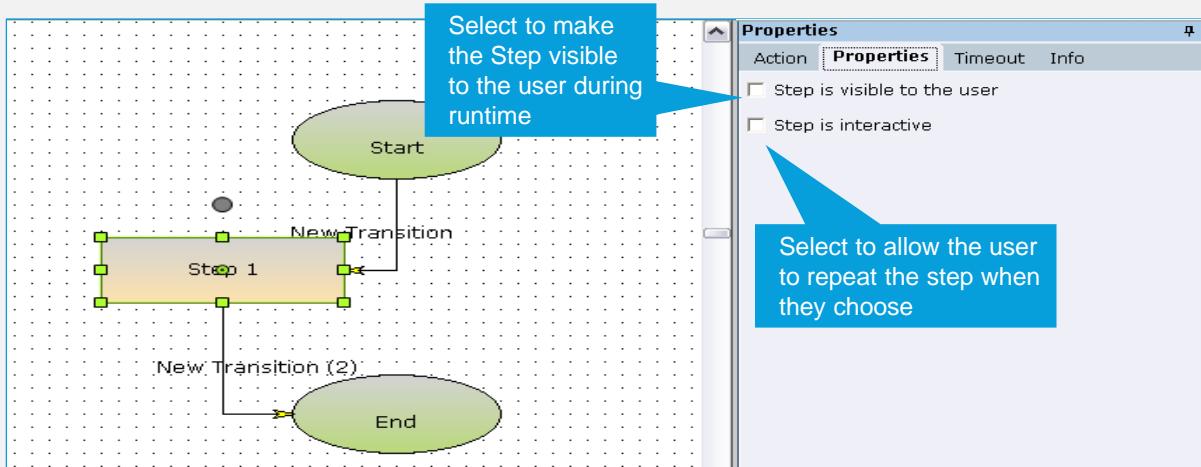
- Visibility – a step that performs internal system actions that are created “behind the scene” and will probably be invisible to the user.
- Interactivity - Only visible Steps can be Interactive! Make a step interactive for actions that can be done anytime during the process, for example - adding bonus points for a customer.
- Timeout – Allows you to define a limited time to wait for an Action to occur, in order to retry the Action upon failure, or to exit the Workflow.

How to Define a Step: Actions



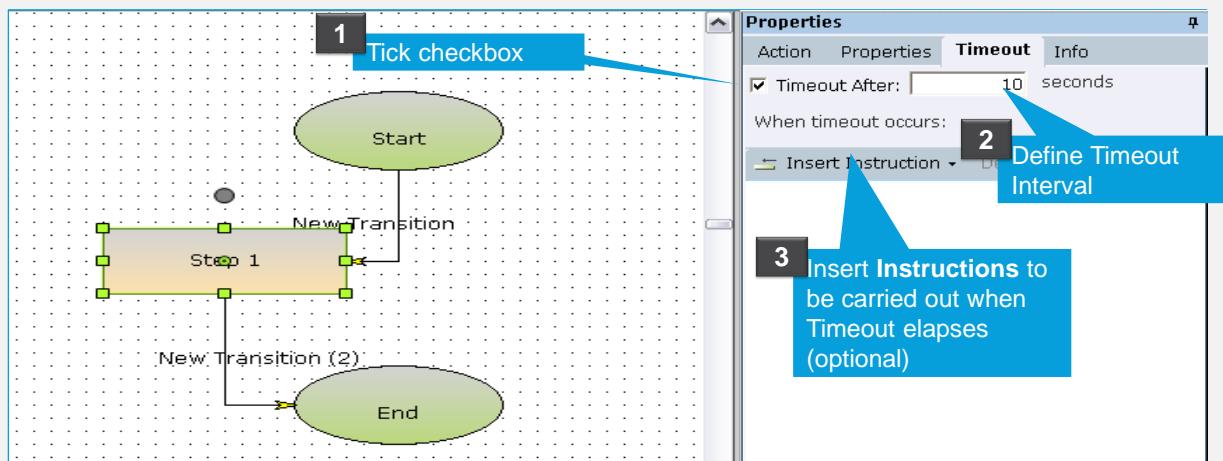
NICE®

How to Define a Step: Properties



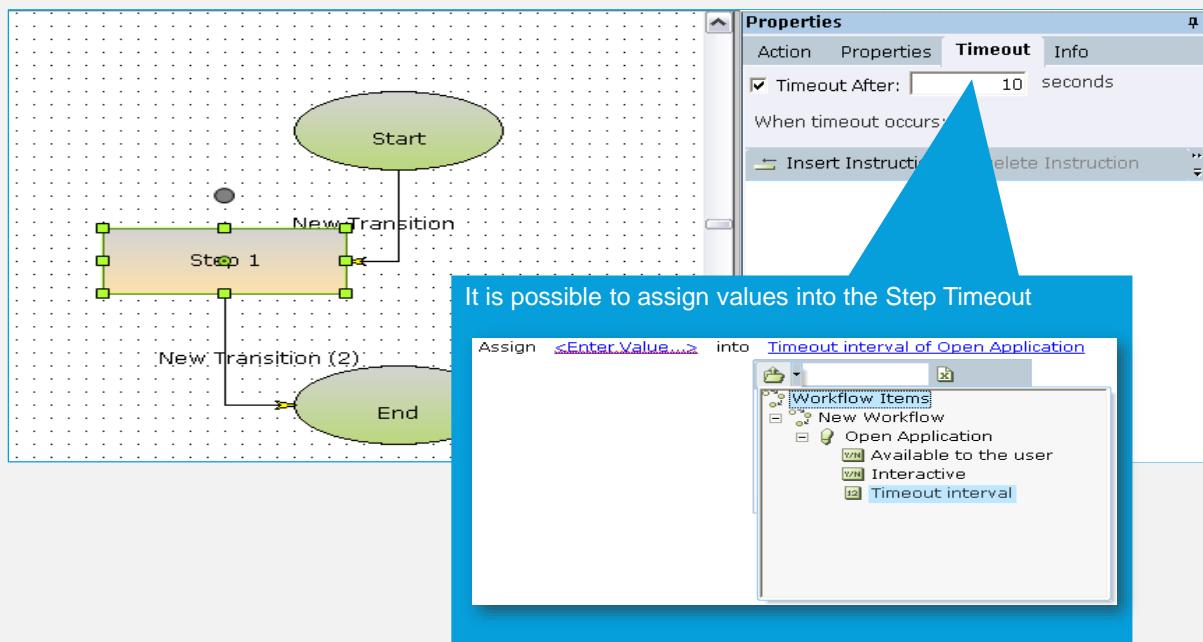
NICE®

How to Define a Step: Timeout



NICE®

How to Define a Step: Timeout



NICE®



Notes from Demo

DO IT YOURSELF

Defining a Step

- In each Workflow step you've created:
 - Make the step visible to the user
 - Add the instruction to show the Callouts
 - Associate the callout to the relevant SE

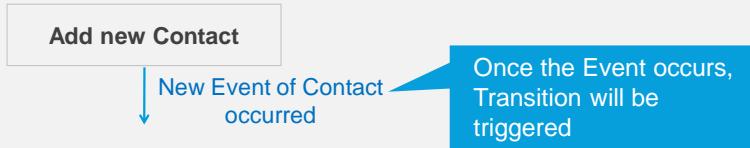
NICE®

Transition Structure

- Transitions define when the next Step is executed.

- You can define a transition by:

- An Event



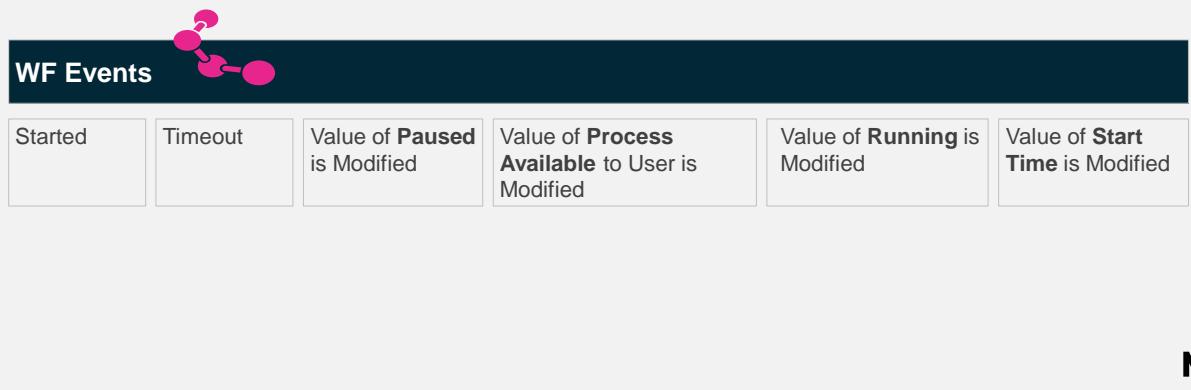
- An Event
Add new Contact
New Event of Contact occurred
Once the Event occurs, Transition will be triggered
 - A Condition Rule
Add new Contact
Contact's age is greater than 60
AND
Contact VIP equals true
Once the Conditions are fulfilled, Transition will be triggered

NICE®

- The definition of a Transition may also include an Action to be performed when the Transition is triggered and before going to the Next Step.

Workflow Events

- An Event-driven Transition can be triggered by:
 - Internal Events in the RT Designer, for example if a BE value is modified
 - External applications Events, for example, a CRM launch
 - Workflow Events. For every Workflow, Step or Transition the following Events are available:

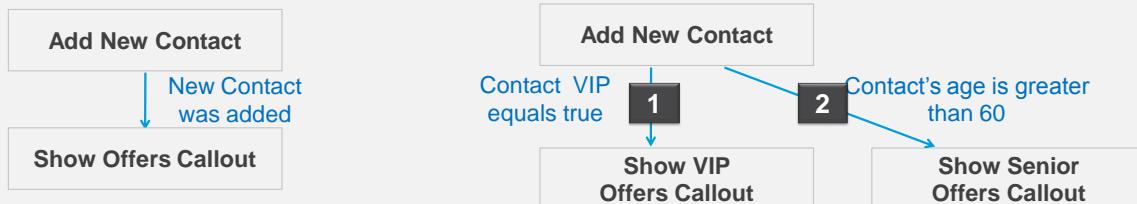


NICE®

- Workflow Events can be used to transition from one Step to another. For example, Transition to the next Step when the previous Step has Timed out.

Transitions and Steps

- For efficiency and maintenance, a single Transition should be defined for each direction (if needed) between each two Step boxes

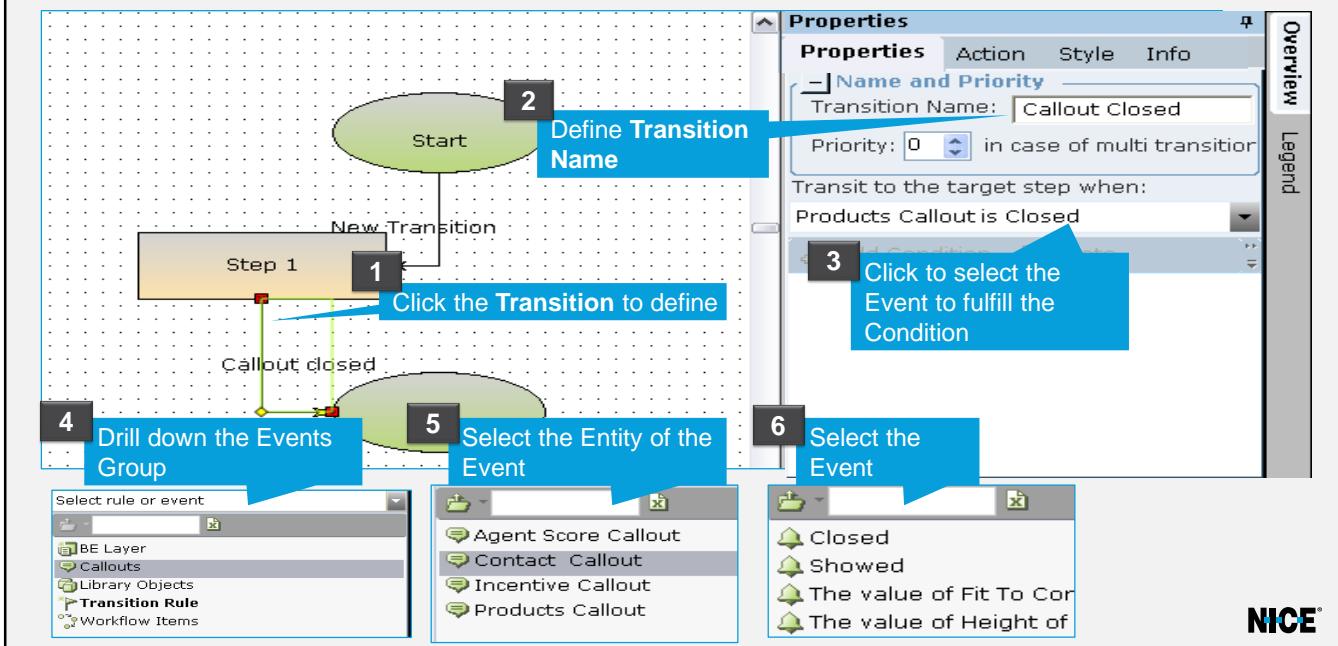


- If 2 transitions are defined to a single step, you must define their priorities

NICE®

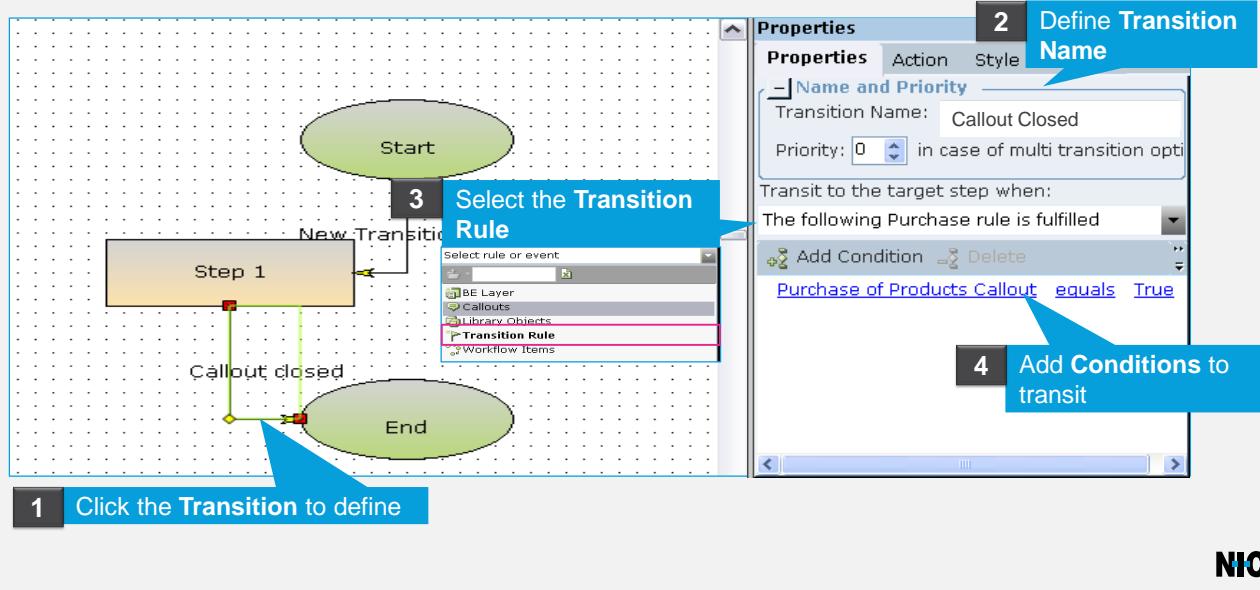
- **Priorities** - If both conditions are met, the transition with the higher priority will be executed, and the other will be discarded.

How to Define an Event-Driven Transition



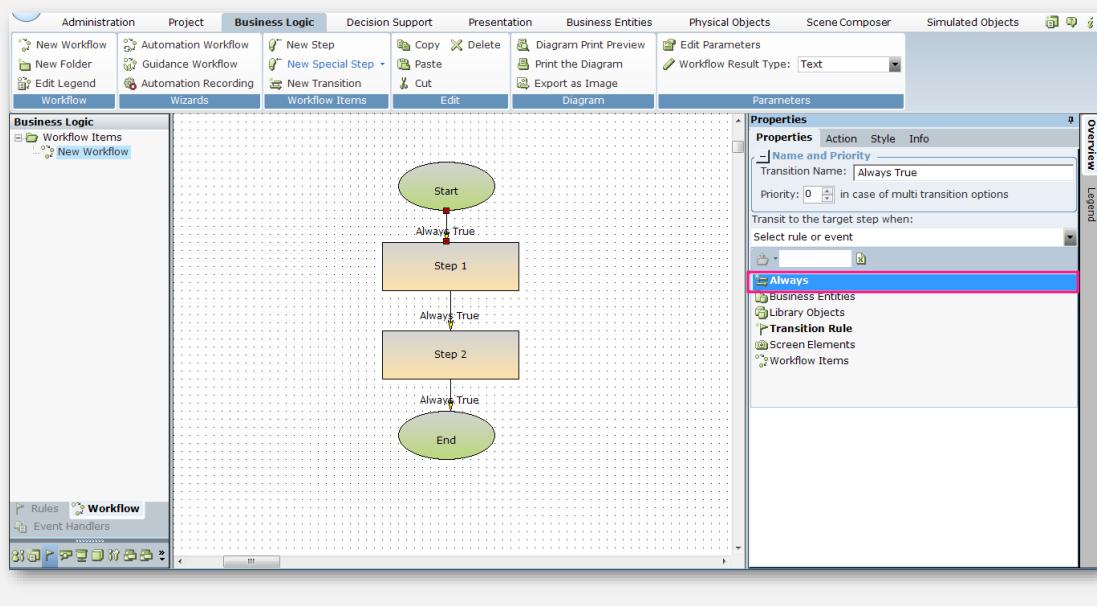
- **The best approach for defining Transitions** is related to the previous step. For example, if a step Opens a certain application window, then the transition to the next Step will be driven by the window displayed.

How to Define a Condition-Driven Transition



- Condition-driven transitions are actually Rules made on-the-fly.
When the condition is fulfilled, the Transition is triggered.

Always True Option for a Transition



- You can define a transition in your flow that is always true and automatically occurs without waiting.



Notes from Demo

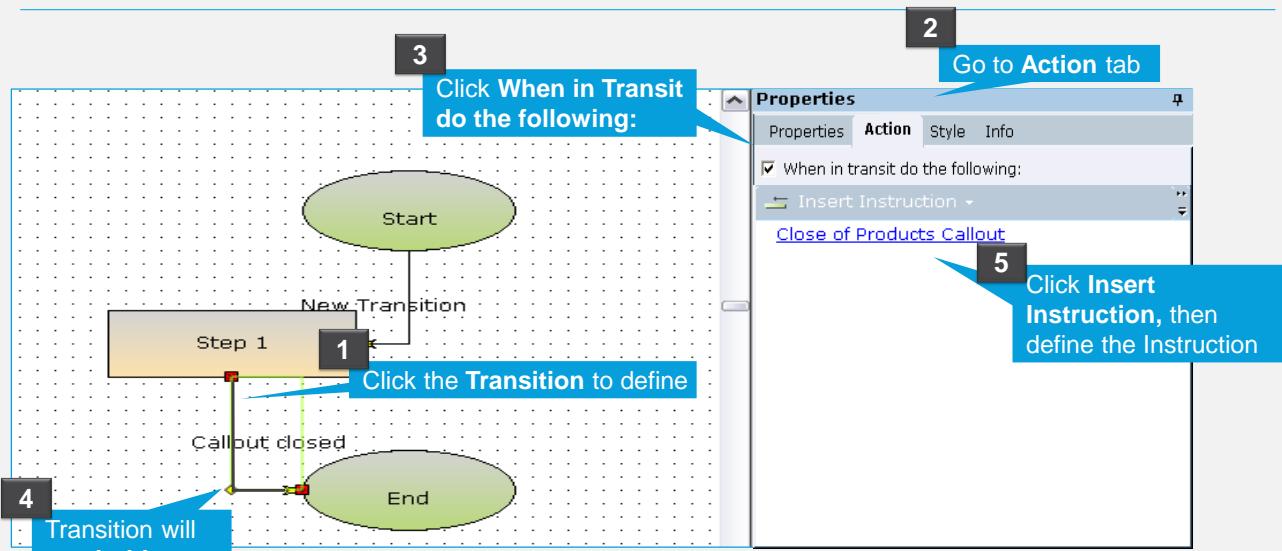
DO IT YOURSELF

Defining a Transition

- Define the transitions in your workflow to transit to their respective target steps as follows:
 - “CRM Opened”
 - When “Open” of “CRM Application” equals “True”
 - “New Account Window Open”
 - Based on BE event with the same name
 - “New Account Added”
 - Based on BE event with the same name

NICE®

How to Define an Action in Transition

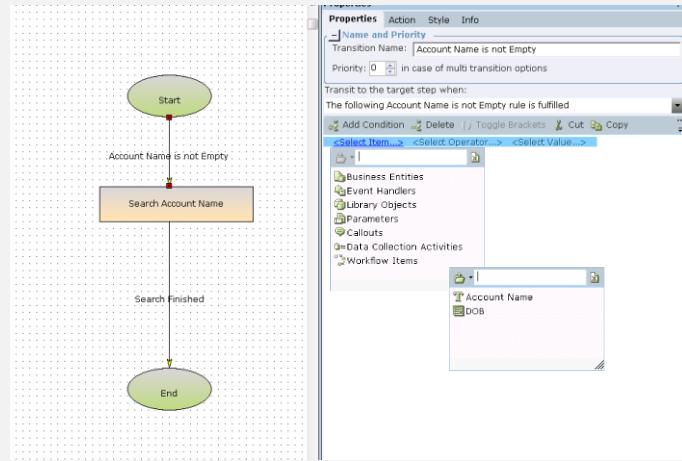


NICE®

- Actions you define in a Transition will be executed **while transiting**, meaning before the next Step.

Use Parameters in Transitions

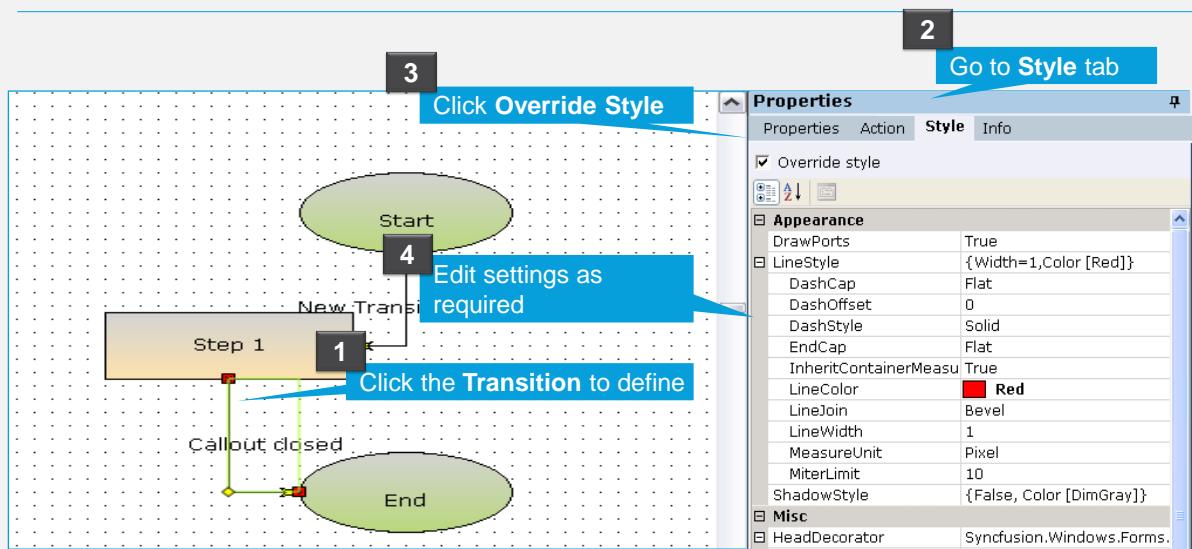
It is possible to use parameters in a transition rule:



NICE®

- Parameters can be assigned to a workflow in a similar manner as assigning a function.
- Workflows can only receive parameters that are primitive types or a list of primitive types.
- After parameters are assigned to a workflow, then when that workflow is invoked, parameters must be passed to it.

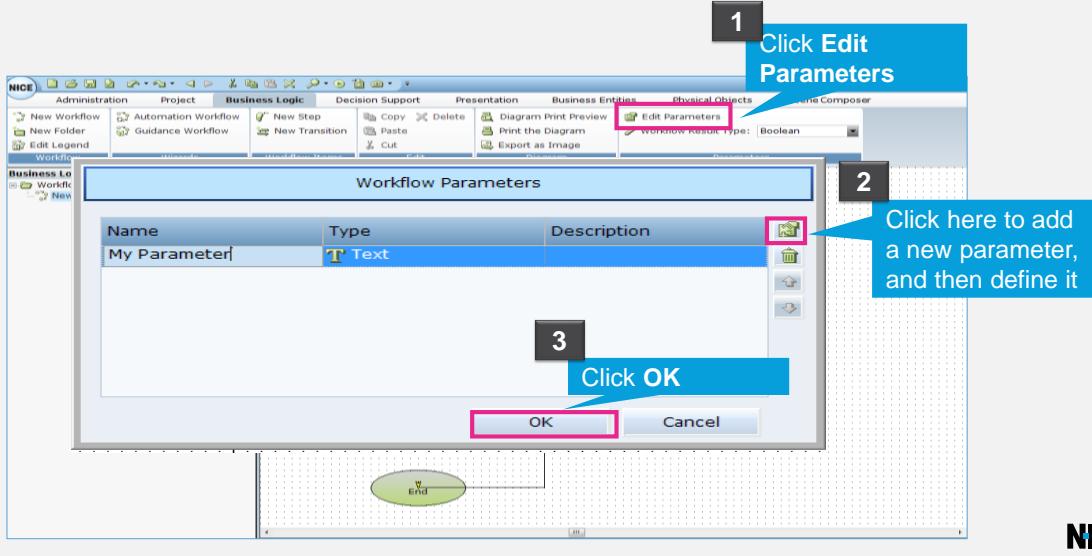
How to Define a Transition Style



NICE®

How to Add Parameters to a Workflow

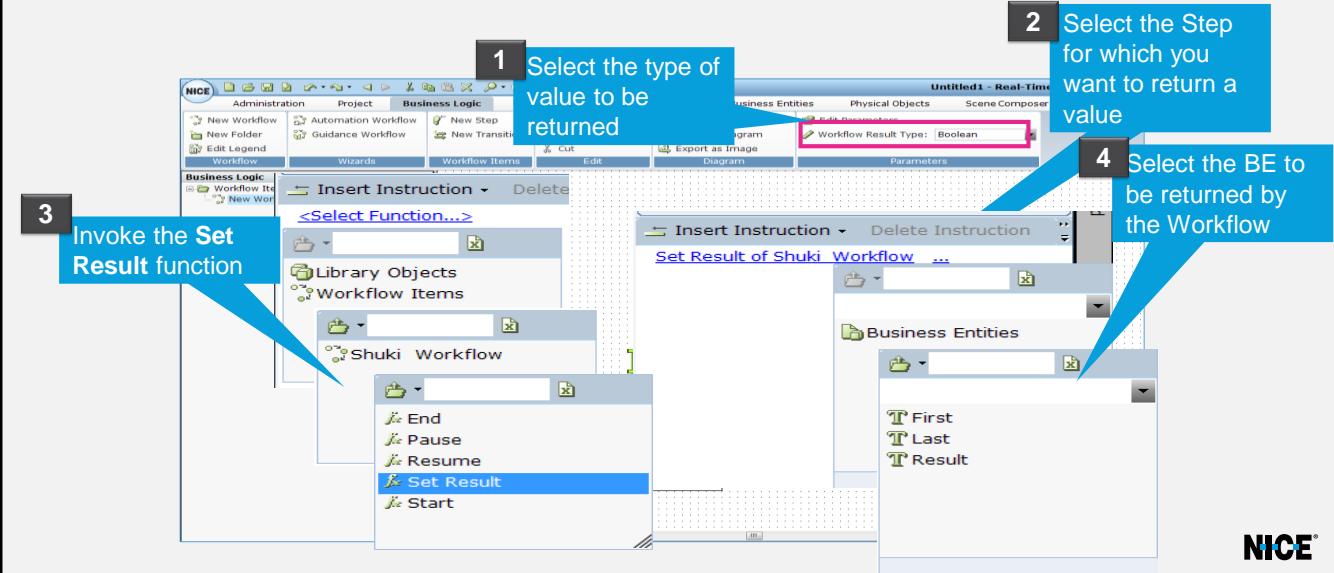
- Parameters can be defined for use within the workflow



- Workflows can only receive parameters that are primitive types or a list of primitive types.

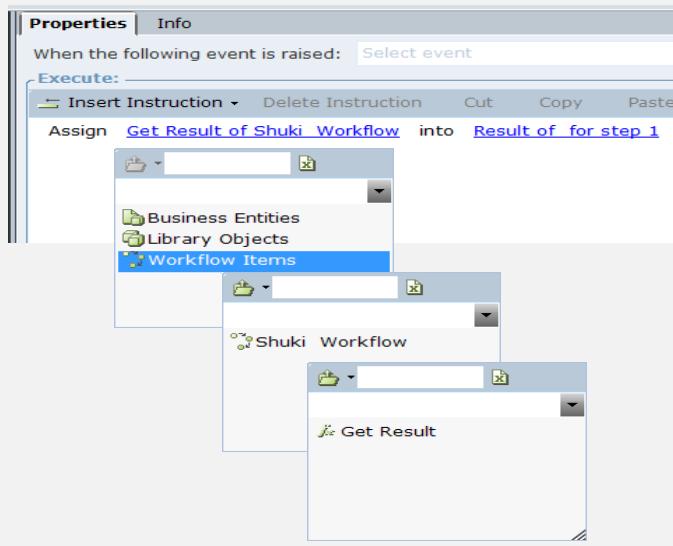
How to Return a Value in a Workflow

- A Workflow can return a result once the Workflow has ended



How to Return a Value in a Workflow

- Retrieve the Workflow result using the Get Result Function and Assigning it to a BE



NICE®

- The **Get Result** Function retrieves the result from a Workflow. It cannot be used within a Workflow, but can be used to retrieve the result from a Workflow once the Workflow has completed.

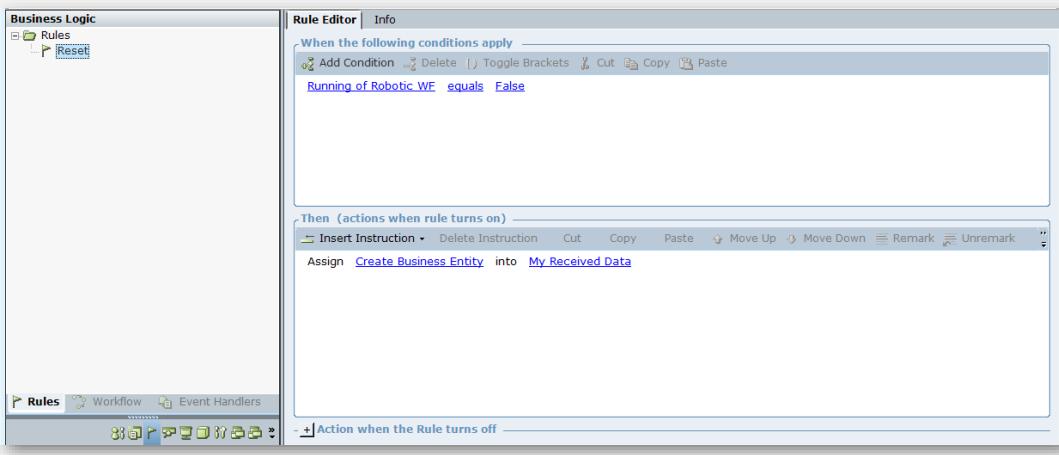
Where / When you should add logic to reset BE's values that are used in a workflow?

Question

?

Reset BE's values in a workflow

- It is recommended to add logic to reset the BE's values when the workflow is not running: Completed or Failed.

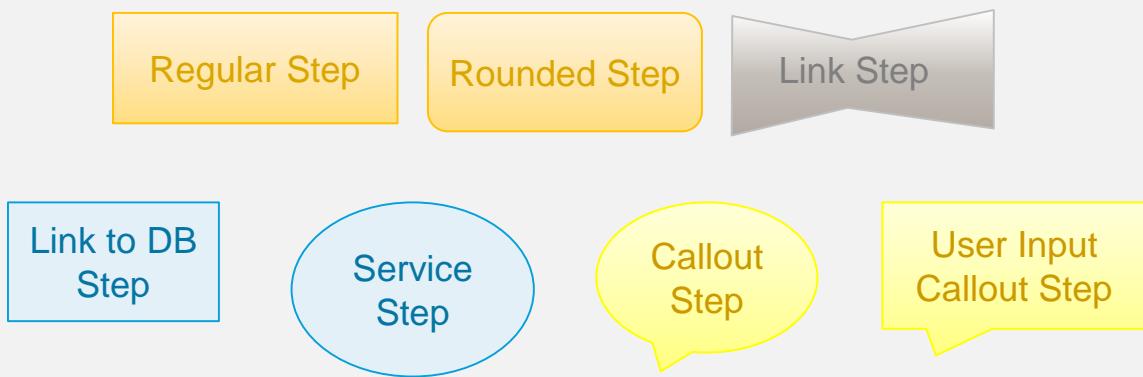


NICE®

- Business entities that are used in a workflow should be reset when the workflow is not running: Completed or Failed.

Workflow Legend

- RT Designer enables you to modify the Workflow's steps' visually so it can be readable at a glance

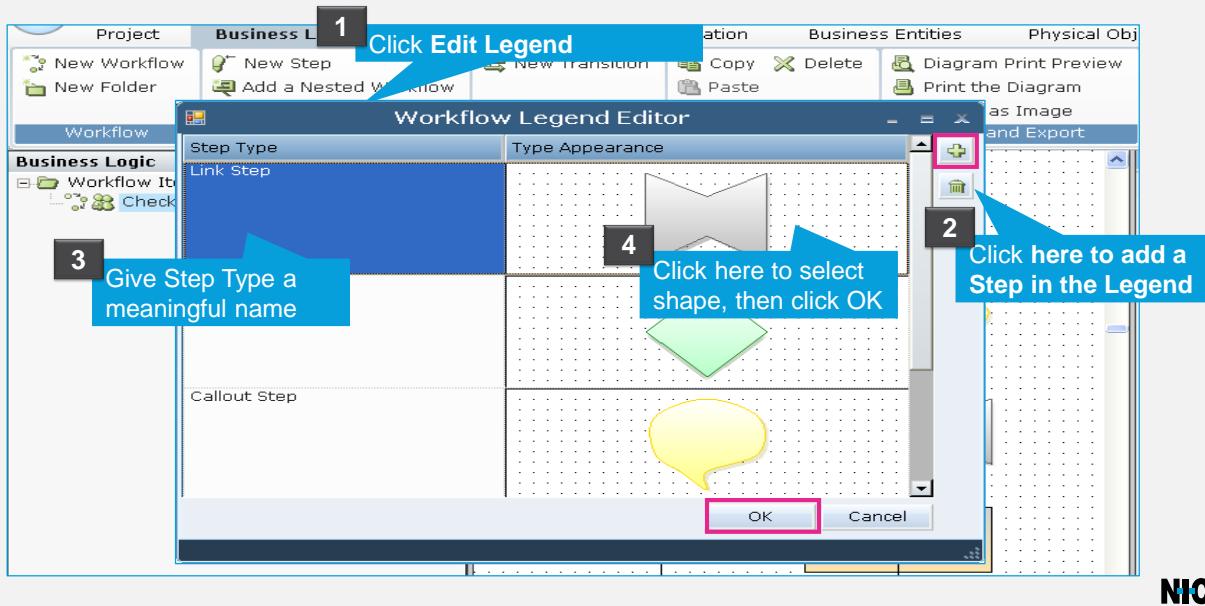


- This is called the Workflow's Legend

NICE®

- Changing the Step's appearance can also help in troubleshooting – get quickly to the step you are looking for.
- If you have several Workflows in the Project, all will be presented in one big Legend.

How to Create a Legend



NICE®

- First create step types in the Legend, then select for each step in the Workflow its relevant type.



Notes from Demo

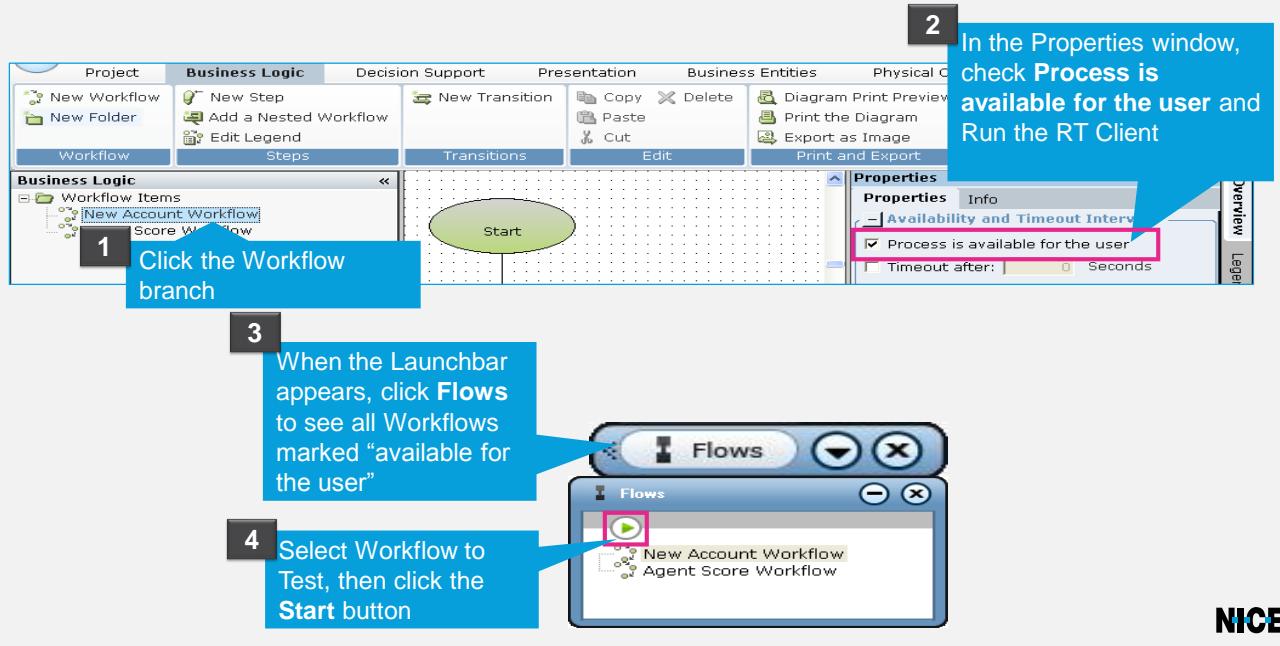
DO IT YOURSELF

Creating a Legend

- Create a new workflow to test legends
- Add three steps to the workflow
- Define three different legends
- Allocate a different legend to each step you've created
- Delete the workflow when you're done

NICE®

How to Test your Solution



- Flows window displays all workflows created in the project, which were defined for the User. From here, you can activate each Flow by demand.

How to Test your Solution

- The Workflow's Steps will appear as the flow progresses
- In the Monitor tool go to Flows Tab, to see which Step your flow is at
(Active Steps will be colored in green)



Click here to view steps according to their order instead of alphabetically

For Every Step you will see transition to it and End time

Name	Value
Active	False
Duration	0
Interactive	False
Start Time	01/01/1900
Timeout interval	20
Visible	True

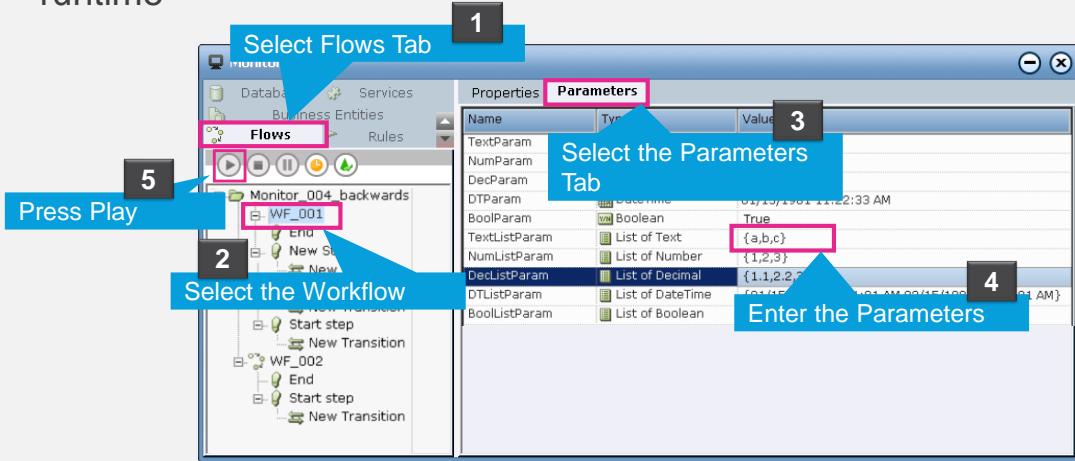
Step and Transition information will appear in real time in the right pane

NICE®

- Steps must be defined as “visible to user” in order for them to appear in the Flows Menu! To see how to make a Step visible to the User, refer to slide 9.
- Another way to test your Workflow can be by showing a testing Callout in a step (for example, with the text “I got here!”).

Triggering Workflow Parameters

It is possible to change the parameters defined in a workflow during runtime



NICE®

- Parameters applied in the Monitor must be invoked using the play button in the Monitor, and not from another source.
- For parameters of type List, there's no validation of the entered value. For other parameters provided by the user, the Monitor validates the content of the entered value.
- For selected parameters such as DateTime, the format is given, however for other parameters such as Text and List of Number, you must manually enter the values using the required format (in some cases, with parenthesis). Examples: List of Boolean: {true,false,true} Number: 123



Notes from Demo

DO IT YOURSELF

Testing your Workflow

- Mark the ‘Guide’ workflow: “Process is available for the user”
- Test your solution
- While the workflow is running:
 - View the workflow’s reaction to the actions you take
 - View the appropriate values in the Monitor application

NICE®

Summary

- Create Workflows – Steps and Transitions
- Test Workflows

NICE®

- Workflows are a series of steps performed in a certain order, and transitions that define when to move between steps.
- Workflows can be triggered by the Agent's demand from the Flows Workflow, or by a Rule or a change in a BE.
- Steps hold Actions to perform.
- Transitions define when the next Step is executed, and can hold an Action to perform before transitioning to the next Step.
- Parameters can be assigned to a workflow in a similar manner as assigning a function.
- When calling a workflow, it is possible that the workflow will return a result value.
- Workflows can be nested under a certain Step, like a sub-procedure.
- Workflows can be created using the Guidance Workflow Wizard intended to guide an Agent



Thank You



NICE®

SOLUTION PUBLISHING



Lesson Objectives

By the end of this lesson you will be able to:

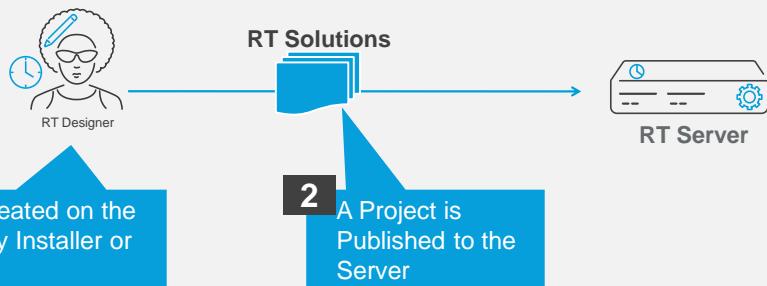
- Publish a Solution
- Assign Solutions to Teams



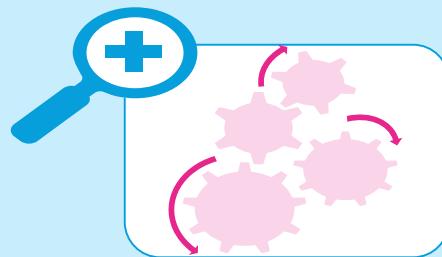
Solution Publishing



Recall: Design to Deployment Flow



Let's see what happens during publishing...

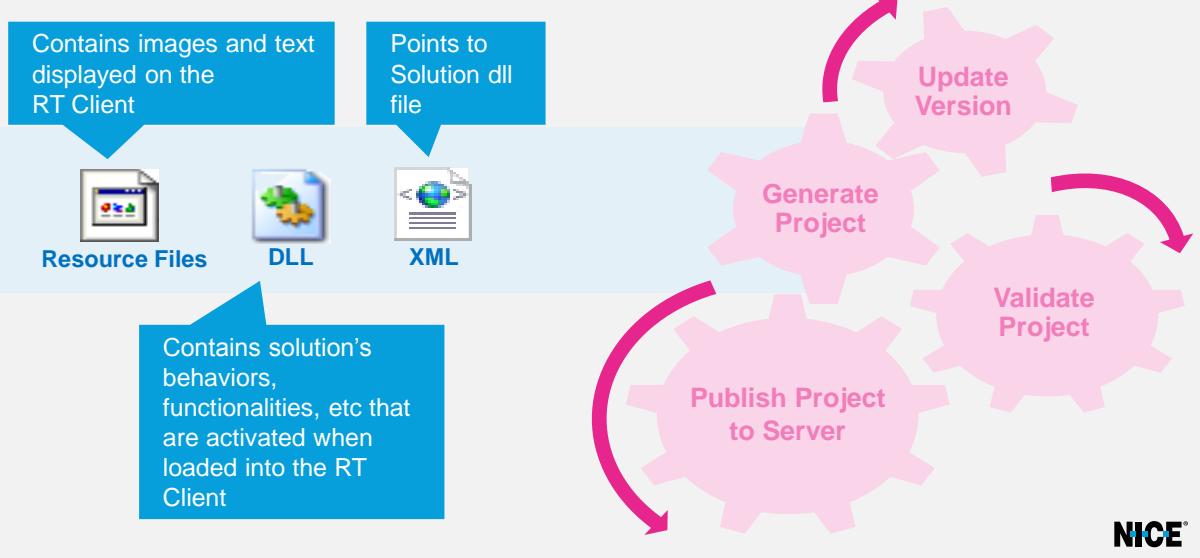


NICE®

- RT Server is the core engine of the Advanced Process Automation Solutions .
- The RT Designer and the RT Client both act as clients to the RT Server, which in turn approves each request at a time.

Publishing a Project

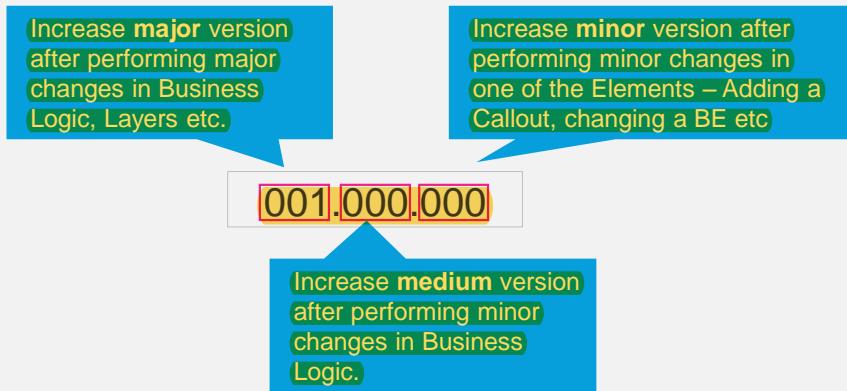
- The Publishing process performs the following:



- Updates the Project Version:** Each time you publish a project, the project's version is updated automatically.
- Validates the Project:** RT Designer compiles the project and checks for errors. If the project has errors they are displayed in the Messages area at the bottom middle of the window. You can jump directly to the window in the RT Designer to edit that object by clicking on it in the Messages area.
- Generates the Project:** If the project does not have any errors, the project files are created. These files are saved under the folder Generated Solutions, which is located in the Installation folder.
- Publishes the Project to the Server:** RT Designer uploads the project files to the RT Server.

Version Control

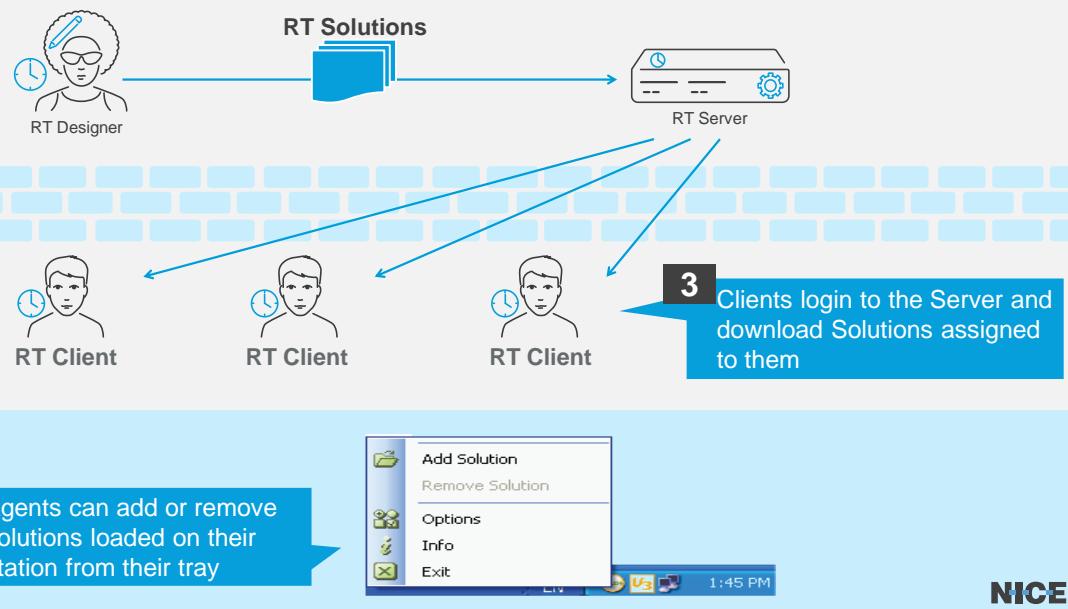
- Solution deployment mechanism allows you to control the version published



NICE®

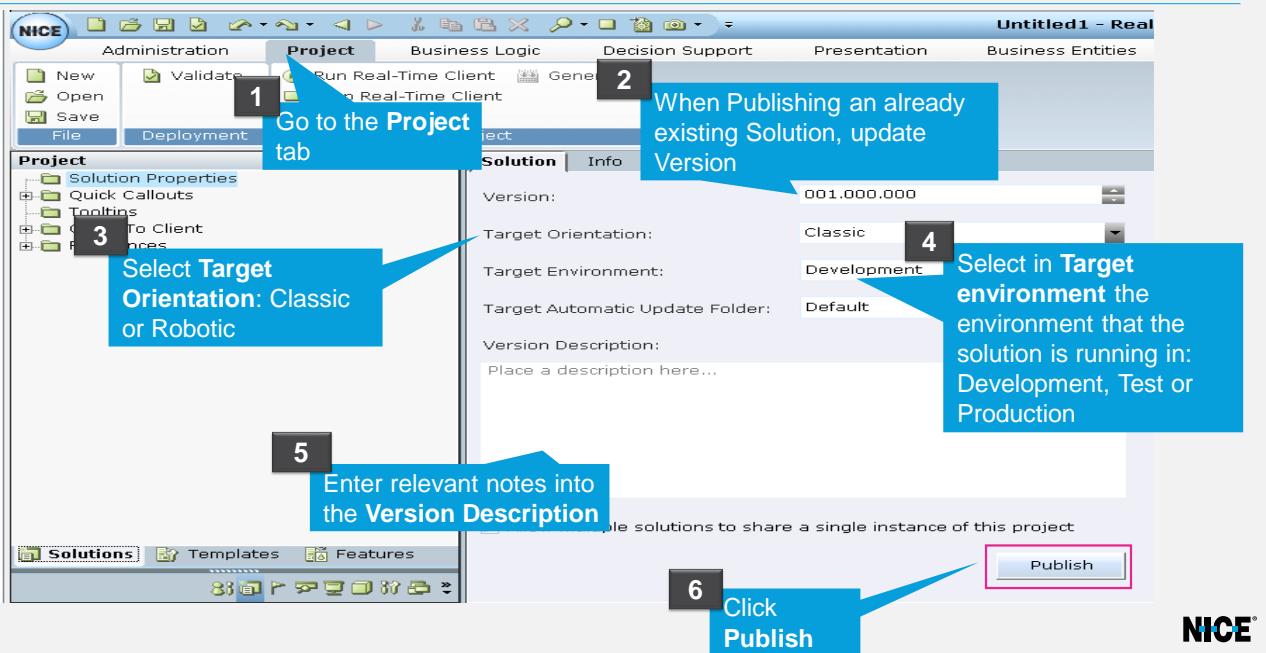
- Each time you publish a project that was published before, the Project's Version is updated automatically from what you define here.

Recall: Design to Deployment Flow



- RT Solution deployment mechanism has several built-in capabilities:
- Version control
- Separate development / test / production environments
- Automatic distribution of updated solution and files
- Server DB connection

How to Publish a Project



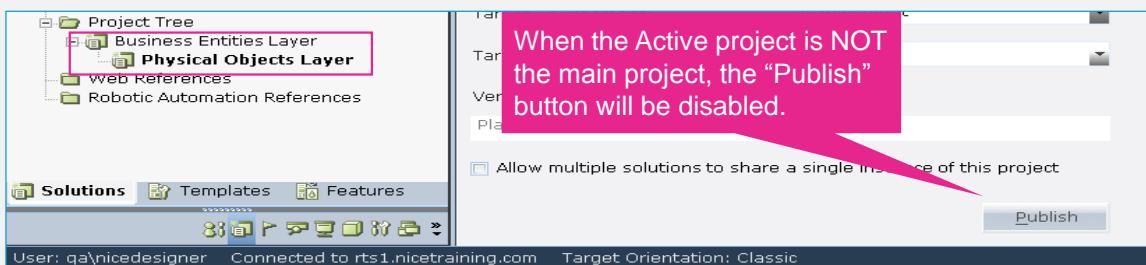
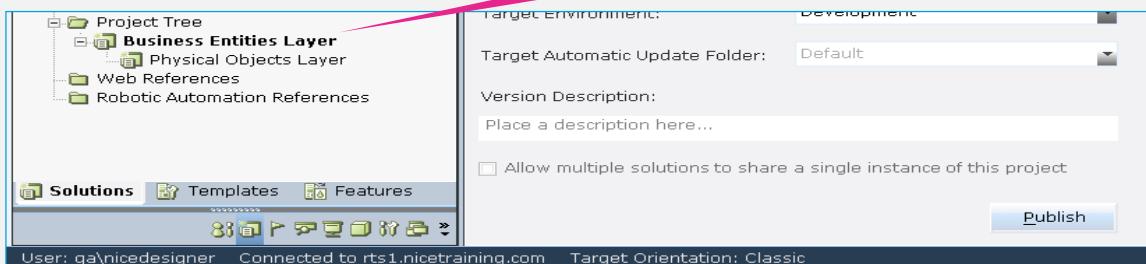
- **Target Orientation:** Indicates the Type of the RT Client in which this Project is deployed. When there is a Thin Client and there are no group assignments, you need to configure deployment at the settings window → Designer → Orientation Items → Web so that instead of dll and XML files, java script files will be generated.
- **Target Environment:** Indicates the purpose of this Project, as follows:
 - **Development:** Specifies that the project is still under development.
 - **Test:** Specifies that the project is ready for testing.
 - **Production:** Specifies that the project is ready to be deployed to Clients.

Version Description: The Version Description only provides a description of the current solution version, and is not a description of the solution itself. After publishing, the Version Description is cleared.

- **Allow multiple Solutions to share a single instance of this project**: In RT Solution implementations involving a great deal of content, you may need to break down the solution file into separate solution files. In such cases, some objects will likely be common to all solution files, such as the value of BEs pointing to screen elements. When you select this checkbox, common objects are shared in memory during runtime, which helps to improve performance. This feature can apply to any object on the project level.

How to Publish a Project

Publishing a solution can only be performed on the main project



NICE®

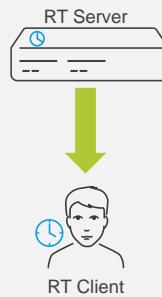
Solution Assignment



NICE®

Solution Assignment

- After publishing a Solution, the system administrator assigns Solutions to teams
- When a user from the team logs in to the RT Client, they download the Solutions assigned to them
- Assignment includes support of the following features:
 - Solutions Assignment
 - Update after assignment
 - Load on Startup

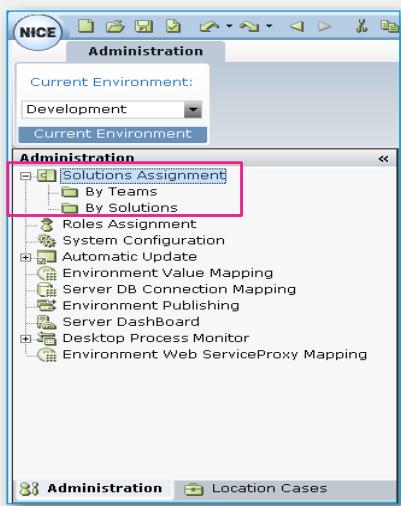


NICE®

- Features explanations:
- **Automatically or Manually assign a Solution to a team:** Will be covered in the coming slides.
 - Automatically: Whenever a Solution is Published, a dialog box will appear to select the teams to assign this Solution to.
 - Manually: Administrator assigns a Solution manually through the Designer GUI – in the Administrator tab.
- **Update after assignment:** When a Solution is assigned to teams, and then an updated version is published, the publisher gets a dialog box that asks whether the current assignment should update the new version or not.
- **Load on Startup:** To support large scale Solutions, where the implementation is broken down into several solutions, only a single main solution should be loaded at start-up and other solutions should be loaded/unloaded according to the context of the interaction.

Solution Assignment in the Designer

- Solution assignment is done from the Administration tab in the Designer, either by Teams or by Solutions.



By Teams

Select a team and assign it Solutions



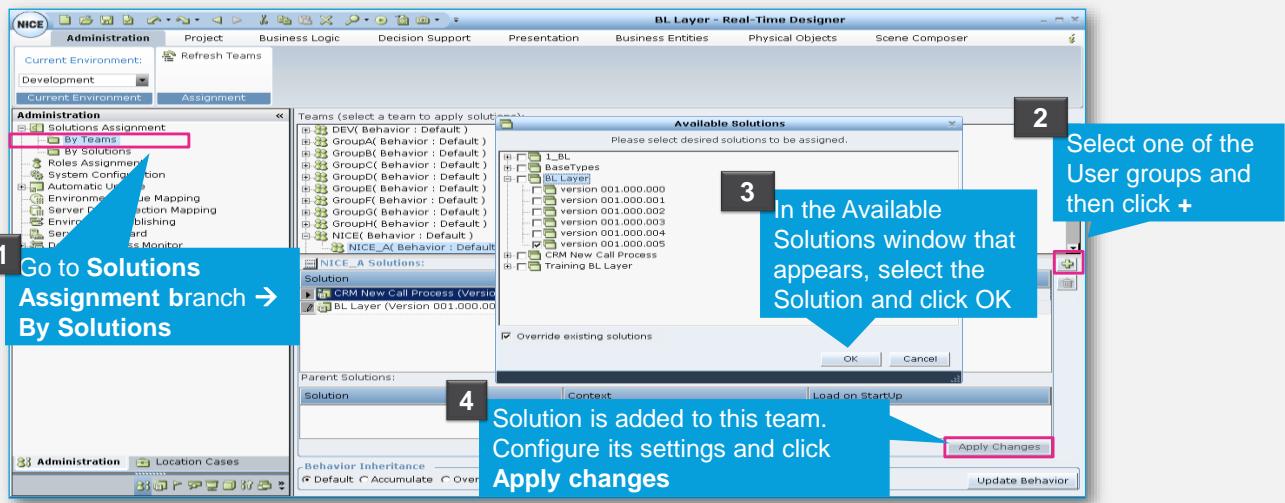
By Solutions

Selects a Solution and assign it teams



NICE®

How to Assign Solutions to Teams



NICE®

- This branch allows assigning a Solution or multiple solutions per team.
- **Teams** branch: Displays a hierarchical tree of the user groups imported from the corporate LDAP server, such as Active Directory. These should represent Agent teams.
- **Available Solutions Window:** Lists the projects that have been published to the RT Server. A solution may appear more than once, each with a different version.
- **Solution Settings (Step 3):** Context specifies the context of the solution, for example citrix, desktop and so on. The solution assignment is assigned to a team in the given context for each RT Client installed. Upon login, the RT Client is assigned to a team and given a role (for example, supervisor) by the RT Server during authentication within the context.

Inherited Solutions - Optional

The selected group **searches up the hierarchy for a Solution**

- Default**: The group behaves as it used to before: If a user group (in which a user is a member) does not have a Solution assigned to it, then the system can search for a Solution to return to the user. To do so, it searches upwards in the User Groups hierarchy (parents). It continues to search upwards in the hierarchy until a Solution is found. Once a group that has a Solution(s) assigned to it is found, the search is terminated, and the identified Solution(s) is (are) returned to the user.
- Accumulate**: The group receives group Solutions **on top of their own Solutions**
- Override**: The child group Solution **overrides** the mother group Solution

Parent Solutions:

Solution	Context	Load on StartUp
NICE Core Cell Project (Version 001.000)		
BL Layer (Version 001.000.005)		

Behavior Inheritance

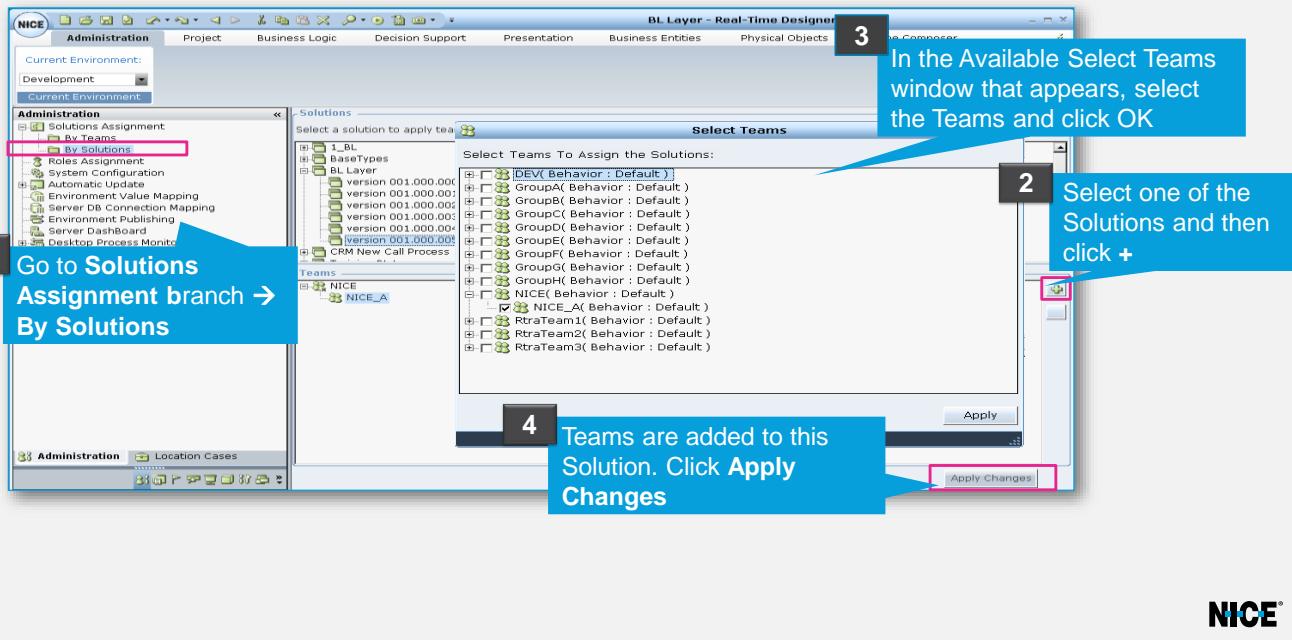
Default Accumulate Override

Remember to click Update Behavior when done

NICE®

- Solution assignment hierarchy enables the delivery of combined Solutions to the group
- Child groups can inherit the Solution from their parent groups
- Inheritance is optional
- **Default:** The group behaves as it used to before: If a user group (in which a user is a member) does not have a Solution assigned to it, then the system can search for a Solution to return to the user. To do so, it searches upwards in the User Groups hierarchy (parents). It continues to search upwards in the hierarchy until a Solution is found. Once a group that has a Solution(s) assigned to it is found, the search is terminated, and the identified Solution(s) is (are) returned to the user.
- **Accumulate:** The group receives group Solutions **on top of its own solution**
- **Override:** The child group Solution overrides the mother group solution

How to Assign Teams to Solutions



NICE®

- This branch allows assigning a team or multiple teams to a Solution.
- **Solutions branch:** Displays a hierarchical tree of the Solutions that were published to the RT Server.
- **Select Teams Window:** Lists the teams that have been imported to the RT Server.



Notes from Demo

DO IT YOURSELF

Assigning Solutions/Teams

- Assign one of your published Solutions to your team, or assign your team to one of your published Solutions.
- Whichever method you used, verify that your actions are verified using the other method.

NICE®

Summary

- Publishing a Solution
- Assigning Solutions to Teams



- Publishing a Solution performs the following actions:
 - Updates the Project Version
 - Validates the Project
 - Generates the Project
 - Publishes the Project to the Server
- After a Solution is published, it must be assigned to the Agent's team in order for him to download it.
- Assignment includes support of the following features:
 - Automatic / Manual assignment
 - Update after assignment
 - Load on Startup



Thank You



NICE[®]

AUTOMATION



Lesson Objectives

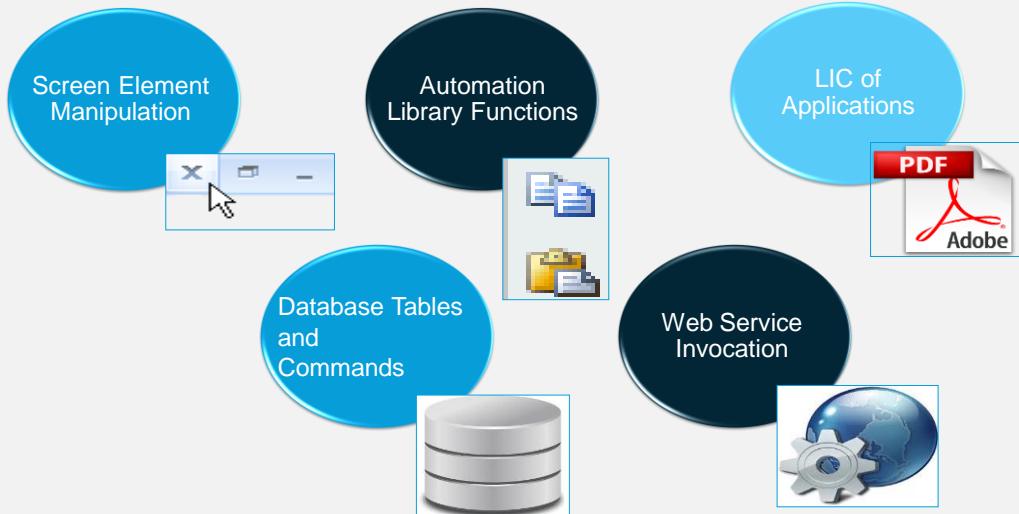
By the end of this lesson you will be able to:

- Automate processes
- Handle special cases



Process Optimization

- The goal is to reduce handling time by automating human actions

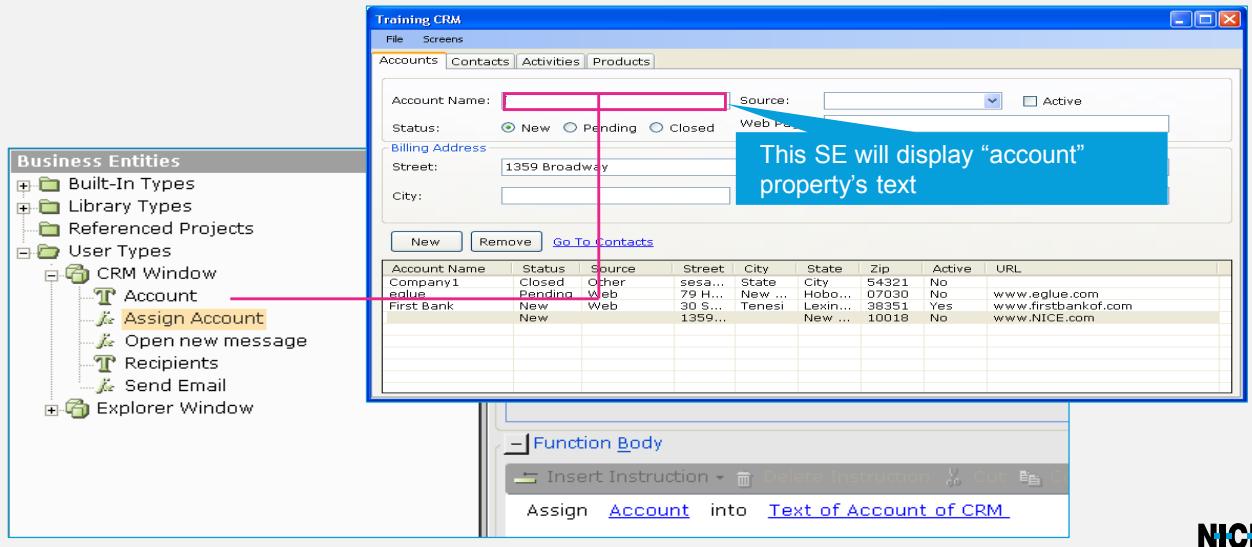


NICE®

- Real-Time is meant to support all efficiency-driven projects.
- Screen Element Manipulation**
 - Enables you to mimic the user's actions on Screen Elements, for example – close a window, type in a field etc.
- Automation Library Functions**
 - Enables you to use system functions, for example – copy to clipboard, get active process etc.
- LIC of Applications**
 - Allows you to launch any application – PDF, WORD etc.
- Database Tables and Commands**
 - Allows you to create DB tables and different command objects
- Web Service Invocation**
 - allows you to import web services and harness their functionality

Screen Element Manipulation

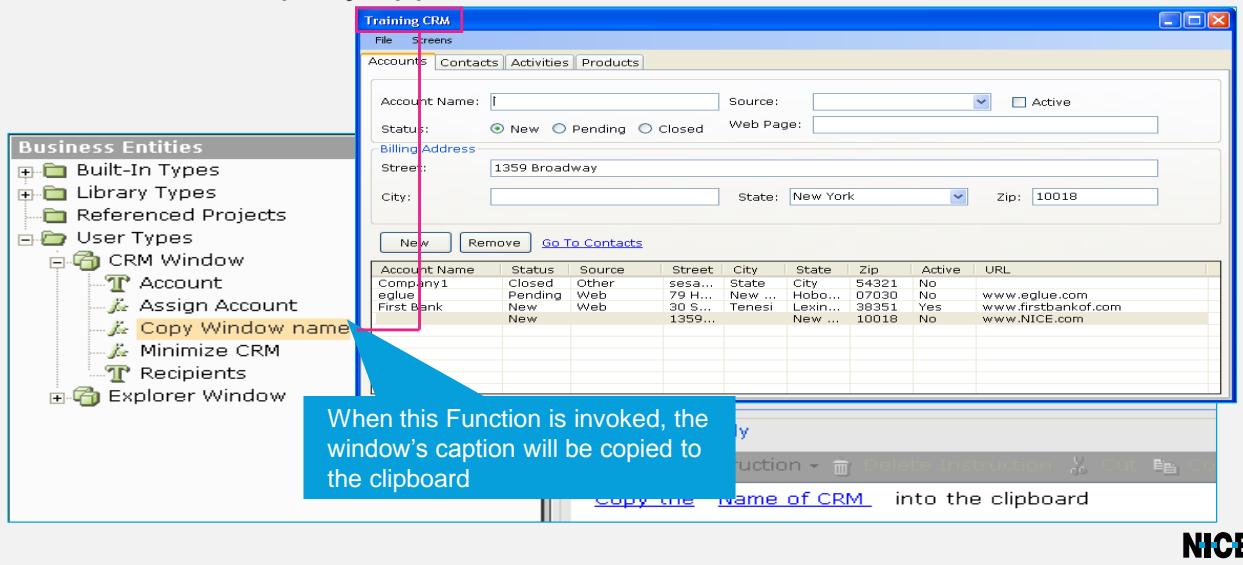
- Assign Value when invoking a Function



- The SE value will change to Subject's Text whenever the Function is invoked, typically from the Logic Layer.

Library Functions

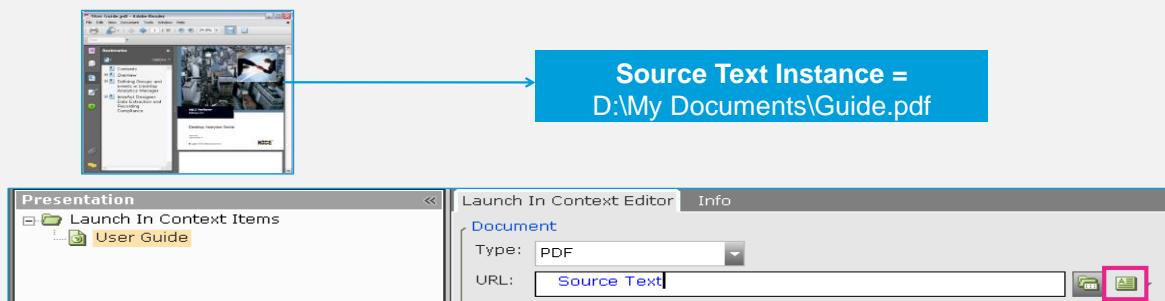
- Automate third party application's Functions



- These Functions help you to complete the automation process, by imitating the user's actions.
- Functions for Win 32, Web, MS Office, File and Clipboard application based are supported.

LIC of Applications

- Assign any URL/path to an LIC Object

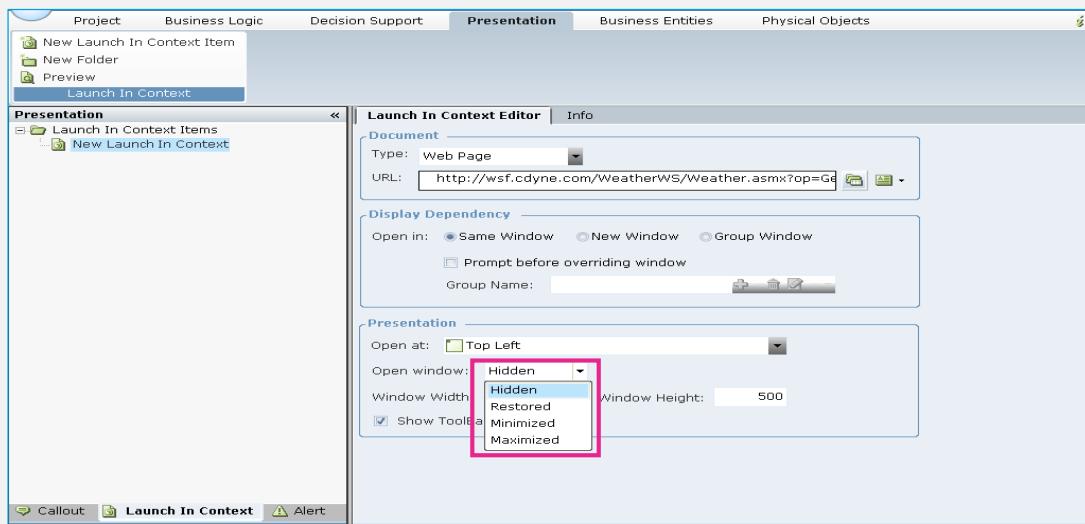


NICE®

- Launch In Context allows you to launch applications automatically, and then proceed. This is used for example, in cases where automation performs actions in third party applications.

LIC of Applications

- Launch an LIC item as Hidden



NICE®

- LIC preferences allow Launching an application or other documents as hidden / restored / minimized / maximized can help during automation implementation.

Advantages of Automation

Reduce Handling Time



- Perform several Actions simultaneously

Process Enforcement



- Prevent mistakes / malicious activities

Consolidated Desktop



- Prevent the need to work with multiple applications

NICE®

- Several automation Workflows can operate simultaneously, in addition to the Agent's actions at that time.
- You can enforce an Actions' order, to make sure there is no step overlooked / forgotten.
- Copy information from application A to application B.

What can you Manipulate?

- Launch any Application



Internet Explorer



Designer



Windows Media Player



Microsoft Office Outl...



WinZip



HP Solution Center

- Invoke any Function



Win 32

Is Window Available

Set focus to main Window



Web

Copy text to Clipboard

Get Active Process



MS Office

Create Outlook Email message

Create Outlook task



Clipboard

Clipboard Text

Copy Text into the Clipboard



File

Copy / Delete File

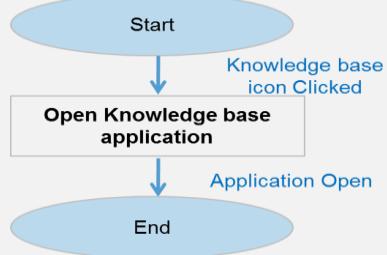
Does File Exist

NICE®

- Automation by itself is straight-forward. But there are few things to take under consideration, which we will learn about in the next few slides.

Automation Concept

- Sequence of Actions and Conditions

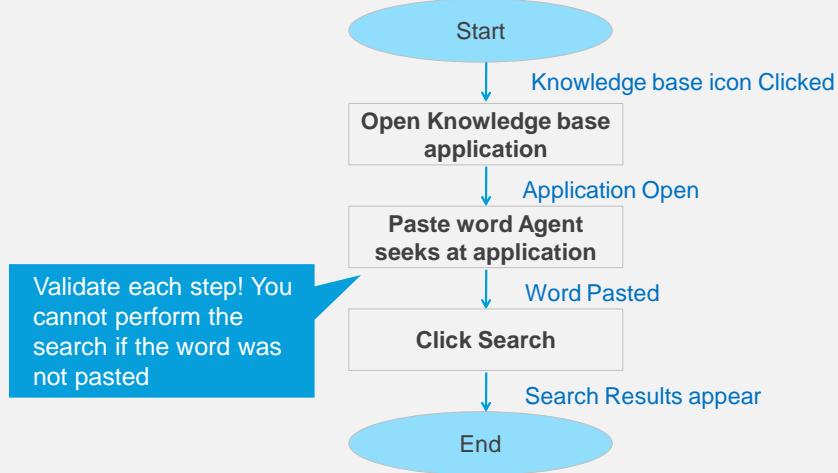


NICE®

- Creating an automation solution is similar to creating any workflow or implementing a logical process where the steps define the actions to be executed and the transitions are used to verify these actions were indeed performed as expected.

Automation Concept

- Transitions validate that each of the expected Actions were performed successfully

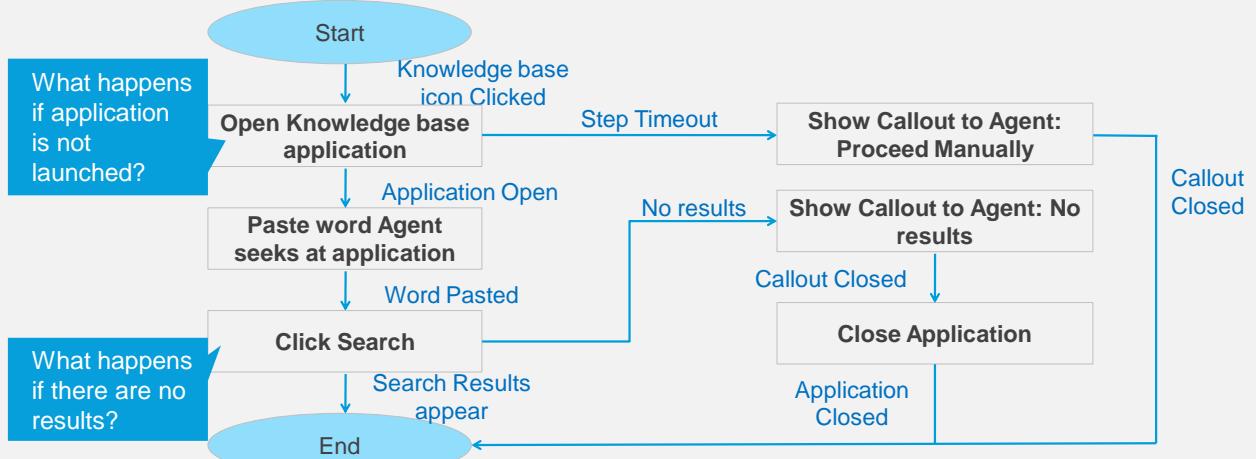


NICE®

- Transition should include validating the expected result from the previous step.
- Since the order in the Workflow is important, validation is crucial.

Automation Concept

- Cover ALL possible scenarios

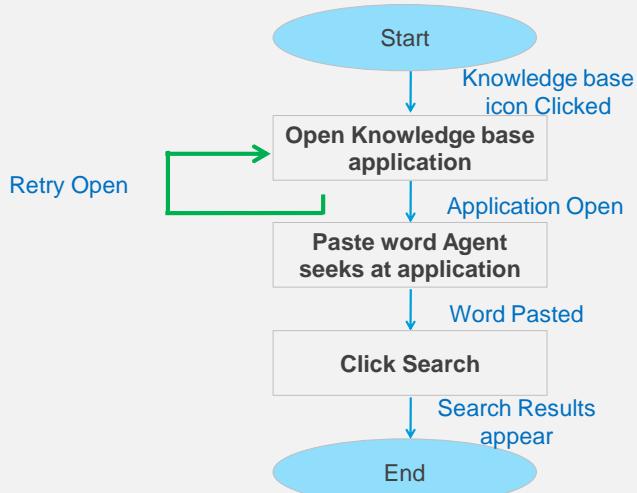


NICE®

- If one of the Workflow's Steps or transitions is unexpected, the Workflow is disrupted and the Agent will experience delays. To prevent this:
 - Use Timeouts
 - Notify the Agent in case of failure

Automation Concept

- Create Transition Loop



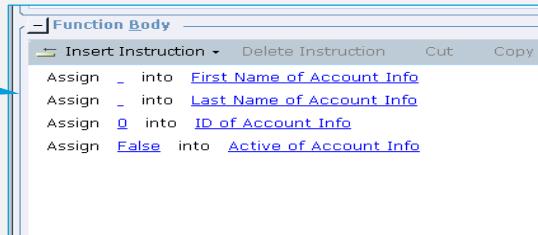
NICE®

- A common use of Transition Loops is for retrying a failed step.

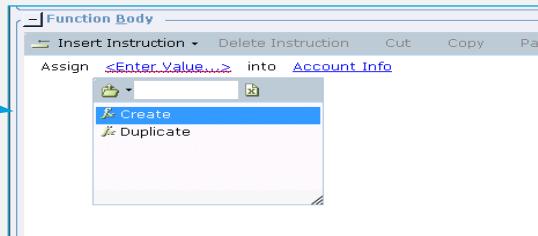
Automation Concept

- Reset BEs between iterations if necessary

Assign empty or default values



Use the **Create** function to reset composite type BEs



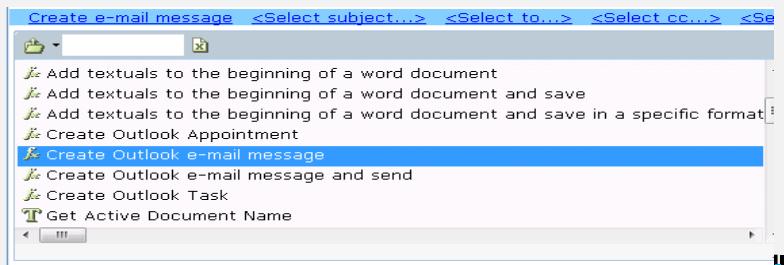
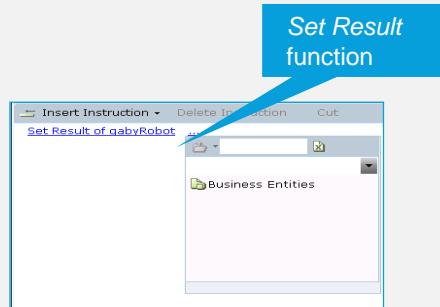
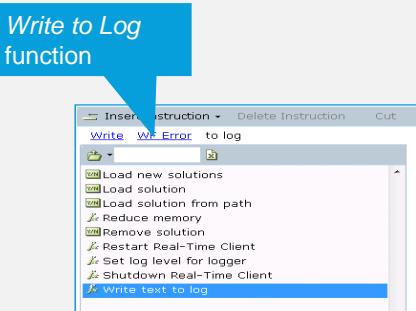
OR

NICE®

- In some cases the leftover data from the previous iteration might create unexpected behavior. In such cases a reset operation is recommended.

Automation Concept

- Error handling



NICE®

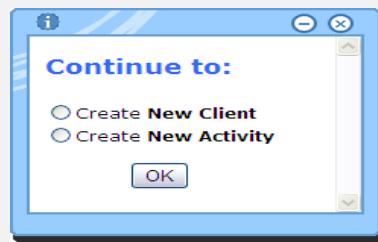
- When running an automation process many factors should be taken into consideration, so when designing a solution, it is recommended to gather error related data and pass it on in one or more ways.

Considerations

1. Notification
vs.
Act behind the scenes



2. Automation
vs.
Human decision



NICE

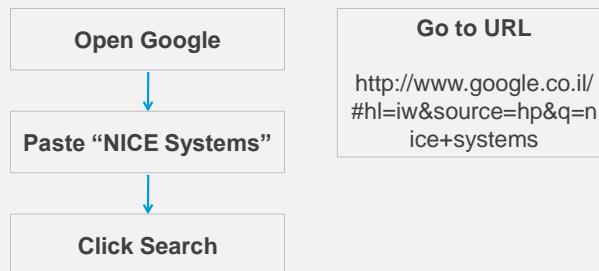
- Notify the Agent that a process is automatically being made:
- To indicate when automation is finished
- To prevent the Agent from interfering or performing the Actions themselves
- Operate behind the scenes (minimize the application where automation occurs) when:
 - You wish to save time, and allow the Agent to perform some other action simultaneously.
 - The process display is not vital to Business Process
 - Not all steps need to be automated, sometimes it is better to leave the Agent the decision of how to proceed. Whenever there is a human factor to consider, give the Agent a choice. In other words, automate the trivial actions and leave the sensitive decision points for the Agent.

Considerations

3. Full Automation
vs.
Information retrieval



4. Set of Actions
vs.
Final target

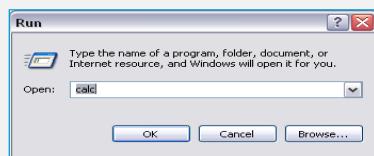
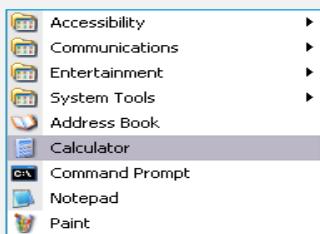


NICE®

- When information to proceed is fully provided – automate!
When information is missing in order to proceed – ask for it from the Agent.
- When possible, Shorten a set of Actions to a single step, in order to prevent disruptions in between steps.

Considerations

- All roads lead to Rome... Define Transitions so they validate all habits and working methods



NICE®

- Different people work in different methods, consider ALL possible scenarios to avoid Workflow disruption.
- This example shows that there are many ways to open the Calculator application. Prevent covering only the straight-forward one.
- A wrong Transition in this case would be: when the Calculator shortcut is clicked. A right one would be: when the Calculator is open.

Summary

- Automating processes
- Handling special cases

NICE®

- Automation is a sequence of Actions and Transitions
- Transitions validate that each of the expected Actions were performed successfully
- Automation uses:
 - Reduce Handling Time
 - Process Enforcement
 - Workforce Management
- Workflows can be created using the Automation Workflow Wizard intended for automation using LICs and Callouts.



Thank You





ROBOTIC AUTOMATION PART 1



Lesson Objectives

By the end of this lesson you will be able to:

- Describe the concept of Robotic Automation
- Configure Real-Time Solutions for Robotic Automation in the Real-Time Designer
- Create a Robotic Automation Workflow
- Invoke a Robotic Automation Solution using a Web-Service



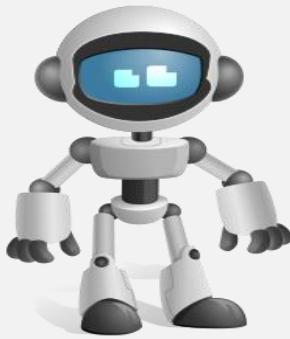
WHAT IS ROBOTIC AUTOMATION



NICE®

What is Robotic Automation

- Robotic Automation is a solution in the APA suite which enables triggering and running of automated workflows

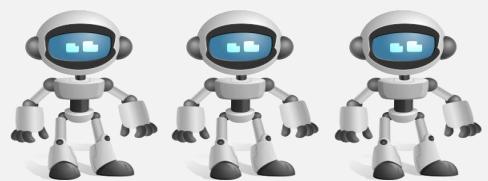


NICE®

- The Robotic Automation solution enables applications to remotely activate workflows that are exposed on the Real-Time server.
- Applications can benefit from the functionality provided by these workflows by sending execution requests.

What is Robotic Automation

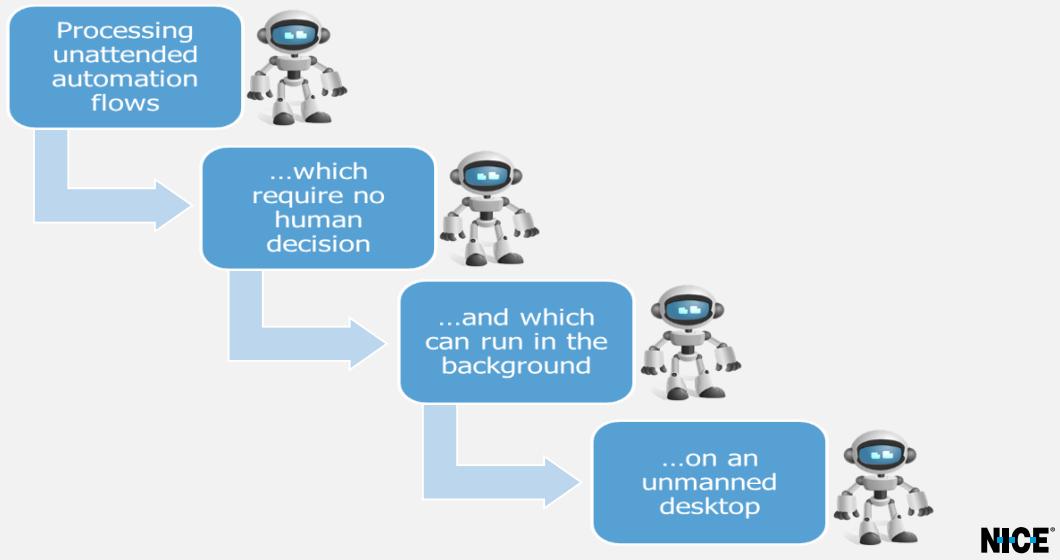
- The typical back office employee needs to perform many mundane repetitive day to day tasks
- Robotics Automation can perform these tasks automatically, freeing the employees to engage in more complicated tasks



NICE®

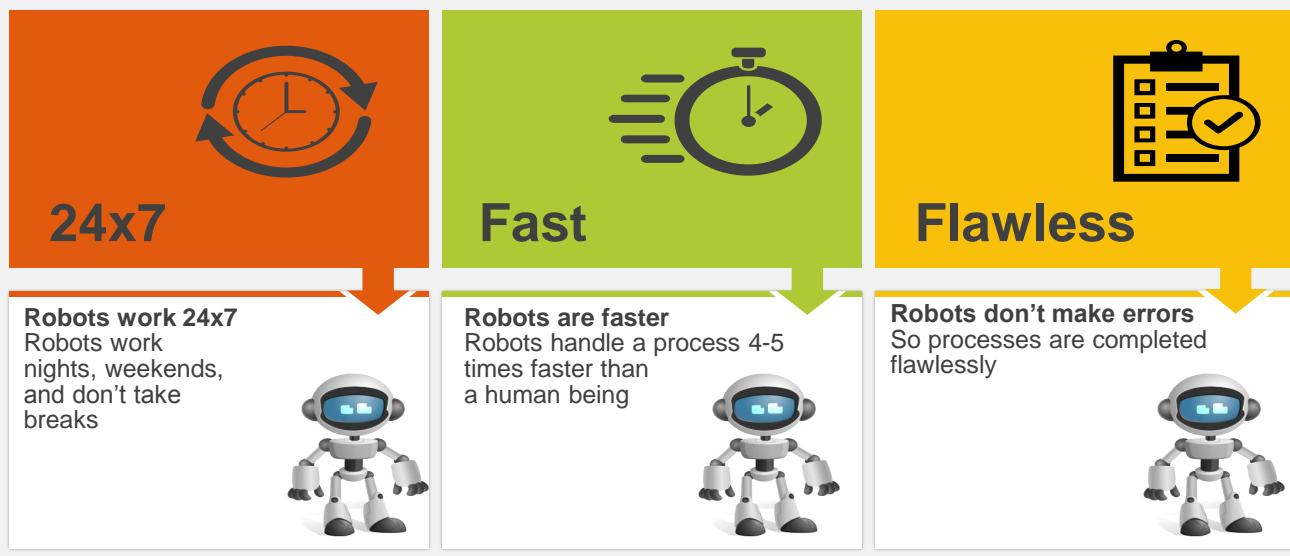
What is Robotic Automation

- Robotic Automation is all about:



- This is where NICE Robotic Automation can be of great value.
- **The whole purpose of Robotic Automation is to process unattended automation flows, which require no human decision in order to be carried out, and can run entirely in the background, on an unmanned desktop.**

Robotic Automation - Pros



NICE®

Robotic Automation - Pros

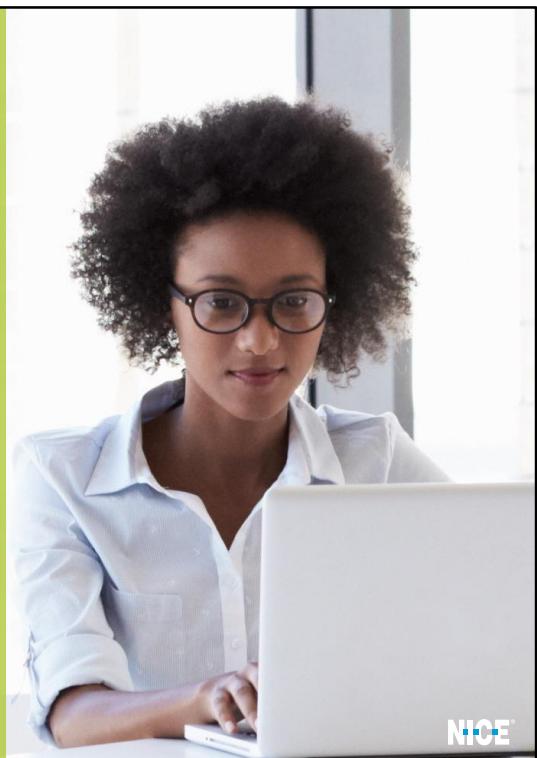
Robots can automate any employee desktop activity



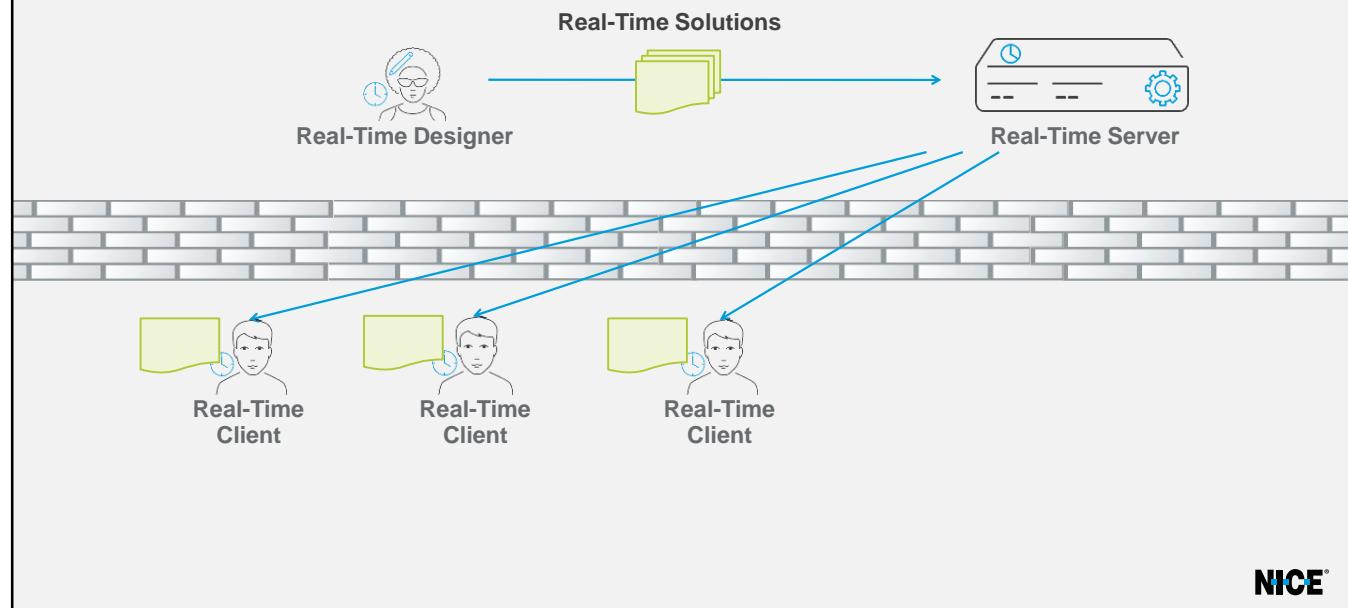
NICE®

- The idea behind the automation aspect of Robotic Automation is the same as that of the regular Process Automation feature that is part of the RTPO package, in that the robots can automate any employee desktop activity, such as mouse selection, field entry, copying and pasting, screen navigation, etc., but this time, without the presence of an agent at the workstation.

ARCHITECTURE AND FLOWS



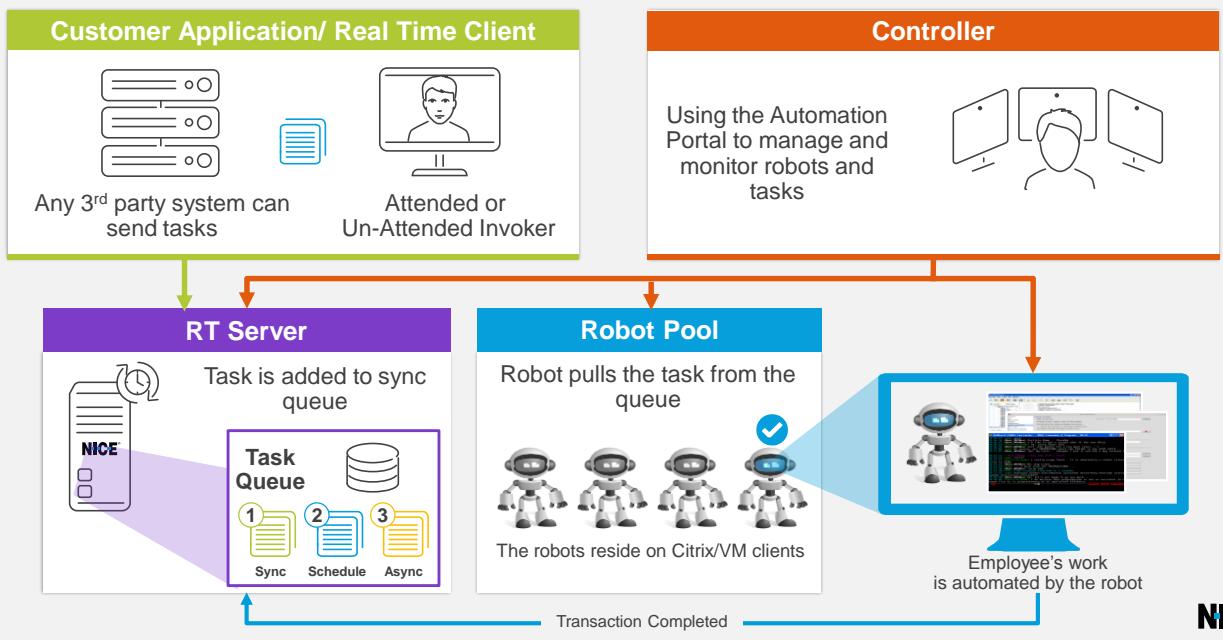
Design to Deployment



NICE®

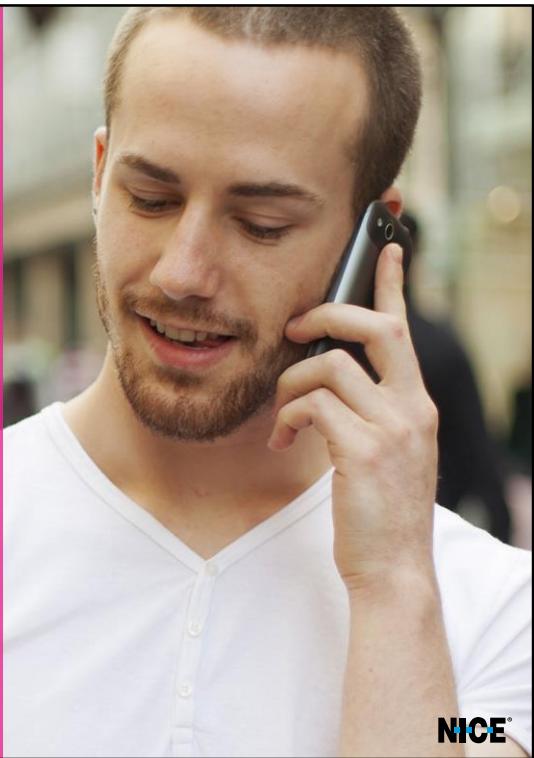
- We're already familiar with the typical Design to Deployment architecture, whereby the solutions are designed on the RT Designer workstation and published to the RT Server, so that when RT clients log in, they can download the solutions that have been assigned to them from the RT Server.

Robotic Automation High-Level Flow



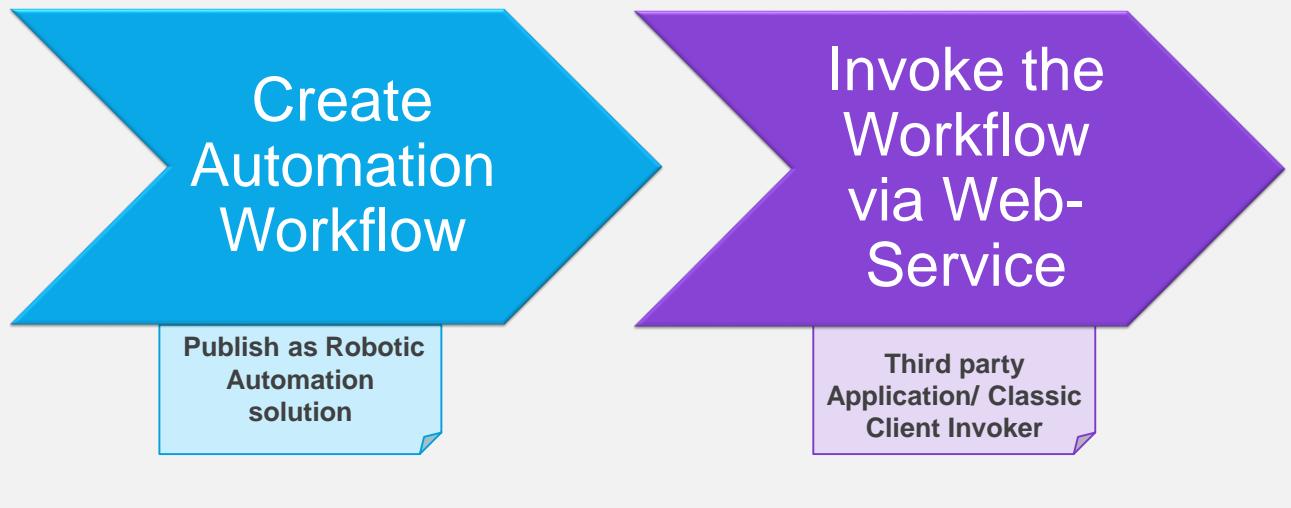
- The third-party application, or the classic RT Client, requests via web service that a specific workflow in a specific solution be invoked with the relevant parameters and prioritize the task.
- That request is received by the RT Server and put into the Task Queue according to its Type and given priority. Once processed by a robot assigned with the relevant solution, the completed transaction and its data are stored in the RT Operational Database for reporting purposes.

IMPLEMENTING A ROBOTIC AUTOMATION SOLUTION



NICE®

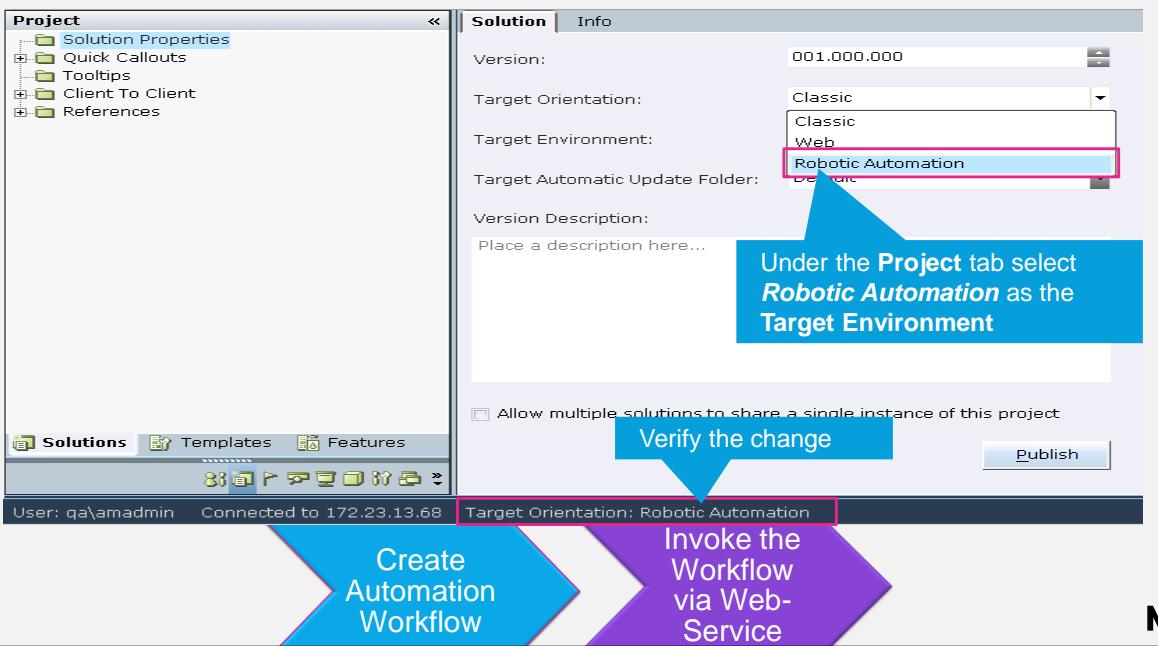
Robotic Automation Solution



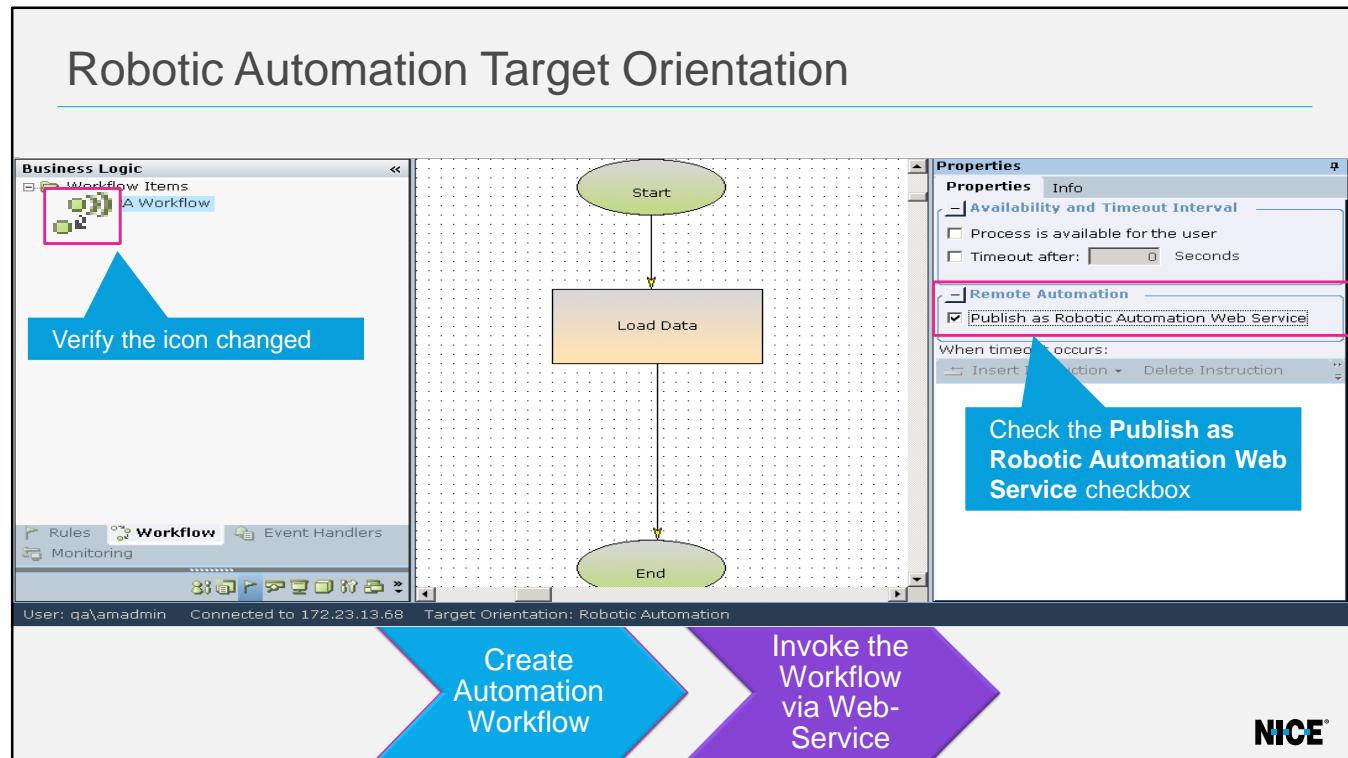
NICE®

- The process of implementing a Robotic Automation Solution involves creating and publishing a Robotic Automation Solution which contains at least one Robotic Automation Workflow, and then invoking this workflow using a Web-Service.

Robotic Automation Target Orientation



- Before we start creating a workflow, the solution must be defined to be published as a Robotic Automation solution so that it can be made available to the third-party applications, or classic RT Clients, that wish to invoke it.
- To specify that a project is to be published as a Robotic Automation solution, the Target Orientation needs to be set as Robotic Automation. The status bar should also update accordingly.



- In order for a solution to be recognized as a Robotic Automation solution, it must have at least one workflow defined as a Robotic Automation Web Service, which will also be made available to the third-party applications, or classic RT Clients, that wish to invoke it.
- Once the Project's Target Orientation was defined to be Robotic Automation, it is possible to specify a workflow to be Published as a Robotic Automation Web Service. Selecting this option will also cause the corresponding icon to appear in the navigation pane.

Robotic Automation Target Orientation

The screenshot shows two main windows. On the left is the 'Solution' window under the 'Info' tab. It displays the version as '001.000.000', the target orientation as 'Classic', and the target environment as 'Robotic Automation'. A blue callout points to the 'Publish as a Robotic Automation solution' button. On the right is the 'Administration' window, specifically the 'Solutions Assignment' section. It shows a tree view of assignments and a list of teams on the right. A blue callout points to the 'Assign to a team which contains at least one robot' section. Below the windows are three arrows: a blue one pointing right labeled 'Create Automation Workflow', a white one pointing right labeled 'Publish as a Robotic Automation solution', and a purple one pointing right labeled 'Invoke the Workflow via Web-Service'.

- After creating the workflow, the Robotic Automation Solution needs to be published and assigned to a team that contains at least one robot.



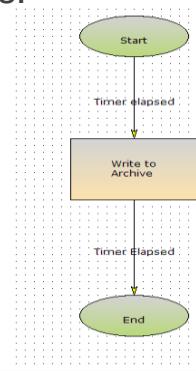
Notes from Demo

DO IT YOURSELF

Creating a Robotic Automation Workflow

- Create a Robotic Automation Workflow, named “Happy Birthday” with the following 2 parameters:
 - “Name” (Type: Text)
 - “Chosen Gift” (Type: Text)
- Create a timer to be used for the transitions
- Publish and assign the solution to the RtraTeam1 team

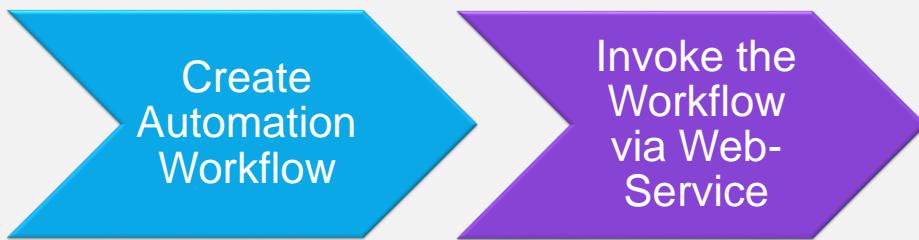
* Actions to be performed in the step will be added later in the lesson



NICE®

Invoke via Web-Service

- Invoker - The Robotic Automation Workflow can be invoked by any third party application able to execute Web Services
- Classic Client Invoker - The RT client can serve as the invoker

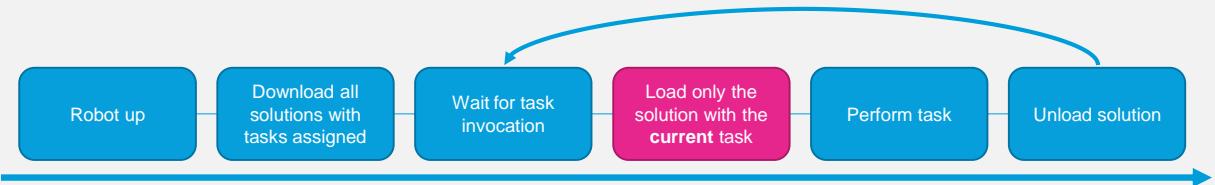


NICE®

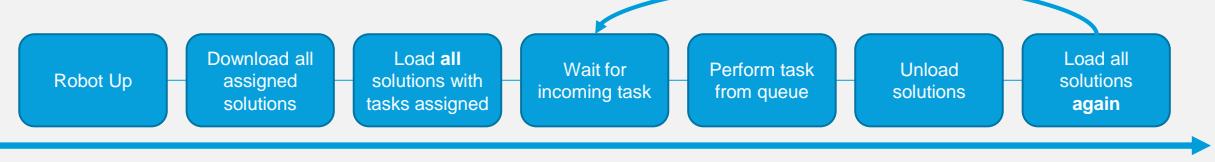
- Once the Robotic Automation Solution has been published to the RT Server, it can be invoked as a web service by any third party application able to do so, including a regular RT Client solution created in the RT Designer.
- In this lesson we will be focusing on using the RT Client as the Invoker.

How Do Robots Load Solutions?

Option 1 (Default) – Load on Demand



Option 2 – Load All solutions



NICE®

- Load on Demand: Robots are configured to load only the solutions needed for the task pulled from the queue, and then after the task is completed, unload these solutions

This method prevents collisions between solutions and can improve speed and performances especially with large projects.

- Load solutions: Robot download and load **all** the solutions with tasks assigned to it

Solution Load - Configuration

Configuration Management

ALERT
AUDIT
AUTH
CLIENT
COGNOS
DATABASE
DE
DEPLOY
DESKTOP ANALYTICS
INTERNAL
LDAP
LOGGING
QUARTZ
RA

Backlog Request expiration time from queue (sec):
21600

Backlogged Requests: buffer needed after Robotic Automation Invocation Timeout (sec):
600

Default robots' scheduled profile:
`<scheduleData xmlns:xsd="`

Enable Re Invoke API:
false

Enable gathering of invocation lifecycle. On error the invocation lifecycle is logged as well as the XML:
false

Enable logging of the gathered invocation lifecycle:
false

Enable to remove tasks in Tasks Control Room:
true

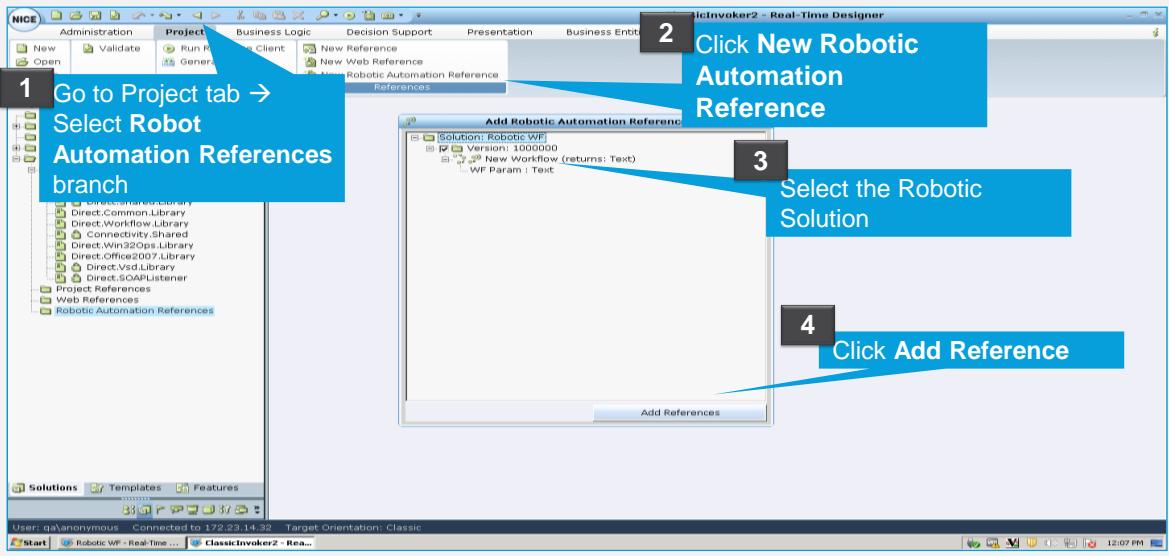
Load only solutions needed for the task pulled from the queue:
true

100000

NICE®

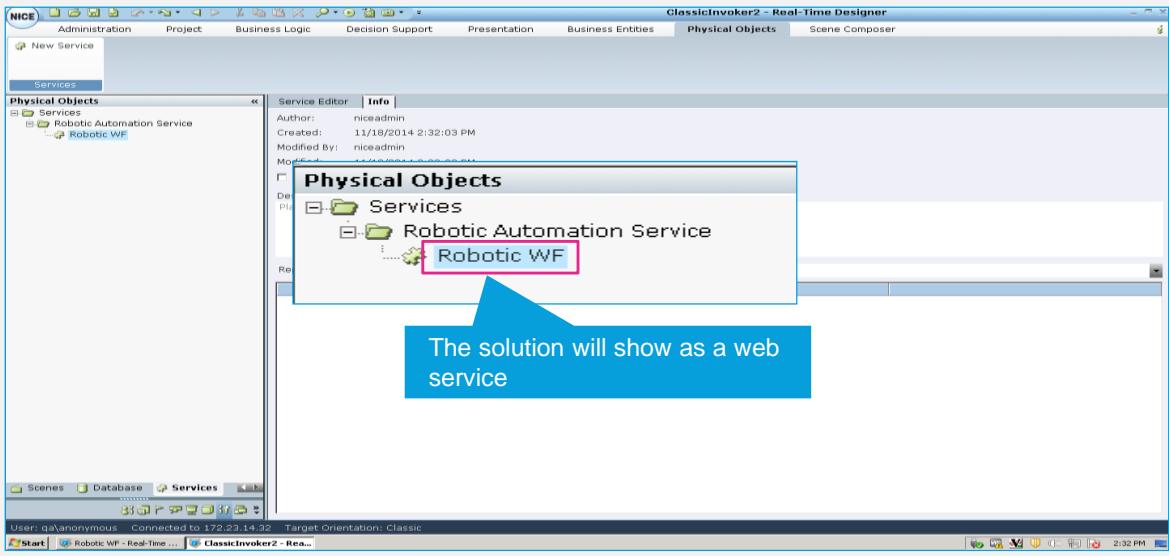
- The default setting is **Load on Demand: true** and can be configured in the server configuration management

How to Create a Classic Client Invoker



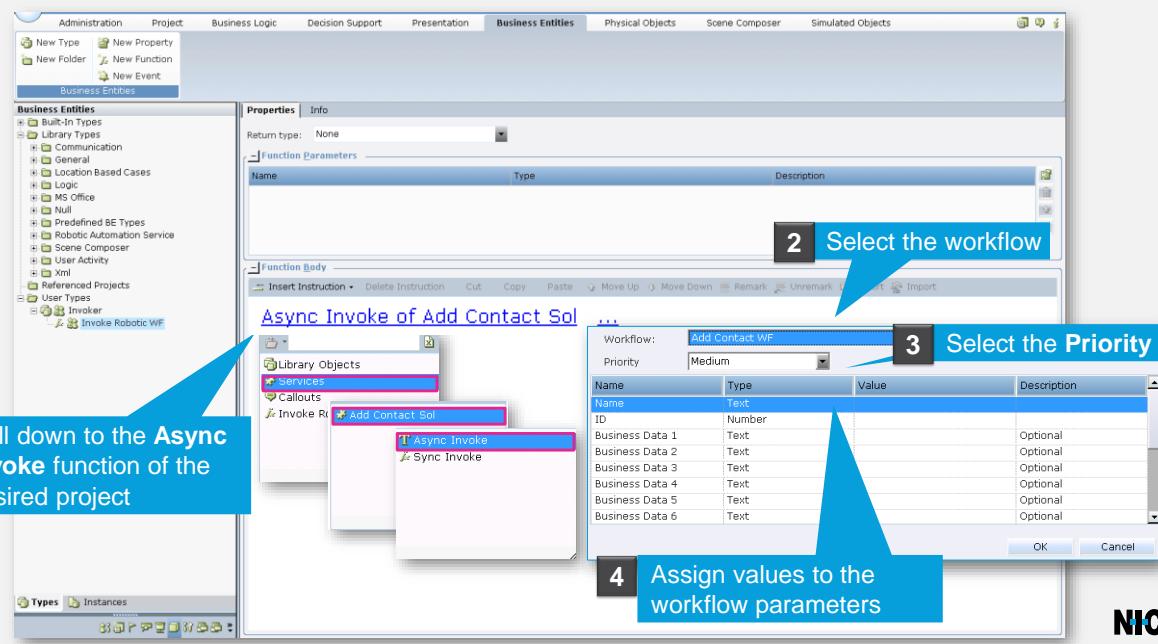
- A regular RT Solution acting as an Invoker of Robotic Automation solutions is referred to as a “Classic Client Invoker”.
- After clicking **New Robotic Automation Reference**, the RT Designer displays a list of all Robotic Automation solutions published to the RT Server, including their versions, the Robotic Automation Workflows they contain, and the parameters of those workflows.
- In case another version of the project was published, the reference may be updated by right clicking it and selecting the “Update Web Reference” option.

How to Create a Classic Client Invoker



- The Web-Service of the selected solution is added to the Services tab in the Designer, presenting all published Robotic Automation Workflows. The service's functionality is now available in the invoking project.

How to Create a Classic Client Invoker



- All robotics workflows that are a part of the referenced project can be selected.
- The parameters can be assigned business entities as well as typed in value.

Workflow Parameters in an Invoker

- In order for a workflow to be invoked, any parameters that are sent to it must not be empty.
- A workflow that receives an invocation with empty parameter(s) will not start and an error message will be added to the client log:

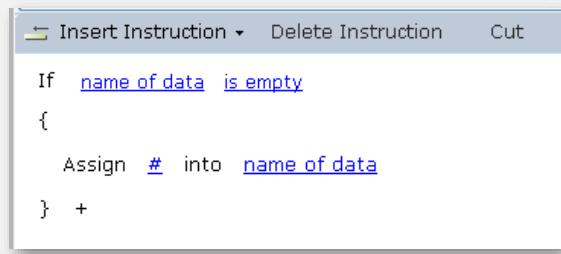
```
Roboticautomation - Automationservice.HandleAutomationRequest  
- Automation workflow is expecting 3 parameters, but 1  
parameters were passed in the Automation request
```

NICE®

- **Note-** by default, parameters that are Number or Decimal type are automatically initiated with '0' as a value and will not be empty.

Workflow Parameters in an Invoker

- A recommended way of action is to add a validation step to the invoker
- The validation will check if any of the parameters are empty, and if so, a predefined logic will assign a specific value (# in this example) instead of the empty parameters:



The screenshot shows a robotic workflow editor interface. At the top, there's a toolbar with 'Insert Instruction' (with a dropdown arrow), 'Delete Instruction', and 'Cut'. Below the toolbar is a code-like block editor. The code is as follows:

```
If name of data is empty
{
    Assign # into name of data
} +
```

NICE®

Workflow Parameters in an Invoker

- On the invoked workflow, a matching logic will address the parameters that were assigned with that value:

```
If name equals #
{
    Assign True into HaveEmptyParameters of My Received Data
}
Else
{
    Assign name into Received Name of My Received Data
}
```

- The logic could differ between parameters that are mandatory to the workflow and parameters that are not:

```
If Company equals #
{
    Assign False into HaveEmptyParameters of My Received Data
}
Else
{
    Assign Company into Received Company of My Received Data
}
```





Notes from Demo

DO IT YOURSELF

Creating an Invoker

- Create an Async Invoker to invoke the Robotic Automation Workflow:
 - Create a new Robotic Automation Reference
 - Create a callout that allows the agent to specify the customer's Name and chosen gift, to be passed on as the Robotic Automation Workflow parameters.
 - Assign the callout as the Quick Callout

NICE®

Third-Party Invokers

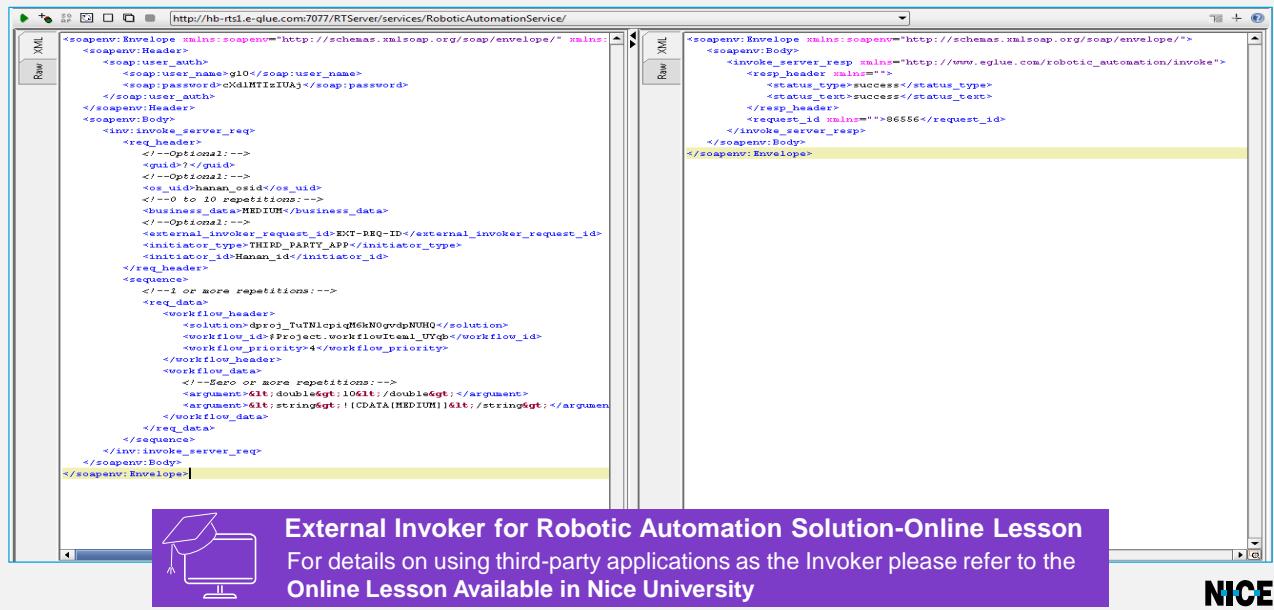
- **GetAllWorkflows** – Returns all the published robotic automation workflows

NICE®

- In order to invoke and run a specific Robotic Automation Workflow when using a third party application (similarly to the Classic Client Invoker) you will need the Solution and Workflow IDs. These can be found by running the **Get All Workflows** function that is part of the Web-Service.
 - For example, this is how **SoapUI**, an application used for invoking, developing, simulating and functional testing of web-services, produces the Solution ID and Workflow ID. This function will also inform you as to how many parameters (if any) the workflow will expect.
 - For additional information on how to use the Web-service in a third party application, please refer to the **Robotic Automation Solution Guide**.

Third-Party Invokers

Third-Party Invokers - Example



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="http://www.e-glue.com/robotic_automation/invoke">
<soapenv:Header>
<ns:auth>
<ns:user_name>g10</ns:user_name>
<ns:password>cXdlMTTzIUAj</ns:password>
</ns:auth>
</soapenv:Header>
<ns:invoke_server_req>
<ns:req_header>
<!--Optional:-->
<ns:guid>?</ns:guid>
<!--Optional:-->
<ns:task_header><ns:uid>10</ns:uid>
<!--0 to 10 repetitions:-->
<ns:business_data>MEDIUM</ns:business_data>
<!--Optional:-->
<ns:external_invoker_request_id>EXT-REQ-ID</ns:external_invoker_request_id>
<ns:initiator_type>THIRD_PARTY_APP</ns:initiator_type>
<ns:initiator_id>Hanuman_id</ns:initiator_id>
</ns:req_header>
<ns:sequence>
<!--1 or more repetitions:-->
<ns:req_data>
<ns:workflow_header>
<ns:solution>dproj_TuTNlcpiqH6kN0gvdpNUHQ</ns:solution>
<ns:workflow_id>Project.workflowItemsm1_UVyb</ns:workflow_id>
<ns:workflow_priority>4</ns:workflow_priority>
<ns:workflow_header>
<ns:label>
<!--Zero or more repetitions:-->
<ns:argument>double<gt;10</gt;</ns:argument>
<ns:argument>string<gt;!(CDATA[MEDIUM))</gt;</ns:argument>
</ns:workflow_header>
</ns:req_data>
</ns:sequence>
</ns:invoke_server_req>
</ns:Body>
</ns:Envelope>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<ns:invoke_server_resp xmlns="http://www.e-glue.com/robotic_automation/invoke">
<ns:rep_header value="">
<ns:status_type>success</ns:status_type>
<ns:status_text>success</ns:status_text>
</ns:rep_header>
<ns:request_id value="x86656">/request_id</ns:request_id>
</ns:invoke_server_resp>
</soapenv:Body>
</soapenv:Envelope>
```

External Invoker for Robotic Automation Solution-Online Lesson

For details on using third-party applications as the Invoker please refer to the
Online Lesson Available in Nice University





Notes from Demo

DO IT YOURSELF

Robotic Automation Exercise

- **Update the workflow:**

- The Workflow should concatenate the two parameters and insert the text into a text file (simulating the archive), which is located on the Designer machine.
- Each added line in the file should have the following structure:
The customer's Name is <Name> and s/he has chosen a <Chosen Gift>.
- The text should be added to the already existing text in the file.
 - Use the *Write text to file* library function. The path to be used in the function will be:
Hostname\Users\niceadmin.NICETRAINING.001\Desktop.
For the *Select Update* parameter select TRUE.

- **Publish and assign the solution to the RtraTeam1 team**

NICE®

ERROR HANDLING



Business Execution Status

- The Business Execution Status is a predefined Boolean which represent business logic error during execution of the workflow.



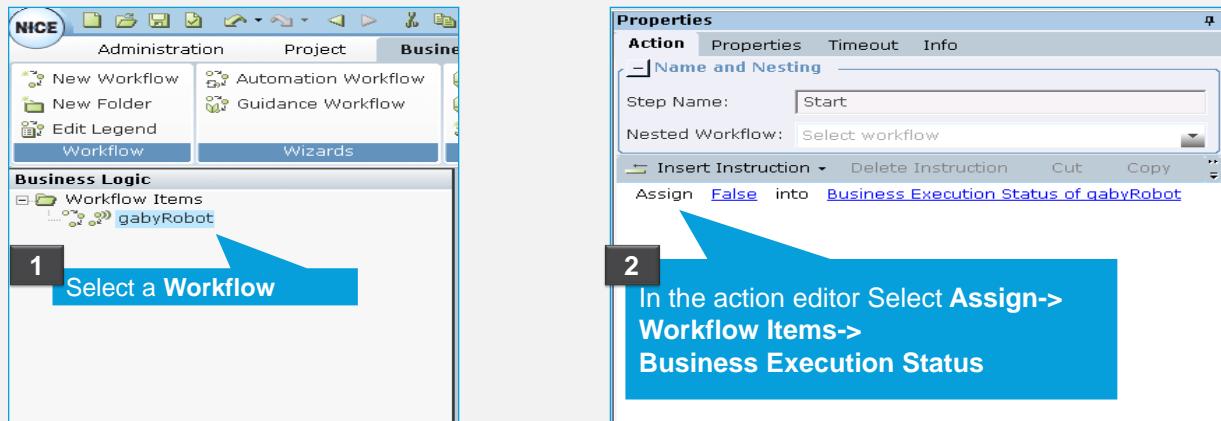
- This value will be saved in the database, and can be displayed in the Automation Portal.



NICE®

- The **Business Execution Status** is added by the implementer per workflow basis to address
 - situations of business problems (business logic rather than technical problem or design issues). For example, if the workflow is intended to transfer funds from one account to another, but the balance is insufficient, the workflow cannot be completed, but not because of a system or technical problem.
 - Additionally, this type of result needs to be monitored in the Automation Portal for tracking purposes.

Business Execution Status



- By default, the **Business Execution Status** value is set to **True** meaning **Execution Successful**.
- Best practice is to have this value changed in case a business error occurs and use the **Set Result** function to state the error reason.

Set Result Function

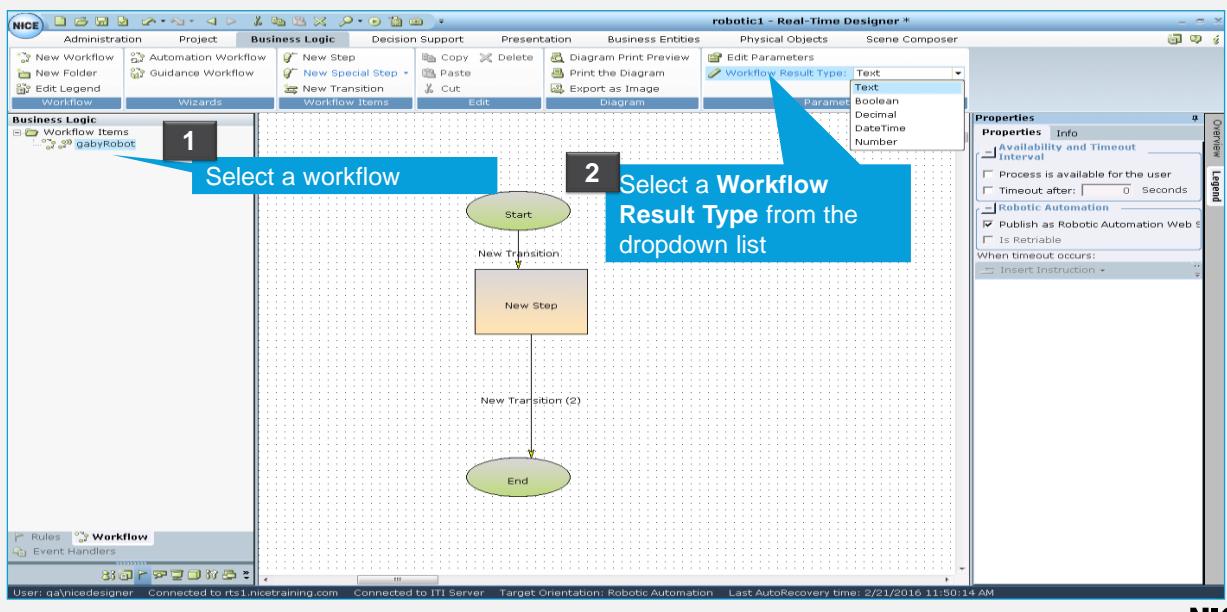
- The Set Result function enables the user to define a text string field that will be returned by the robot upon completion of the workflow
- For example this can be used to indicate an error had occurred



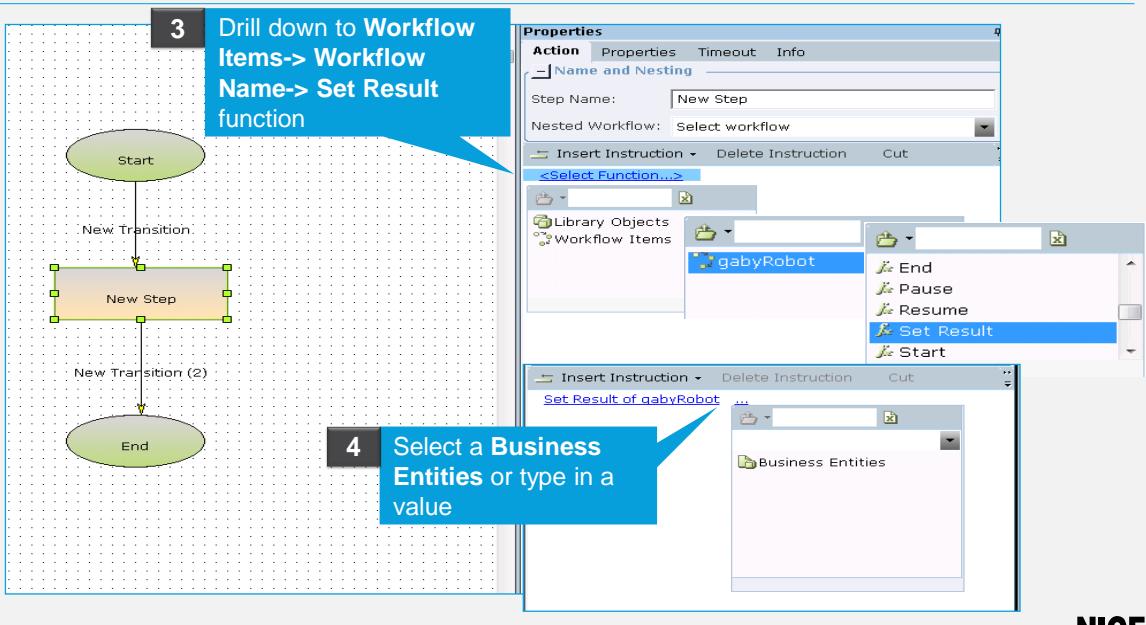
NICE®

- This string can be added by the implementer to provide information required by the project.
- For example, the user can set this field to display a text string that indicates that a workflow was completed successfully, or alert to a workflow failure in order to understand potential workflow problems.
- In addition, if a 3rd-party application used in the process displays an error message, the robot can capture the error message and attach it as the result.

Set Result Function



Set Result Function



Summary

- The concept of Robotic Automation
- Configuring Real-Time Solutions for Robotic Automation in the Real-Time Designer
- Creating a Robotic Automation Workflow
- Invoking a Robotic Automation Solution using a Web-Service

NICE®



Thank You





ROBOTIC AUTOMATION PART 2



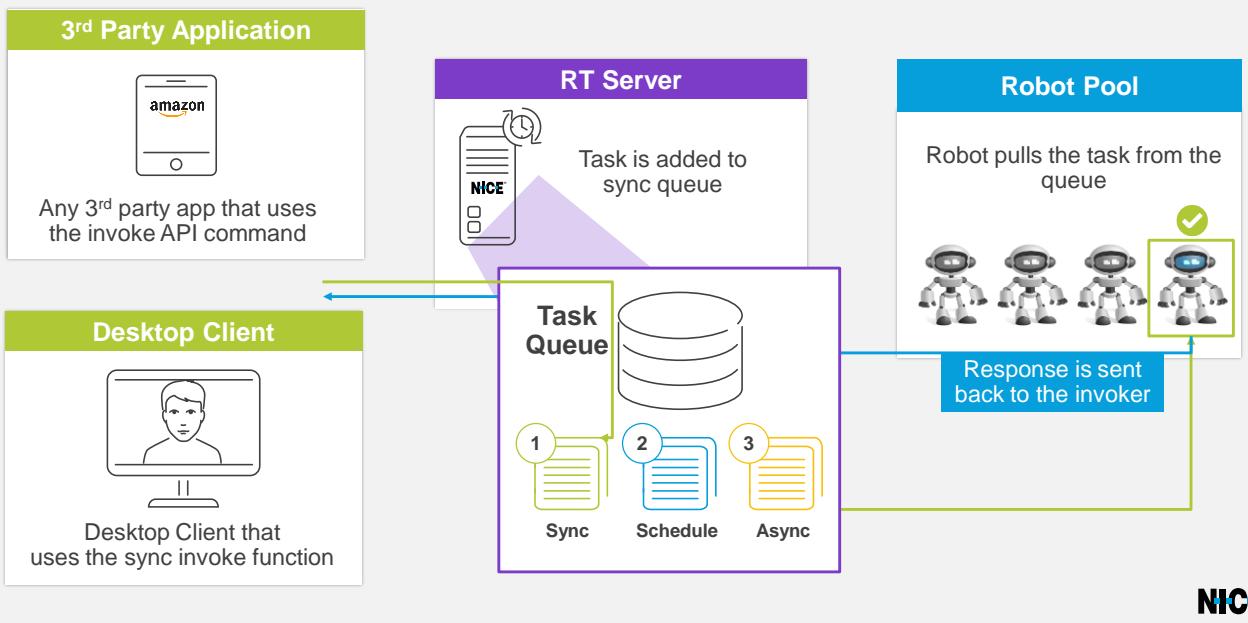
Lesson Objectives

By the end of this lesson you will be able to:

- Describe the concept of Sync Robotic Automation
- Invoke a Sync Conversation
- Schedule Robots
- Describe the concept of task queues and priorities

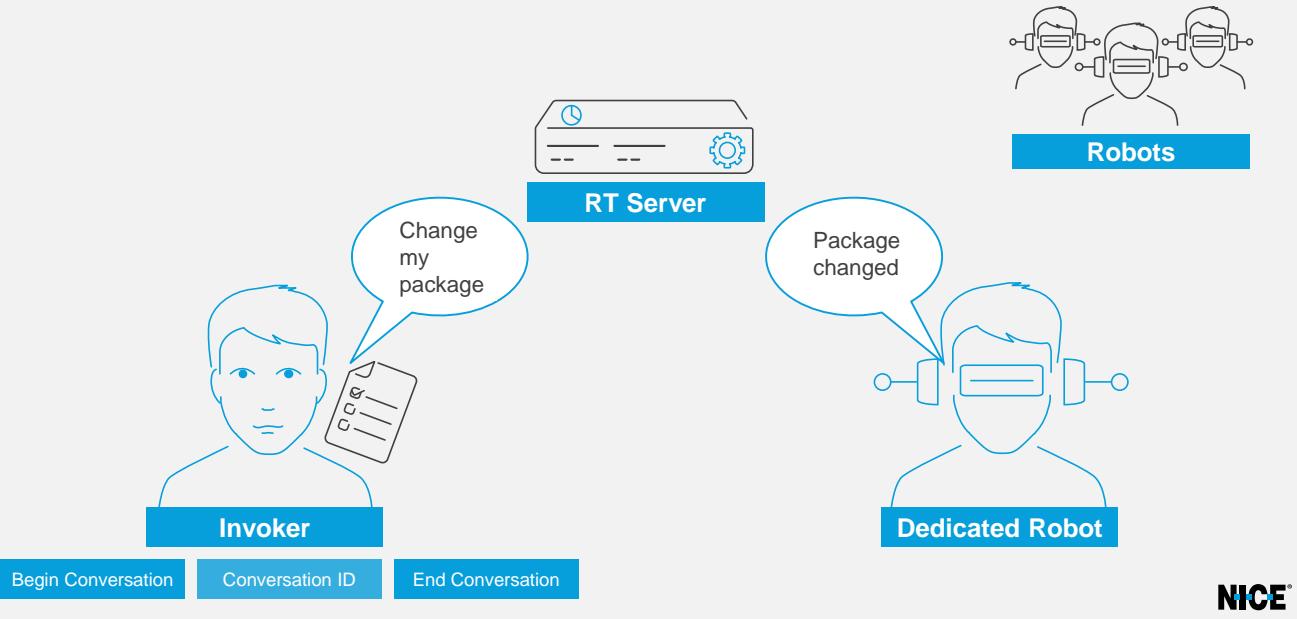


Sync Automation Flow



1. An invoker (a third-party application that uses the invoke API command or the Desktop Client that uses the sync invoke function) sends a request to begin a conversation with the Real-Time Server.
2. The Real-Time Server creates a conversation ID.
3. The invoker sends the first task. The Real-Time Server assigns a dedicated robot to the conversation, and adds the task to the sync task queue. The robot is assigned for the lifecycle of the conversation (a conversation can include several automation requests).
4. The assigned robot executes the task, sends a reply to the invoker and then waits for the invoker's next task or for the termination of the conversation.
5. Each time the Robotic Automation client processes a task, it returns a response to the Real-Time Server, and the server saves the results to the database for reports and analysis.
6. Additional tasks are sent (one at a time).
7. The invoker sends a request to end the conversation.

Real Time Robot – Sync Conversation



- When the initial task is sent to the server, the server creates a conversation and assigns a dedicated robot.
- This robot then executes this initial task and responds to the invoker.
- The conversation continues, with each of the tasks executed by the same robot. The robot sends a reply to the invoker after each task.
- This continues until the conversation ends (a conversation ends when the invoker sends an end conversation call or the conversation times out).
- **NOTE:** A conversation can last up to two hours after which it will be terminated.
- If a synced invocation is orphaned, backlogged or removed, the conversation is closed by the microservice

Enabling Sync Invocation (Conversation Mode)

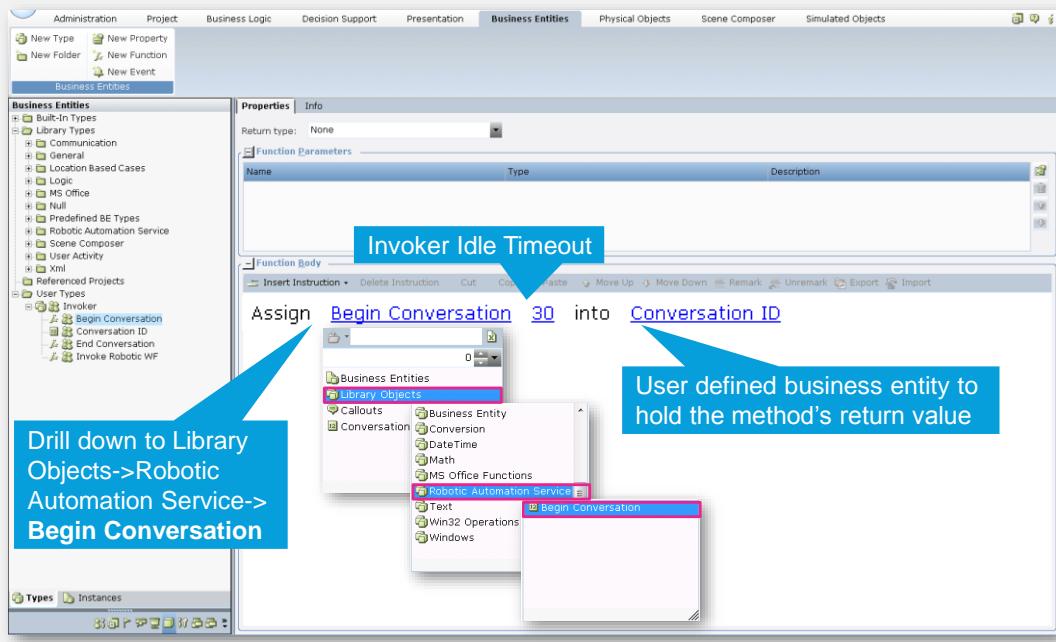
- To enable sync invocations (conversation mode), you must edit the **RTCClient.exe.config** file on each **robot**.

```
327 <realTimeRemoteAutomationConfig keepAliveInterval="10000" checkWorkingHoursInterval="300"
connectionLostRetries="10" connectionLostRetryInterval="60"
conversationTimeoutInterval="600000" roboticSchedulerUrl="
https://KS0607141705.e-glue.com:1912/RTServer/services/RARobotManagementService"
websocketHost="KS0607141705.e-glue.com" websocketPort="1912" websocketConnectionString="
wss://{{activeMQHostName}}:{activeMQPort}/RTServer/rest/nice/rti/ws/robotic"
clientHeartBeat="10000" serverHeartBeat="10000" messageTimeout="300000"
enableConversationMode="true" nextInboundQueueTimerTimeout="2000" heartBeatBuffer="10000" />
```

NICE®

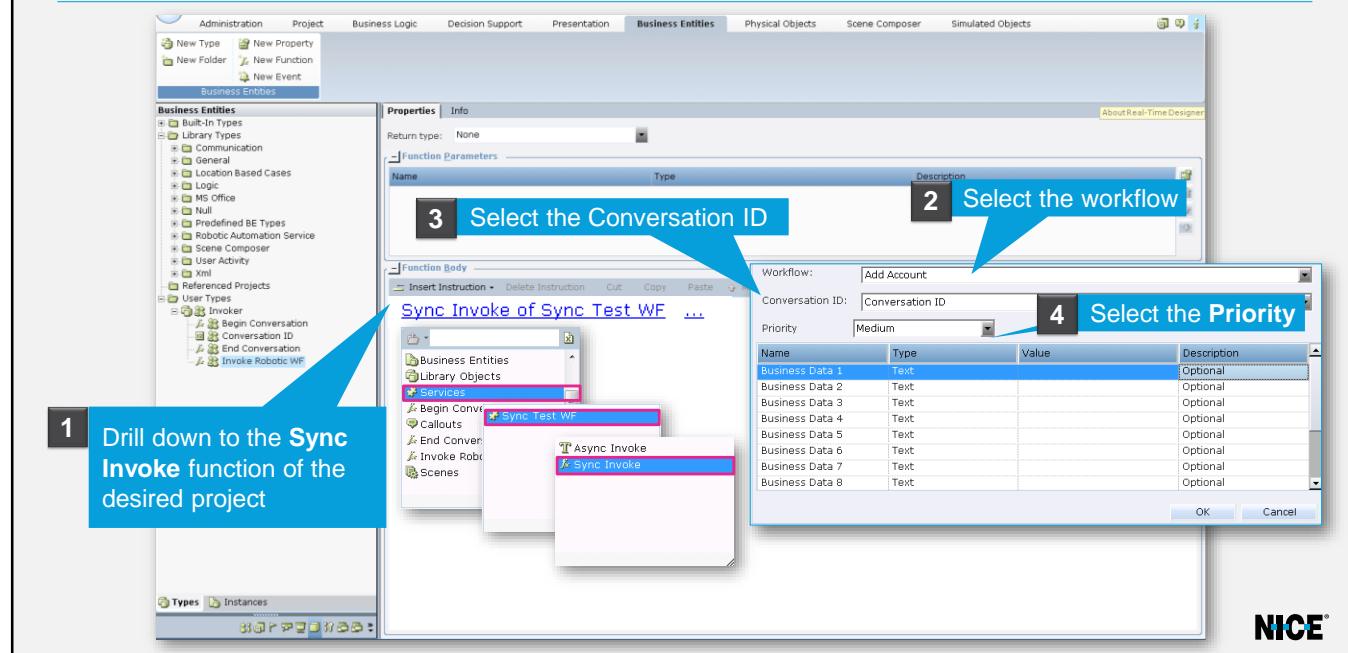
- When it set on false the robot will not check the sync queue
- This configuration is only relevant for robot client config file

How to Create a Sync Invocation - Begin Conversation



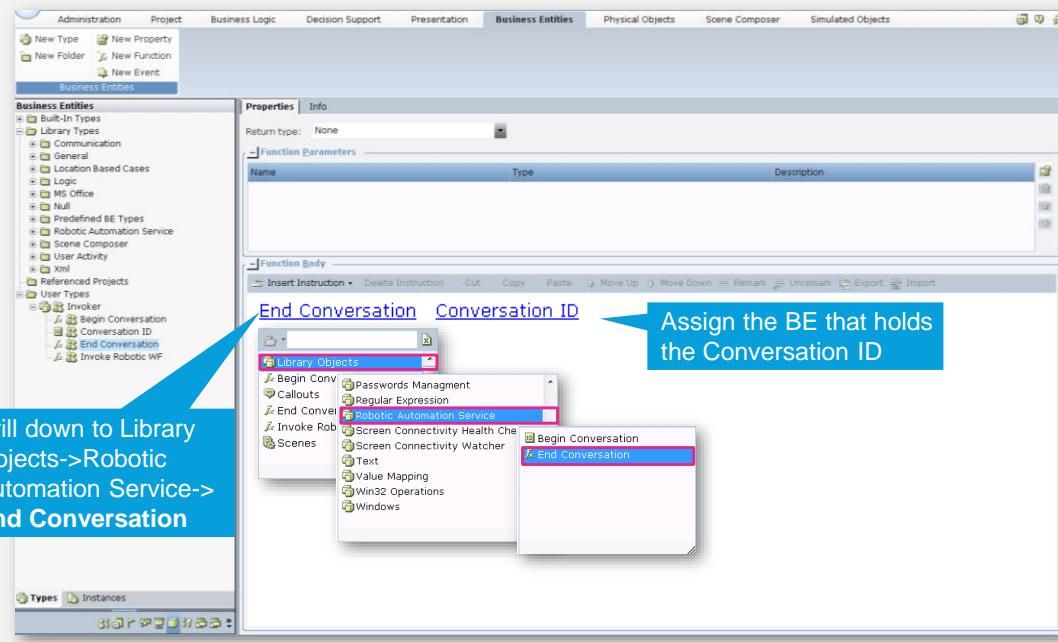
- Use the **Begin Conversation** method to send a request to begin a conversation with the Real Time Server.
- You must specify the invoker idle timeout (this timeout checks the idle time between the begin conversation response and the first invocation).
- It then checks the time between the first invocation response and the second invocation, and so on).
- The Real-Time Server creates a conversation ID.

How to Create a Sync Invocation - Invoke Sync

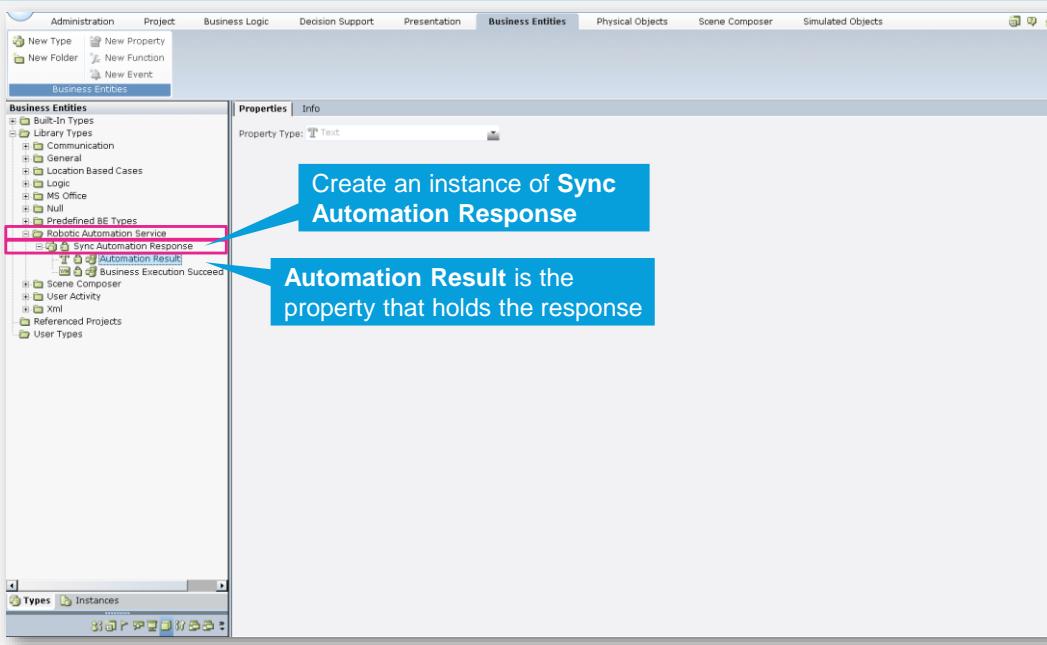


- The Priority: This only impacts the first invocation inside a conversation, as once the first invocation is executed, a dedicated robot services the conversation.

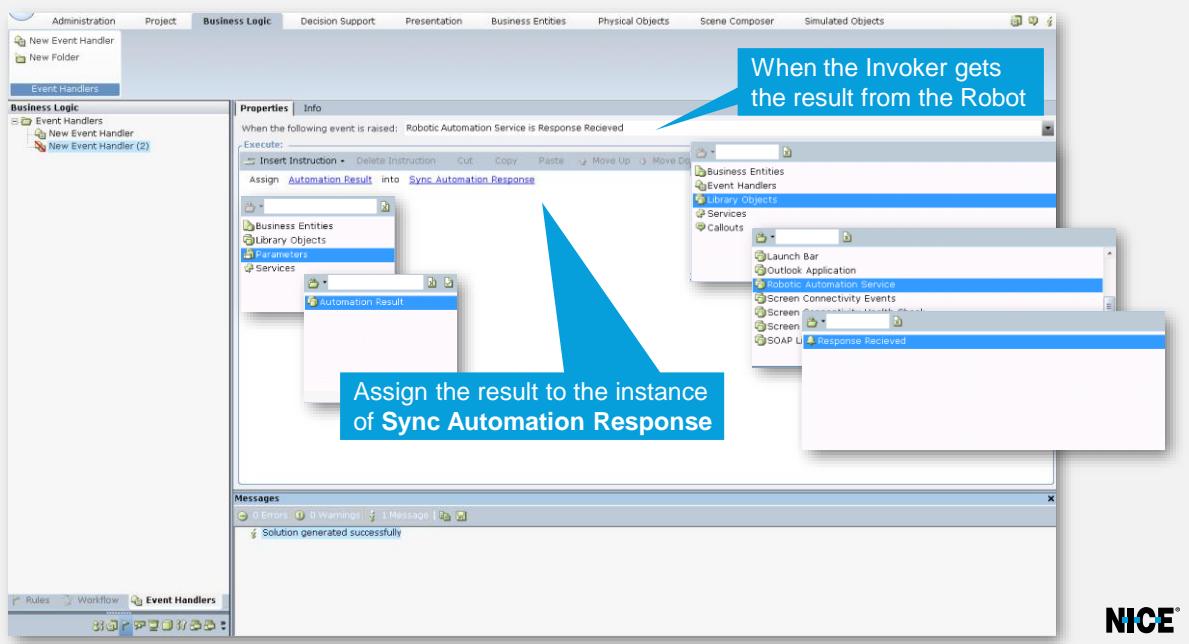
How to Create a Sync Invocation - End Conversation



Get Robot Response



Get Robot Response Example



NICE®

LIVE DEMO

Sync Automation Exercise

• Update the Robotic Workflow

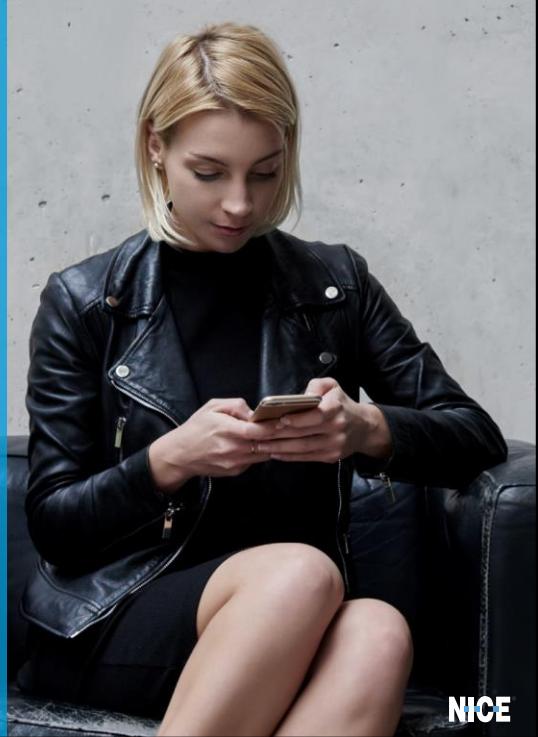
- Add logic to check if the gift that the customer chose contains space(" ") and set the business execution accordingly
- If the gift contains space, set the result to: The gift is Invalid
- Else, set the result to: The gift is Available
- Republish and assign the solution to the RtraTeam1 team

• Update the invoker

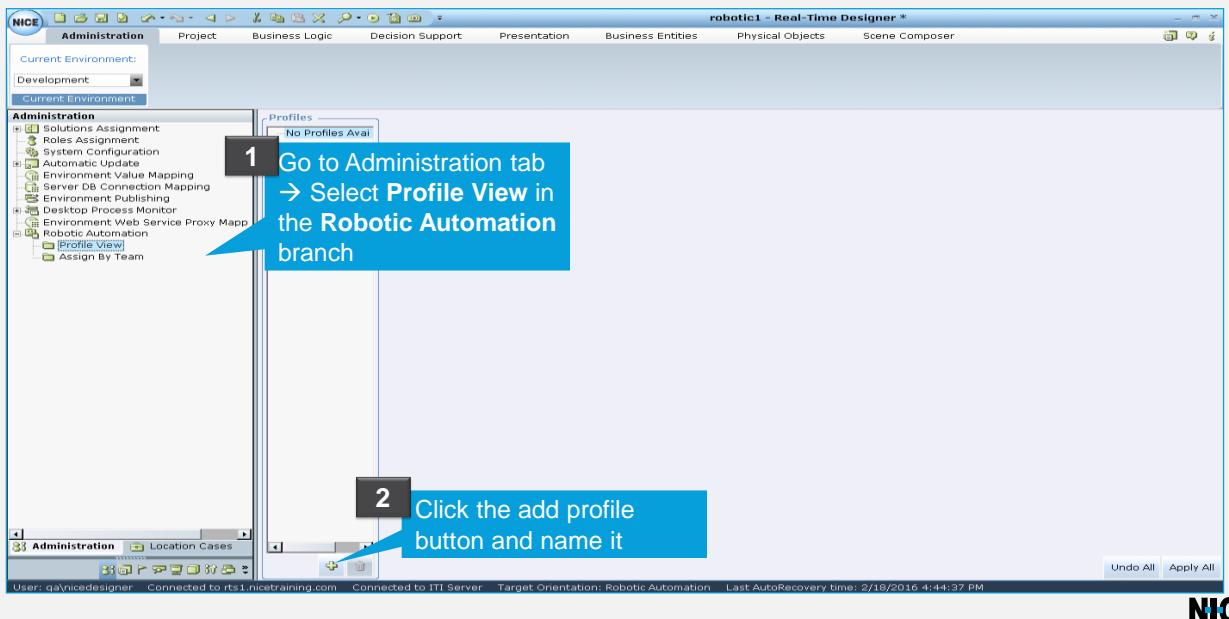
- Update the robotic reference to the latest published version
- Create an instance of Sync Automation Response and present it in the callout
- Add a Sync invocation
- Test your solution

NICE®

SCHEDULING ROBOTS

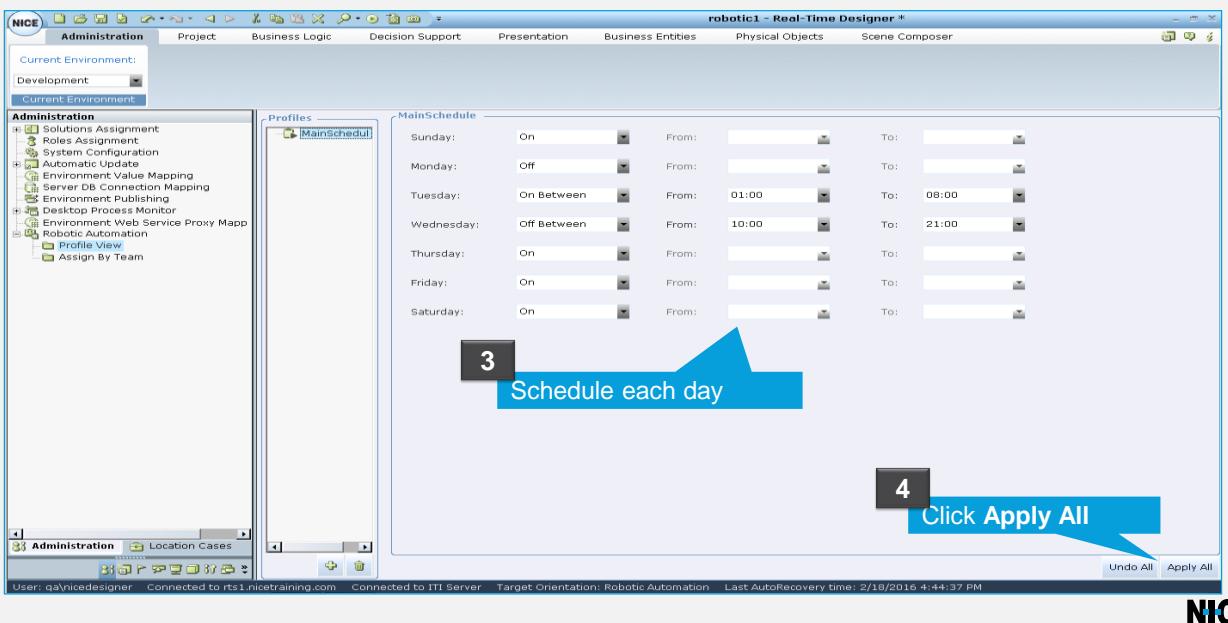


Robot Service Schedule



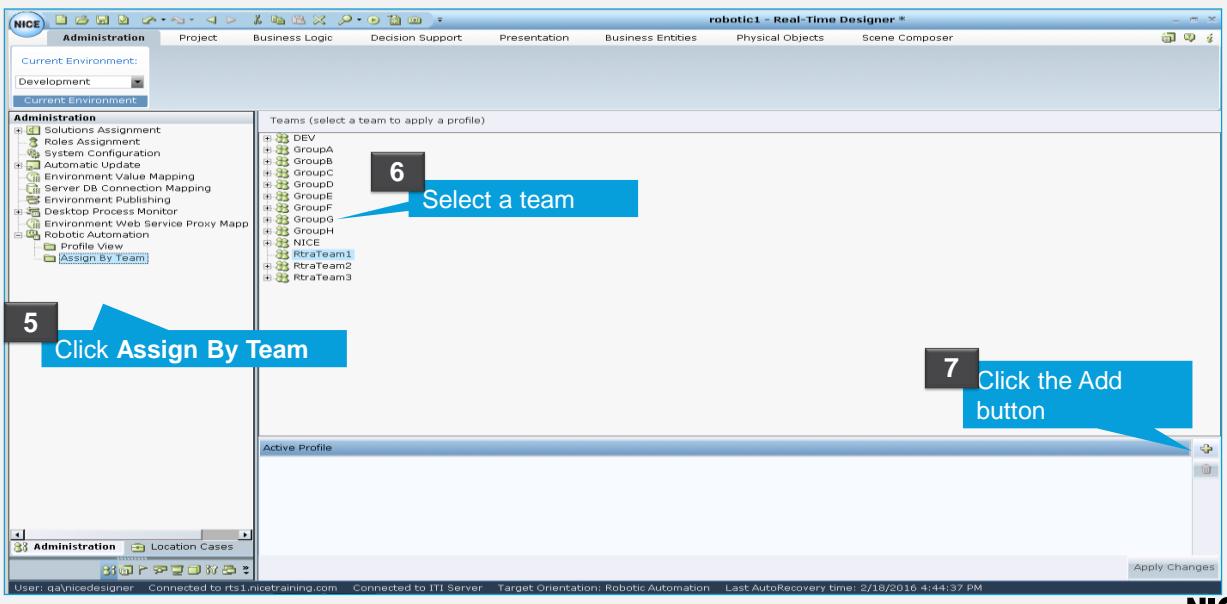
- It is possible to create a schedule profile and assign it to a team that contain robots

Robot Service Schedule

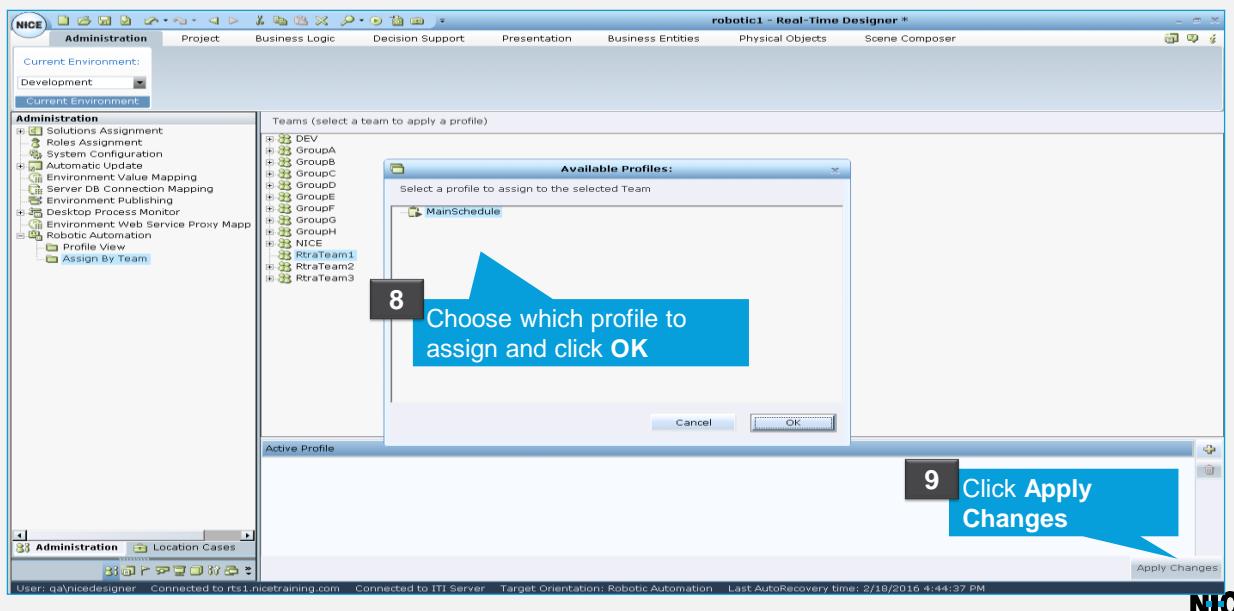


- There are 4 scheduling options:
 - On – the robots will be active all day.
 - On Between – when selecting this option it is require to choose a time frame in which the robots will be active.
 - Off – the robots will be inactive for the entire day.
 - Off Between - when selecting this option it is require to choose a time frame in which the robots will be inactive.
-
- Clicking **Undo All** will undo the last changes you made.

Robot Service Schedule



Robot Service Schedule

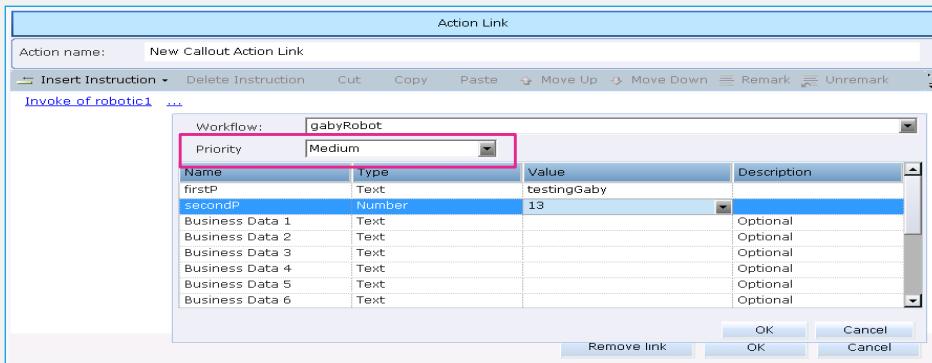


PRIORITIES



Task Priority

- In the Designer, when the Invoke function is selected, the 'Priority' parameter can be changed to High or Low (Default is Medium)
- If a 3rd party invoker is used, this parameter will be Medium unless defined otherwise

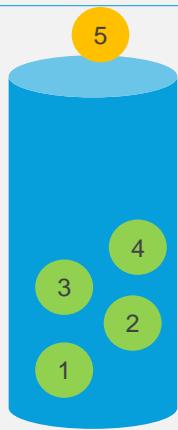


NICE®

Task Priorities

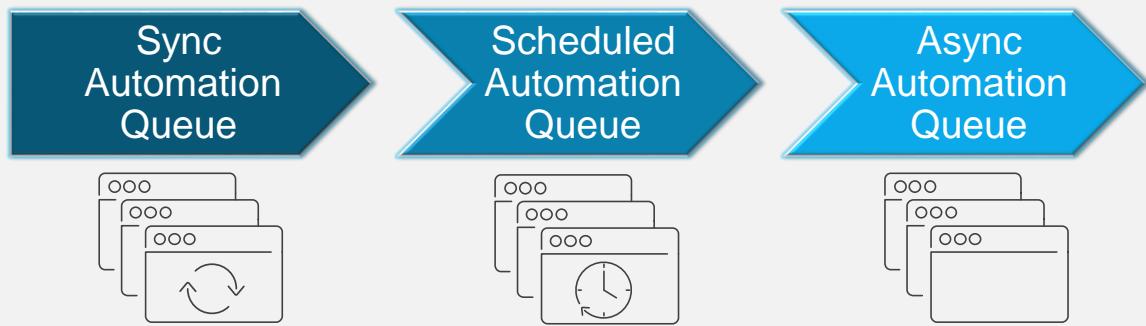
**NOTE**

Be aware not to over prioritize your solution

**NICE**

Automation Queues Architecture

- There are three automation queues:

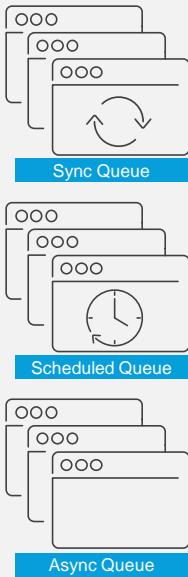


NICE®

- **Sync Automation Queue** is used for Real-Time robots. The process begins when a request to initiate a conversation is sent by another client or 3rd party application. Then, when the first invocation will enter the queue a dedicated robot will be assigned to serve the conversation until it ends. A conversation can include multiple invocations for different tasks.
- **Scheduled Automation Queue** is used for scheduled invocations which were created in the Automation Portal. This is used for periodic tasks like maintenance tasks.
- **Async Automation Queue** is used for tasks which were invoked by a 3rd party application or a classic invoker (another attended Client).
- As opposed to the sync queue, those tasks are processed asynchronously by any available robot.

Automation Queues Architecture

- Priorities:



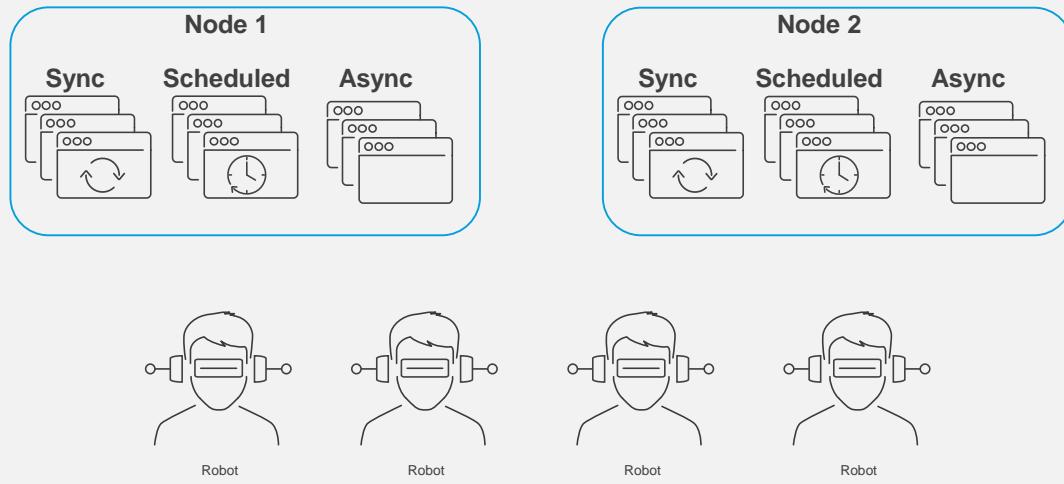
- First invocation priority + FIFO
- Time range + FIFO
- Priorities + FIFO

NICE®

- The unattended client (Robot) checks the three queues in the following order: sync, scheduled and then async.
- The robot filters the tasks based on the solutions assigned to its team. For example, if solution A is assigned to the robot's team, and there is a task from solution A in the sync automation queue, it pulls the task. If not, it proceeds to check the scheduled automation queue, and so on.
- **In the Sync automation queue**, priorities can be set for the first invocation inside a conversation. This impacts the only the first invocation, as once the invocation is executed, a dedicated robot services the conversation. High priority tasks are processed first, followed by medium priority tasks and finally low priority tasks. Tasks in the sync automation queue with the same priority are processed on a first in, first out basis.
- **In the Scheduled automation queue** tasks are processed according to the time range which was set to them. Tasks with the same range are processed on a first in, first out basis.
- **In the Async automation queue** tasks have priorities assigned to them by the invoker, and are processed accordingly. High priority tasks are processed first, followed by medium priority tasks and finally low priority tasks.

Automation Queues Architecture

- Cluster environment



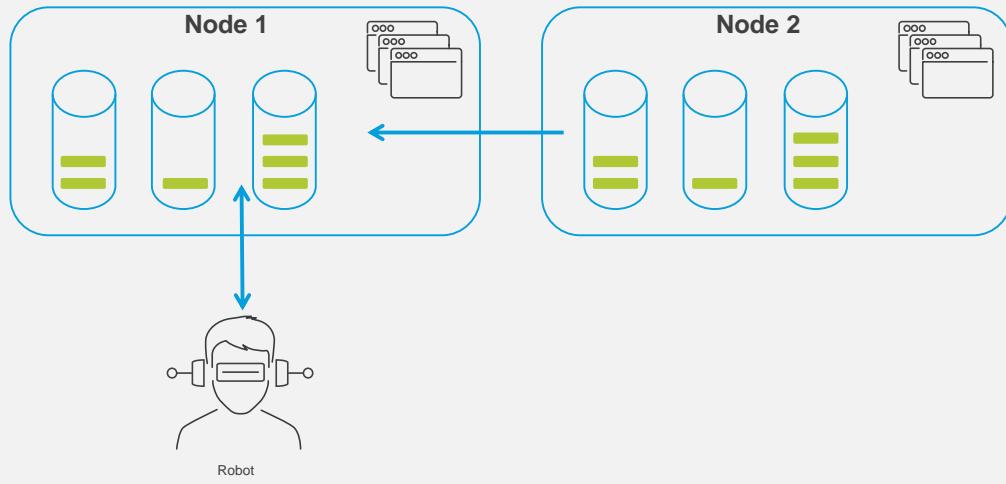
NICE®

- In a cluster environment, each node has all three automation queues. The robot processes the tasks in the queues on its node first so tasks are processed in the following order:

1. Sync automation queue tasks (giving priority to conversations where the first invocation has a high priority, then medium priority and finally low priority)
2. Scheduled automation queue tasks
3. High priority async queue tasks
4. Medium priority async queue tasks
5. Low priority async queue tasks

Automation Queues Architecture

- Cluster environment



NICE®

- If there are no tasks available to a robot in its node, tasks from other node will be distributed across the nodes and the robot will start to process these tasks. A robot completes all the tasks (at all priority levels) in the async automation queue on their node, before taking tasks from a different node. As a result, in some cases low priority tasks on one node will be processed before high priority tasks on a different node.
- The robotic client signals node 1 its availability. Since node 1 queues are empty, this triggers the sharing process, in which tasks are transferred from node 2 to node 1. Those tasks are executed by the robot according to the same algorithm discussed earlier.
- The process is handled by the ActiveMQ and the spread of the task is determine by its algorithm.
- **NOTE:** The master node is responsible for the network of brokers and must be up to use shared queues. If the master fails, each node will process their own tasks only.

Summary

In this lesson we covered:

- The concept of Sync Robotic Automation
- Invoking a Sync Conversation
- Scheduling Robots
- The concept of task queues and priorities

NICE®



Thank You



NICE[®]

AUTOMATION PORTAL



Lesson Objectives

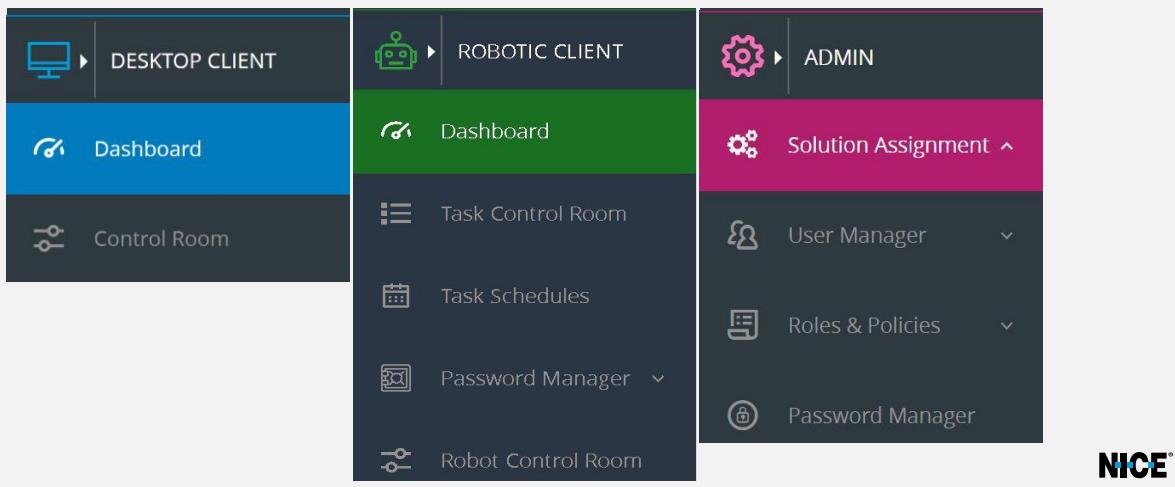
By the end of this lesson you will be able to:

- Use the Desktop Client Control Room and Dashboard
- Use the Robotic Client Control Rooms and Dashboard
- Schedule a task for Automation
- Manage Passwords
- Use the Admin Functions



Automation Portal

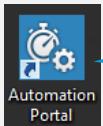
- The Automation Portal is an easy way to monitor and control your APA system. The Portal is divided into three modules:



- Desktop Client - You can see a Dashboard of all client statuses or drill down to the Control Room where you can receive more details and perform some actions.
- Robotic Client - If you have a license for Robot Automation you use the Robotic Client to view and manage your robots and your scheduled and non-scheduled tasks.
- Admin - Here you can perform administrative tasks such as assigning roles which can limit each user's access to their own area.

Using the Automation Portal

- To access the Automation Portal click on the icon:



Verify that the URL for the shortcut is in the following form:

<https://<Server FQDN>:1912/webapps/rtclients>

The Automation Portal only works in a secured environment (HTTPS)

- Login page:



Best viewing experience

NICE®

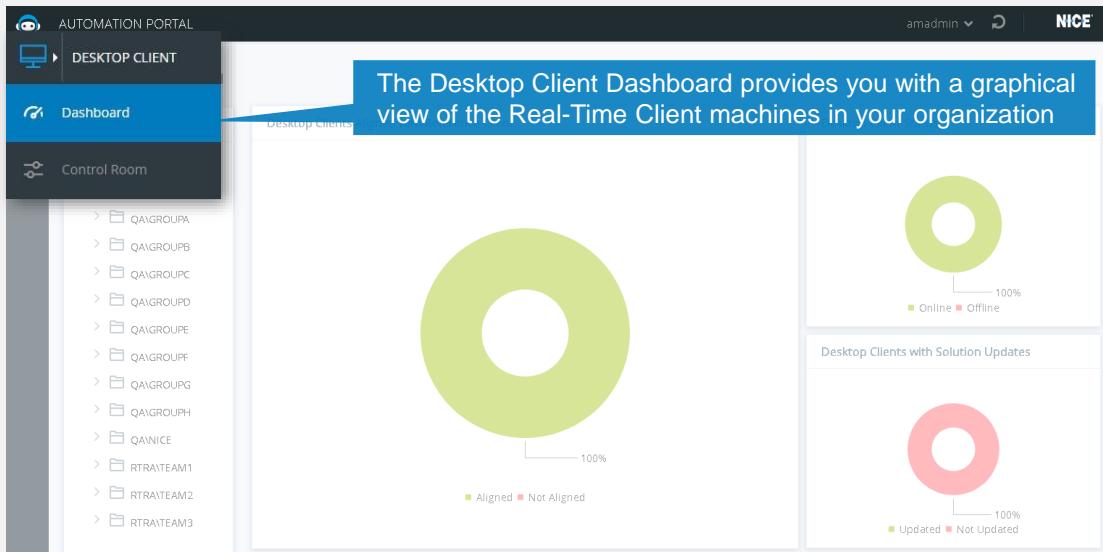
- Automation Portal is supported only on the following web browsers:
- Google Chrome Microsoft
- Internet Explorer 11 and higher
- Minimum screen resolution is 1366 x 768 px.
- Automation Portal only works in a secured environment (HTTPS).
- NOTE: Verify that the URL for the shortcut is in the form:
https://FQDN:1912

Desktop Client



NICE®

Desktop Client Dashboard



Desktop Client - Dashboard

The screenshot shows the 'Desktop Client Dashboard' within the 'AUTOMATION PORTAL'. The left sidebar displays the 'Organizational Hierarchy' with 'All Teams' expanded, showing sub-teams like QADEV, QAIGROUPA, QAIGROUPB, QAIGROUPC, QAIGROUPD, and QAIGROUPE. Below this, 'RTRATEAM2' and 'RTRATEAM3' are listed under 'All Teams'. A callout box points to this sidebar with the text: 'View the client file deviations (aligned/not aligned with the reference client) for the selected team and its entire sub hierarchy'.

The main dashboard area contains three donut charts:

- Desktop Clients Aligned with Reference Client:** A large green donut chart representing 100% alignment. A callout box points to it with the text: 'Click any pie chart to open the Control Room with the relevant filters applied'.
- Desktop Client Status:** A green donut chart representing 100% online clients. The legend indicates a green square for 'Online' and a red square for 'Offline'.
- Desktop Clients with Solution Updates:** A red donut chart representing 100% clients not updated. The legend indicates a green square for 'Updated' and a red square for 'Not Updated'.

A callout box points to the bottom chart with the text: 'Clients which have been assigned a solution but have not yet been loaded/ clients that didn't yet load the latest assigned solution'.

The top right corner of the dashboard features the 'NICE' logo.

Desktop Client Control Room

The screenshot shows the 'Control Room' section of the 'Desktop Client' interface. On the left, there's a sidebar with filters for 'Pending' (1), 'No Pending' (2), 'REFERENCE CLIENT' (3), 'Not Aligned' (0), 'Aligned' (0), 'TEAMS' (0), 'Not Specified' (0), 'Add / Remove' button, 'LOADED SOLUTIONS' (0), 'Not Specified' (0), 'Add / Remove' button, and 'ASSIGNED SOLUTIONS' (0), 'Not Specified' (0), 'Add / Remove' button. The main area displays three clients: 'Administrator' (DANDC631B, qalg10, 14 days) and 'Administrator' (DANDC631B, qalg16, 5 days). A prominent blue banner in the center states: 'The Control Room provides you with a view of the Real-Time Clients in your organization'. The top right corner shows 'Last Refresh: 12/21/2017 11:17 AM' and a search bar. The bottom right corner features the 'NICE' logo.

SOLUTION VERSION	STATE	ACTIONS
DANDC631B	qalg10	[Edit] [Delete] [Info] [More]
DANDC631B	qalg16	[Edit] [Delete] [Info] [More]

Desktop Client - Control Room

The screenshot shows the 'Desktop Client Control Room' section of the NICE Automation Portal. At the top, there's a placeholder for a 'Reference Client' featuring a stylized head with a clock icon. Below this is a search bar and a table titled 'Showing 1 - 3 of 3 Clients'. The table includes columns for OS LOGIN, HOST NAME, USER NAME, TIME IN STATE, SOLUTION VERSION, and STATE. The first row, which is highlighted with a pink border, represents the 'Reference Client' (OS LOGIN: niceadmin, HOST NAME: RTC1, USER NAME: qa\glo, TIME IN STATE: 2 days, SOLUTION VERSION: Test CCA 001.000.000, STATE: Online). The second and third rows represent other clients (HOST NAME: RTC1, USER NAME: qa\niceagent, TIME IN STATE: a few seconds, SOLUTION VERSION: Test CCA 001.000.000, STATE: Online; and HOST NAME: RTD1, USER NAME: qa\nicedesigner, TIME IN STATE: 2 hours, SOLUTION VERSION: N/A, STATE: Offline). Below the table, there are nine icons of stylized human heads, each with a blue clock icon, representing Real-Time (RT) clients. The icons are labeled 'RT Client' underneath them. The NICE logo is visible in the bottom right corner.

Showing 1 - 3 of 3 Clients					
OS LOGIN	HOST NAME	USER NAME	TIME IN STATE	SOLUTION VERSION	STATE
niceadmin	RTC1	qa\glo	2 days	Test CCA 001.000.000	Online
niceadmin	RTC1	qa\niceagent	a few seconds	Test CCA 001.000.000	Online
niceadmin	RTD1	qa\nicedesigner	2 hours	N/A	Offline

- It is considered best practice to set a reference client
- The reference client is the client used as a standard for all other clients. It is generally a client that is installed on the administrator's machine, and is updated with all latest versions and .dll files. The other clients are then compared to this reference client.
- The reference client appears pinned at the top of the list of clients. For each client (other than the reference client), the following installed files are compared with the versions of the reference client files:
 - **Program Files:** *.exe and *.dll in the directory where the Real-Time Client is installed, typically, C:\Program Files\NICE Systems\Real-Time.
 - **Config Files:** *.config, *.cfg and *.xml files in the Application Data directory.

Desktop Client - Control Room

The screenshot shows the 'Desktop Client Control Room' section of the Automation Portal. On the left, there are filters for 'DESKTOP CLIENT STATE' (Online: 1, Offline: 2), 'SOLUTION UPDATES' (Pending: 3, No Pending: 0), 'DESKTOP CLIENT REFERENCE' (Aligned: 3, Not Aligned: 0), 'TEAMS' (Not Specified), 'LOADED SOLUTIONS' (Not Specified), and 'ASSIGNED SOLUTIONS' (Not Specified). The main table displays three clients:

OS LOGIN	HOST NAME	USER NAME	TIME IN STATE	SOLUT...	SOLUTION VERSION	STATE
niceadmin	RTC1	qa\g0	2 days			Green
niceadmin	RTD1	qa\nicedesigner	2 hours			Green
niceadmin	RTC1	qa\niceagent	a minute	Test CCA	001.000.000	Green

A blue callout box with the text 'Click the menu button and set a Client to be the reference version or to remove it' points to a context menu for the third client (RTC1). The menu items are: Set as Reference (highlighted with a red box), Request Restart, Force Restart, and Delete.

NICE®

Desktop Client - Control Room

The screenshot shows the 'Desktop Client Control Room' page. On the left, there are several filter sections: 'DESKTOP CLIENT STATE' (Online: 1, Offline: 2), 'SOLUTION UPDATES' (Pending: 3, No Pending: 0), 'DESKTOP CLIENT REFERENCE' (Aligned: 2, Not Aligned: 1), 'TEAMS' (Not Specified), 'LOADED SOLUTIONS' (Not Specified), and 'ASSIGNED SOLUTIONS' (Not Specified). Below these filters is a pink 'Filters' button. To the right of the filters is a table titled 'Showing 1 - 3 of 3 Clients' with columns: OS LOGIN, HOST NAME, USER NAME, TIME IN STATE, SOLUTION, SOLUTION VERSION, and STATE. The table contains four rows of client data. At the bottom of the table is a pink 'Client properties and actions' button.

OS LOGIN	HOST NAME	USER NAME	TIME IN STATE	SOLUTION	SOLUTION VERSION	STATE
niceadmin	RTC1	qa\g0	2 days			
niceadmin	RTC1	qa\niceagent	a few seconds	Test COX	001.000.000	
niceadmin	RTC1	qa\g0	2 days			
niceadmin	RTD1	qa\nicedesigner	2 hours			

NICE®

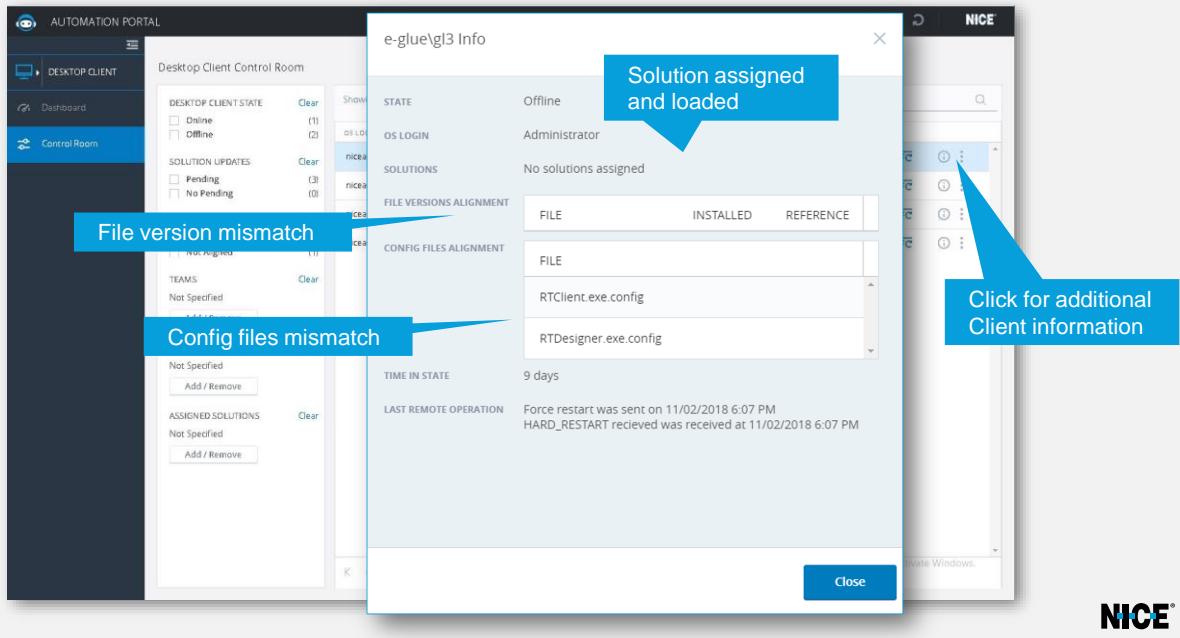
- You can filter Clients upon any of the following:
 - Client state – online or offline
 - Clients with solution updates – solution has been assigned but not yet downloaded
 - Clients aligned with reference client – client dll and config files match the reference client
 - Teams
 - Loaded Solutions – solution versions loaded by the clients
 - Assigned Solutions - solution versions assigned to the clients
- Client Properties:
 - OS Login - The login details for the machine on which the client is installed.
 - Host Name - The machine name on which the client is installed.
 - User Name - The Real-Time client user name.
 - Time in State - The length of time the client has been in its current state.
 - Solution Name - The name of the solution installed on the client machine.
 - Solution Version - The version of the solution installed on the client machine.
 - State - Indicates whether the client is:
 - Online or offline
 - File alignment with reference client
 - Client is pending a solution update
- **NOTE:** When you clear the filter, the unchecked options still appear. To remove these, press CTRL+R to refresh the page.

Desktop Client - Control Room

The screenshot shows the 'Desktop Client Control Room' page. On the left, there's a sidebar with filters for 'DESKTOP CLIENT STATE', 'SOLUTION UPDATES', 'DESKTOP CLIENT REFERENCE', 'TEAMS', 'LOADED SOLUTIONS', and 'ASSIGNED SOLUTIONS'. Below the sidebar is a 'Filters' button. The main area displays a table of clients with columns for 'DS LOGIN', 'SOLUTION VERSION', and 'STATE'. A blue callout box points to the 'STATE' column, stating: 'Indicator whether the Client is running or not (Online vs. Offline)'. Another blue callout box points to the 'SOLUTION VERSION' column, stating: 'Dll and config file alignment with reference client'. A third blue callout box points to the 'ASSIGNED SOLUTIONS' row for the first client, stating: 'Solution has been assigned but not yet downloaded'. At the bottom right of the main area is a 'Client properties and actions' button. The top right corner of the page shows the user 'amadmin' and the NICE logo.

- State - Indicates whether the client is:
 - Online or offline
 - File alignment with reference client
 - Client is pending a solution update
 - Reference client

Desktop Client - Control Room



- By clicking the info button you can view additional information about the selected Real-Time Client, over and above what appears in the Control Room grid.
- Use the info window for further client investigation such as list of files that do not match or multiple solution assignments.
- If all the program files match then **an All installed file versions match the reference file versions** message appears.
Similarly, if all the config files match, **an All installed config files match the reference config files'** message appears.

Restarting Real-Time Clients

- An administrator can remotely restart the client from the Automation Portal:



Request Restart



Force Restart

NICE®

- Real-Time Clients sometimes need to restarted to load a new solution, update a configuration file, or if the client freezes.
- An administrator can remotely restart the client from the Automation Portal without the need to physically be at the machine or sign in with their own user ID
- **Request Restart** – A message pops up on the agent's workstation, asking to restart. If the agent accepts the request, the Real-Time Client is restarted. This option enables the agent to complete their work and save all files. If the agent rejects the request, the administrator can send it again later or force a restart, the restart request does not automatically get resent
- **Force Restart** - This immediately restarts the Real-Time Client. The agent does not receive a warning message. If an agent is in the middle of a solution, it will not be completed. The agent's files will not be saved. This option should be used with caution.

Restarting Real-Time Clients

- To restart Real-Time Client remotely:

The screenshot shows the NICE Automation Portal interface. On the left, a sidebar has sections for 'AUTOMATION PORTAL', 'Control Room', 'CLIENT STATE' (with 'Online' and 'Offline' counts), 'SOLUTION UPDATES' (with 'Clear'), and 'ASSIGNED SOLUTIONS' (with 'Not Specified'). A central table titled 'Showing 1 - 3 of 3 Clients' lists three clients: DANDC631B (OS LOGIN: Administrator, USER: qa@l0, STATE: Online, ACTION: Set as Reference, three dots menu open) and DANDC631B (OS LOGIN: qa@l0, USER: qa@l16, STATE: Online, ACTION: Request Restart, Force Restart, Delete). A blue callout points to the three dots menu with the text: 'Locate the client you want to restart and then click on the three dots [::]'. In the foreground, a modal window titled 'Select Request Restart or Force Restart' contains the message 'Desktop Client Restart' and 'Are you sure you want to send a restart command to NICE_SYSTEMS\TalC?'. It has 'Yes' and 'No' buttons. A blue callout points to the 'Yes' button with the text: 'Respond to the confirmation'.

- Note:** Make sure you have permissions to edit Clients

Restarting Real-Time Clients

- To see the status of the restart action:

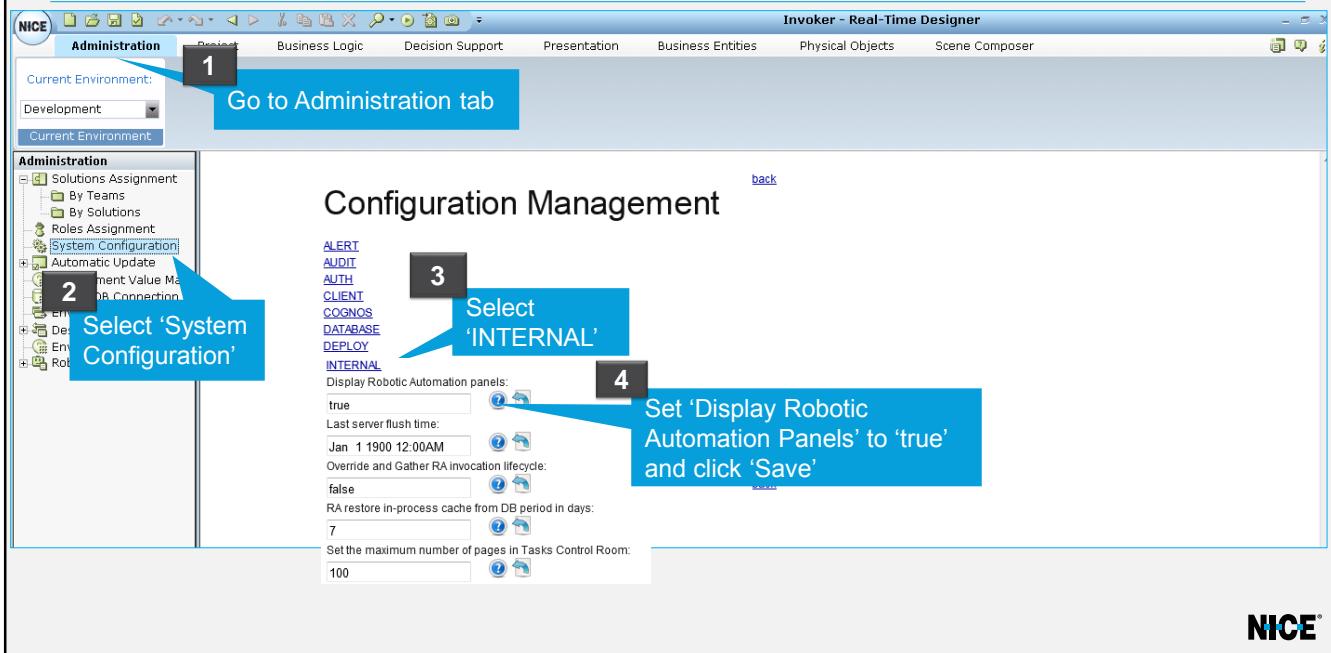
The screenshot shows the NICE Automation Portal's Control Room interface. On the left, there are several filter panels: Client State (Online: 0, Offline: 3), Solution Updates (Pending: 1, No Pending: 2), Reference Client (Aligned: 1, Not Aligned: 0), Teams (Not Specified), Loaded Solutions (Not Specified), and Assigned Solutions (Not Specified). A central grid displays client details for three clients named 'Administrator'. An information window titled 'e-glue\g3 Info' is open over the grid, showing details for the first client. The window includes sections for OS Login, Host, Solutions, File Versions Alignment, and Last Remote Operation. The 'Last Remote Operation' section highlights a recent forced restart. A blue callout box points to the 'Information Window' button in the top right of the window, with the text 'Click on the Information Window button'. Another blue callout box points to the 'Last Remote Operation' section with the text 'Refer to Last Remote Operation'.

- If the agent rejected a restart request, you can resend the request later or force a restart. The restart request does not automatically get resent
- All actions are reported in one of the following log files:
 - On the Real-Time Server:**
 - <Install Dir>\RTServer\logs\RTServer.log
 - <Install Dir>\RTServer\logs\RTSMicroServices\RTServerMHService.out.log
 - On the Real-Time Client:**
 - <Install Dir>\RTClient\logs\RTClient.log

Robotic Client



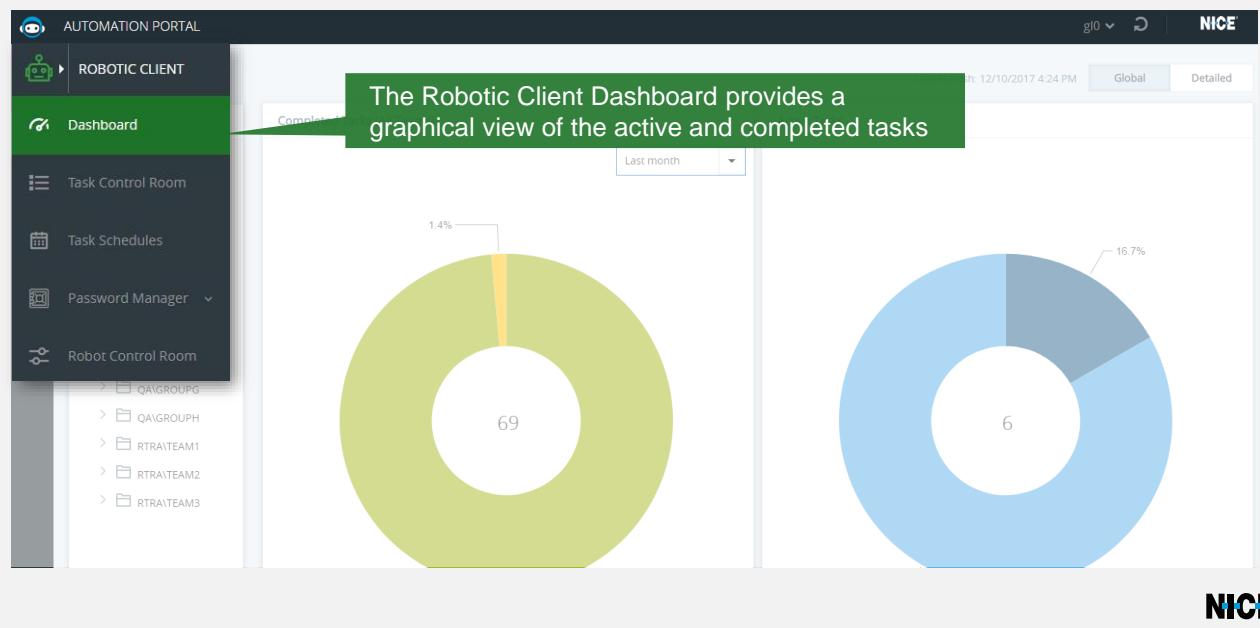
Enabling Robotic Client Panels In Automation Portal



*Using Robotic Client requires a separate Robotic Automation license.

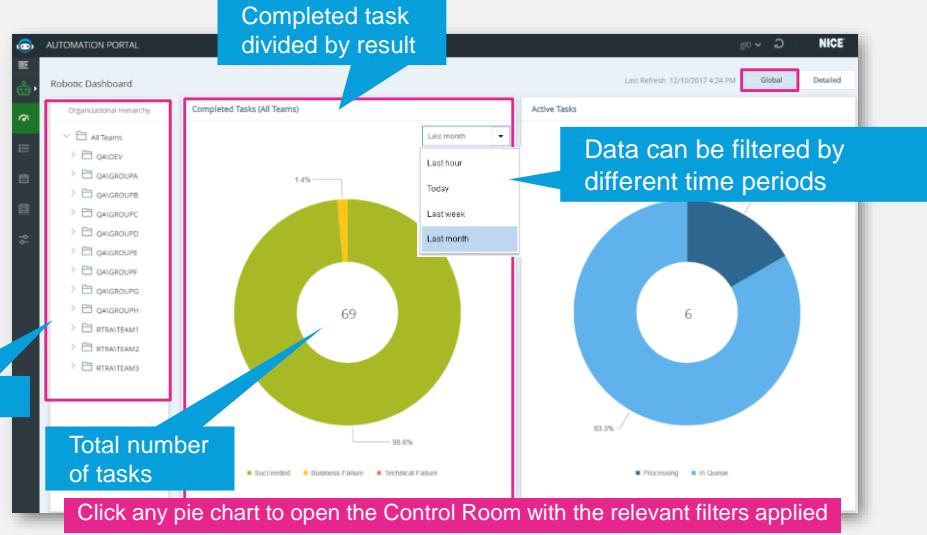
- In order to use the Robotic part of the Automation Portal we first need to enable it.
This can be done from the Designer or directly from the Server.

Robotic Client Dashboard



Robotic Client Dashboard

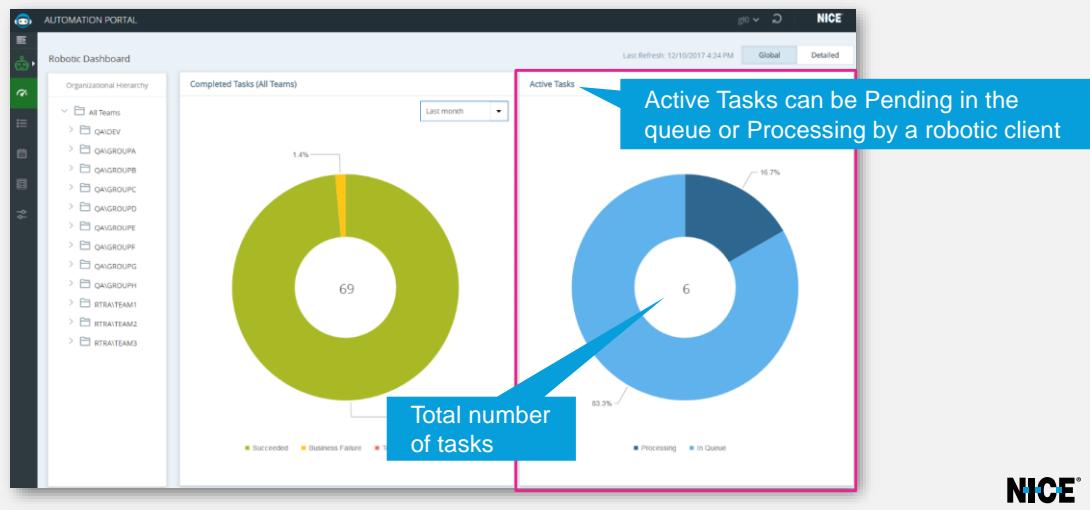
- The Robotic Client Dashboard has two widgets: **Global**(default) and Detailed.



- Global**(default)- provides a graphical view of the active and completed tasks.
- The dashboard is automatically updated approximately once every 60 seconds.
- To manually refresh the dashboard, press F5 or click the refresh icon.

Robotic Client Dashboard

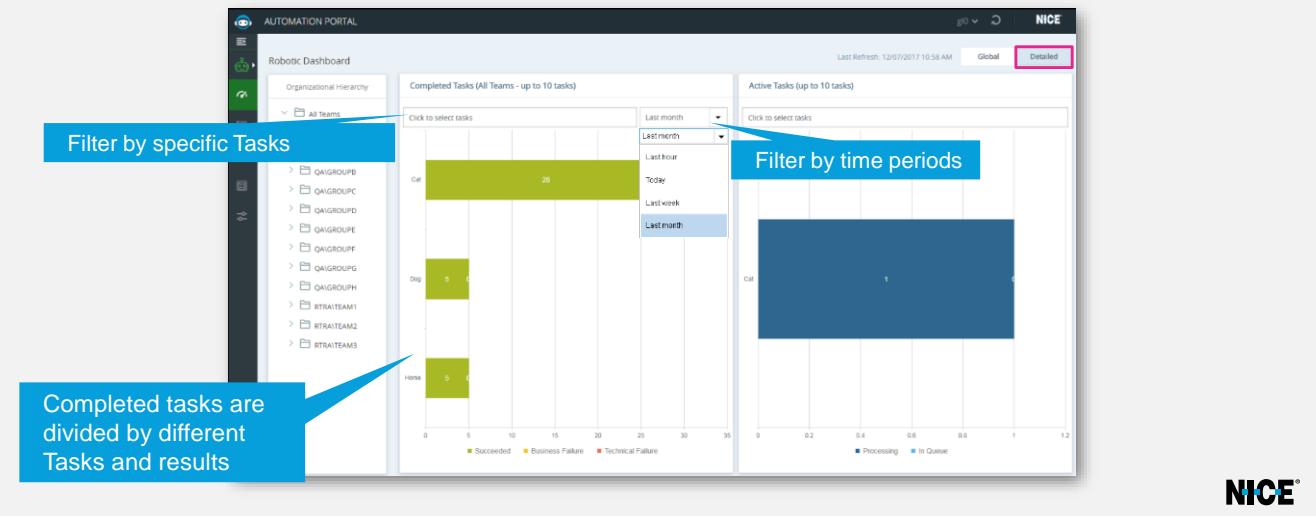
- The Robotic Client Dashboard has two widgets: **Global**(default) and Detailed.



- The Active Tasks area provides a color-coded pie chart (in percentages) of the active tasks (Queued or Processing). The total number of active tasks, appears in the center of the chart.

Robotic Client Dashboard

- The Robotic Client Dashboard has two widgets: Global(default) and Detailed.



- The Detailed pane view displays a graphic view of how many times each task ran.
- Up to 10 tasks can be shown at a time. You can select which tasks to show. The default is to show the top 10 tasks by priority.
- Priority was determined when the task was invoked. Priority cannot be changed

Task Control Room

The screenshot shows the 'Task Control Room' section of the Automation Portal. On the left, there's a sidebar with 'Dashboard', 'Task Control Room' (which is selected and highlighted in green), 'Task Schedules', 'Password Manager', and 'Robot Control Room'. Under 'Task Control Room', there are filters for 'TEAM NAME' (Not Specified) and 'TEAMS' (LOW, Medium, High). A modal window titled 'Showing 1 - 12 of 12 tasks' is open, listing the following tasks:

- View task status
- Filter and search for specific tasks, and view task details
- Remove tasks that are waiting in the queue and have not started

Task Name	Type	Run Type	Created	Duration	Last Run	Priority	Actions
Hello World Workflow	Scheduled	04/16/2018 11:20 AM	02:48:47		04/16/2018 1:44 PM	High	Start Details Remove
4 Steps Workflow	Sync	04/16/2018 1:40 PM	00:03:00	00:00:42	04/16/2018 2:07 PM	Medium	Start Details Remove
4 Steps Workflow	Sync	04/16/2018 2:04 PM	00:00:00	00:02:52	04/16/2018 2:07 PM	Medium	Start Details Remove
4 Steps Workflow	Scheduled	04/15/2018 6:44 PM	00:00:00	00:01:46	04/15/2018 6:45 PM	High	Start Details Remove
Hello World Workflow	Async	04/15/2018 6:47 PM	00:00:15	00:00:00	04/15/2018 6:48 PM	High	Start Details Remove
Hello World Workflow	Async	04/15/2018 6:48 PM	00:00:00	00:00:11	04/15/2018 6:48 PM	High	Start Details Remove

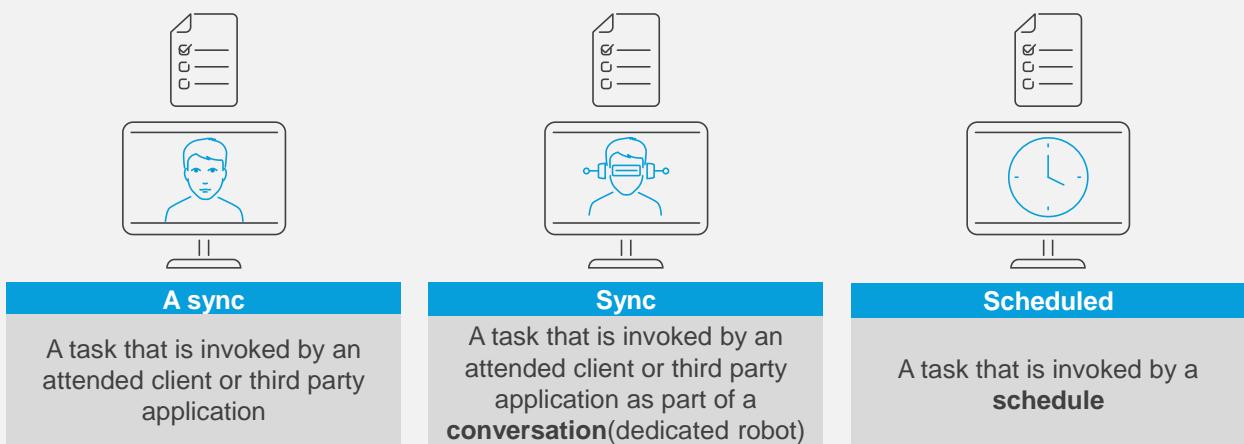
NICE®

Task Control Room

Showing 1 - 26 of 26 Tasks								
	TASK NAME	TYPE	INVOCATION TIME	TIME IN QUEUE	PROCESSING T...	COMPLETION TIME	PRIORITY	STATUS
<input type="checkbox"/>	Hello World Workfl...	Sync	04/15/2018 7:05 PM	00:00:00	00:00:03	04/15/2018 7:05 PM	Medium	●
<input type="checkbox"/>	Workfl...	Async	04/16/2018 11:10 AM	00:01:04	00:00:03	04/16/2018 11:11 AM	Medium	●
<input type="checkbox"/>	Task type	Async	04/16/2018 11:18 AM	02:50:35			High	●
<input type="checkbox"/>	Invocation date		04/16/2018 11:20 AM	02:48:47			High	●
<input type="checkbox"/>	4 Steps			00:03:00	00:00:42	04/16/2018 1:44 PM	Medium	●
<input type="checkbox"/>	4 Steps Workflow	Sync	04/16/2018 1:44 PM	00:00:00	00:02:52	04/16/2018 2:07 PM	Medium	●
<input type="checkbox"/>	4 Steps Workflow	Scheduled		00:01:46		04/16/2018 6:45 PM	High	●
<input type="checkbox"/>	Hello World Workfl...	Async	04/15/2018 6:47 PM	00:00:00	00:00:11	04/15/2018 6:48 PM	High	✗
<input type="checkbox"/>	Hello World Workfl...	Async	04/15/2018 6:48 PM	00:00:00	00:00:11	04/15/2018 6:48 PM	High	✓
<input type="checkbox"/>	Hello World WF 1	Sync	04/15/2018 6:51 PM	00:00:00	00:00:02	04/15/2018 6:51 PM	High	✓

- What you can do in this view:
 - Identify Type of Task
 - Search for a specific task - Type in the Search field. The Task Name and Priority columns are searched, as well as the parameters in the Info window. The search finds partial words or you can use wildcards.
 - Sort according to any of the columns shown except for Time in Queue and Processing Time.
 - Remove a task - Select one or more tasks and click Remove. This removes the task from the queue. It does not delete the task. You will still see the task in the Tasks Control Room for tracking purposes.
 - View task details
- By default, up to 100 pages of tasks appear in the Tasks Control Room. This number of pages is configurable
- **NOTE:** from version 6.7, all times that appear in the Robotic Client module are shown in local time. Previously, in earlier versions, the times shown in the Dashboard and Task Control Room were in UTC

Task Control Room- Task Type



NICE®

- Identify **Type of task:**
- **A sync** -A task that is invoked by an attended client or third party application. This task will run only one time.
For example, an attended client invokes a task to place and ship an order.
- **Sync** -A task that is invoked by another robot. This task is part of a conversation.
For example, canceling an order is initiated by an attended robot, but then that robot invokes a different robot to log the cancellation and send a confirmation message.
- **Scheduled**-A task that is invoked by a schedule. A separate instance of the task appears for each time that it is invoked by its schedule.
For example, if a schedule is set to run Monday - Friday, then each week you will see 5 instances of this task

Task Control Room- Task Status



Succeeded: Tasks that finished with no errors



Business Failure: From a technical aspect the robot completed the workflow successfully. However, from a business perspective the workflow failed



Technical Failure: Tasks can fail for technical reasons such as a robot was unavailable, or in the case of a scheduled task, the end of the starting range was reached and the task never started or was forced to stop



Processing: Tasks that are currently running



In Queue: Tasks that have been invoked and are waiting for a robot

NICE®

- **Business failure:** For example, if the workflow is intended to transfer funds from one account to another, but the balance is insufficient, the workflow cannot be completed, but not because of a system or technical problem. Causes for Business Failures are defined in the Designer by a Business Entity called 'Business Execution Status'.
- **Processing:** You will not see these tasks when you filter by team because the robot that ran the task is not identified until completion

Exceptions in RT Client

- When a technical or a business condition is not met, there is a break in the RT Client workflow. This break is called an exception.
- Real Time Client has two exceptions types:



Business Exceptions



Technical Exceptions

NICE®

- Even if there are many exceptions while the workflow is running, the workflow might still succeed. You can review the exceptions in the Task info dialog box after the workflow is complete. All exceptions are stored in the RT Server database
- Business Exceptions: If the business logic set in a solution does not match a condition in the RT Client a business exceptions occurs. A business exception will be raised by RT Client if **BusinessExecutionStatus** flag is set to “**False**” while execution of a workflow.

Technical Exceptions



Technical Exceptions

Incorrect Parameters Exception	The task fails due to entry of incorrect number parameters	Execution Exception	The task fails due to flaws in execution. For e.g. division by 0, file not found, incorrect number format, etc.
Execution Timeout Exception(Orphan)	Occurs when the task times-out during its execution. This could be due to a disconnect between RT Server and the RT Client	In-queue Timeout Exception (Backlog):	Occurs when the task is not picked up from the queue after end of the Backlog Timeout value in the Server Configuration

NICE®

- If an exception occurs inside the work-flow, libraries, rules and transitions a technical exception occurs. Technical exception also includes In-queue timeout and Execution timeout
- **A Robotic RT Client is capable of handling exceptions for the following libraries:** Math, Clipboard, File, Launch In Context, Conversion, Regular expressions, Administration, Environment, Text, Remote Channel, Customer Entity, Real-Time Authentication, Business Data, Web Site, Data Base Table, Data Base Server Command, Data Collection, Call-out, Business Entity, Run Time Functions, Credential Management, Date Time, Encryption, Error Handling, Launch Bar, Location Based Cases, MS Office Functions, Password Management, Robotic automation service, Value Mapping.
- Apart from the above libraries, RT Client is also capable of handling exceptions due to custom library objects

Exception Details on Task Info page

Viewing Additional Task Info and Parameters

- Here is more information about each task than can be seen in the Task Control room, such as the hostname of the robot that performed the task and parameters that were sent to the task at the time it was invoked.
- Only manually invoked tasks send parameters. You can not edit parameters in this window

Task Control Room- Removing a Task from the Queue

The screenshot shows the 'Robotic Task Control Room' section of the Automation Portal. On the left, there are several filter options: 'TASK STATUS' (Removed, Business Failure, Technical Failure, Processing, In Queue, Succeeded), 'TIME PERIOD' (Not Specified, Set time period), 'PRIORITY' (Low, Medium, High), 'TASK NAME' (Not Specified, Add / Remove), and 'TEAMS' (Not Specified, Add / Remove). A central table lists 26 tasks, with the first two rows visible:

NAME	PRIORITY	STATUS	
Workflow with 2 Pa...	Medium	Running	
Hello World Workfl...	High	In Queue	
4 Steps Workflow	Medium	Running	
4 Steps Workflow	Medium	Running	
Hello World Workfl...	Medium	Running	
Hello World Workfl...	High	Running	
Hello World WF 1	Medium	Running	

A blue callout box highlights the 'Remove' icon for the second task and contains the text: 'A removed task remains in the Task Control room for tracking purposes. You can apply a filter to hid the removed tasks'. Another blue callout box highlights the 'Remove' icon for the last task and contains the text: 'You can only remove a task that did not start (status = In Queue). A task that currently running cannot be removed.'

- A task that is removed is no longer included in Dashboard statistics.
- Remove a task - Select one or more tasks and click Remove . This removes the task from the queue. It does not delete the task.
- You will still see the task in the Tasks Control Room for tracking purposes. If the task was a scheduled task, it will be re-invoked at its next scheduled time. To re-invoke a manual task, you have to use the re-invoke API.

Task Schedules

The screenshot shows the 'Task Schedules' section of the NICE Automation Portal. The left sidebar includes 'Dashboard', 'Task Control Room' (selected), 'Task Schedules' (highlighted in green), 'Password Manager', and 'Robot Control Room'. The main area displays a table of scheduled tasks:

SCHEDULE DETAILS	NEXT SCHEDULED RUN	LAST SCHEDULED RUN	STATE	ACTIONS
Daily between 10:10 AM - 11:00 AM	12/11/2017 10:10 AM	12/10/2017 10:10 AM	ON	More
Daily between 10:14 AM - 11:00 AM	12/11/2017 10:14 AM	12/10/2017 10:14 AM	ON	More
Daily between 1:55 PM - 2:30 PM	12/10/2017 1:55 PM	12/10/2017 1:18 PM	OFF	More
Daily between 6:00 AM - 9:00 AM	12/11/2017 6:00 AM	12/10/2017 6:00 AM	ON	More
Daily between 10:10 AM - 11:00 AM	12/11/2017 10:10 AM	12/10/2017 10:10 AM	ON	More
Daily between 10:14 AM - 11:00 AM	12/11/2017 10:14 AM	12/10/2017 10:14 AM	ON	More

A callout box highlights the 'Task Schedules' section with the following text: "Tasks that do not require a manual trigger and can be run by an unattended client (robot) can be prescheduled to enter the task queue at regular intervals. Once a task is in the queue, it will run when a robot becomes available."

- Scheduling tasks requires a Robotic Automation license.

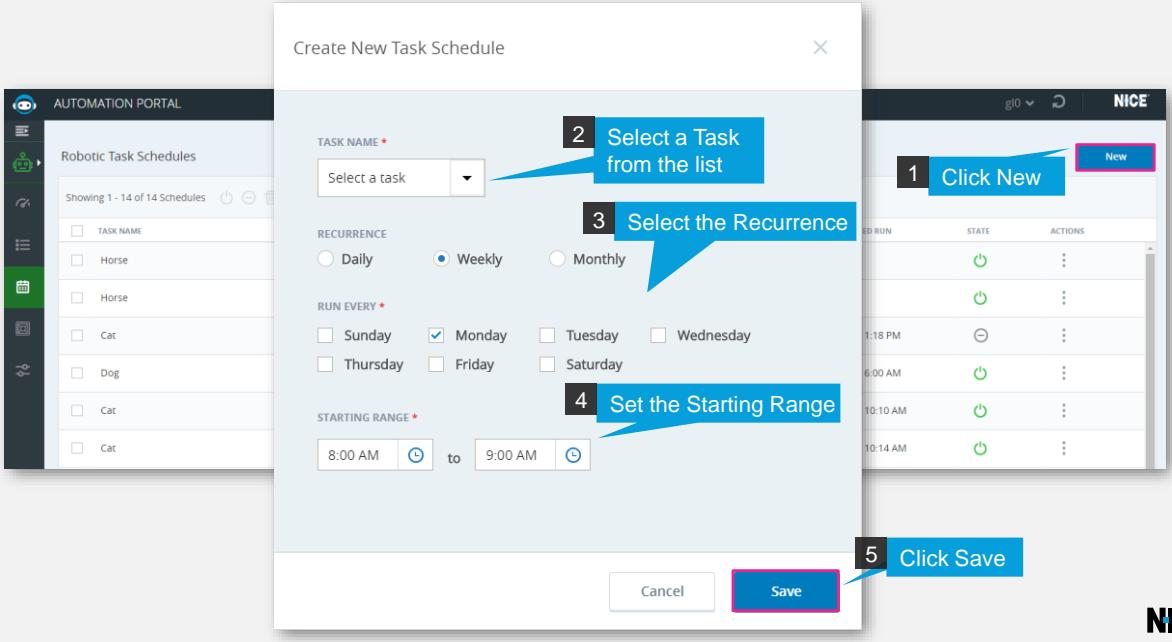
Scheduling a Task

To make a change to a single schedule, click a schedule's **Action menu**

TASK NAME	SCHEDULE DETAILS	NEXT SCHEDULED RUN	LAST SCHEDULED RUN	ACTIONS
Horse	Daily between 8:00 AM - 9:00 AM	12/07/2017 8:00 AM		
Horse	Daily between 2:20 PM - 4:00 PM	12/07/2017 2:20 PM		
Cat	Daily between 1:55 PM - 2:30 PM	12/10/2017 1:55 PM	12/10/2017 1:18 PM	
Dog	Daily between 6:00 AM - 9:00 AM	12/11/2017 6:00 AM	12/10/2017 6:00 AM	
Cat	Daily between 10:10 AM - 11:00 AM	12/11/2017 10:10 AM	12/10/2017 10:10 AM	
Cat	Daily between 10:14 AM - 11:00 AM	12/11/2017 10:14 AM	12/10/2017 10:14 AM	

- **Enable a schedule:** Reinstate a schedule that has been disabled. The task will be invoked at the next scheduled time. Scheduled times that passed while the task was disabled will not be invoked.
- **Disable a schedule:** Stop a schedule from sending more tasks to the queue.
- **Delete a schedule:** Tasks that are already in the queue from this schedule will not be deleted and will run as scheduled.
- Editing changes made to a schedule will not affect tasks that are already in the queue, only those that haven't been invoked yet.

Scheduling a Task



- Task Name: only published tasks with no parameters can be scheduled
A task can be scheduled more than once-for example, schedule a task that updates the currency rates on your website at 9:00 am and then again at 5:00 pm. This requires 2 separate schedules for the same task.
- Starting Range: When you create a schedule, you set a time range for the task to start. If the task does not start within that range, it will be purged from the queue and will enter the queue at it's next scheduled time slot.
- For example, A task is scheduled to start daily between 8:00am and 9:00 am. On Monday at 9:00 am it hasn't started because a robot wasn't available(becomes a Technical failure), It will not run on Monday but will re-enter the queue on Tuesday at 8:00 am.
- Once a task schedule is created, the schedule invokes its task at each scheduled time.

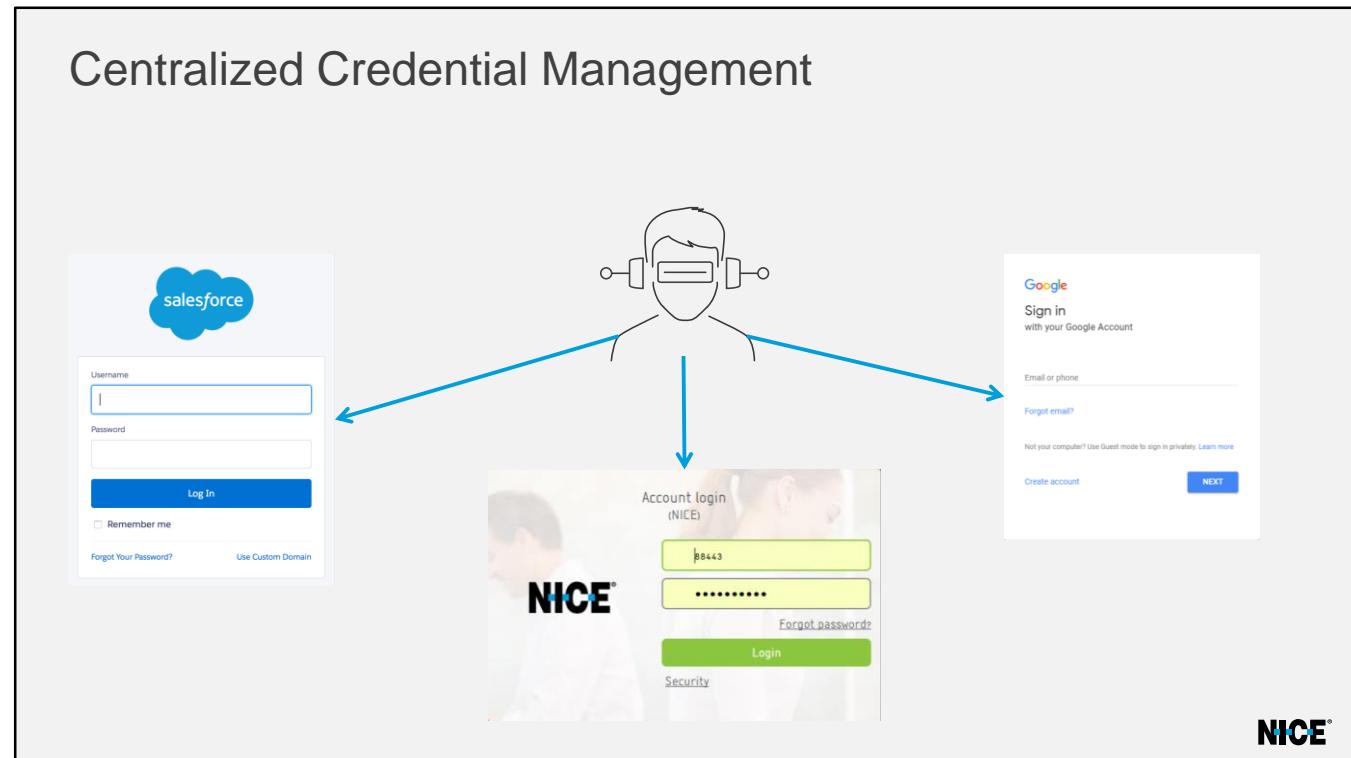
Password Manager

The screenshot shows the 'Robotic Credentials Management' page within the 'Password Manager' section of the Robotic Client module. The left sidebar lists 'Dashboard', 'Task Control Room', 'Task Schedules', 'Password Manager' (which is expanded to show 'Applications' and 'Credentials'), and 'Robot Control Room'. The main area displays an 'Organizational Hierarchy' tree under 'All Teams' with nodes like 'E-GLUE\N-GLUE_GROUP', 'NICE_SYSTEMS\NICE_...', and 'QA\QA_GROUP'. Below this, a table lists 'Showing 1 - 3 of 4 Credentials':

NAME	APPLICATION	USER NAME	EXPIRY DATE
Robot Name 1	Bizdoc	GL0	01/01/18
Robot Name 2	SAP	GL1	01/02/18

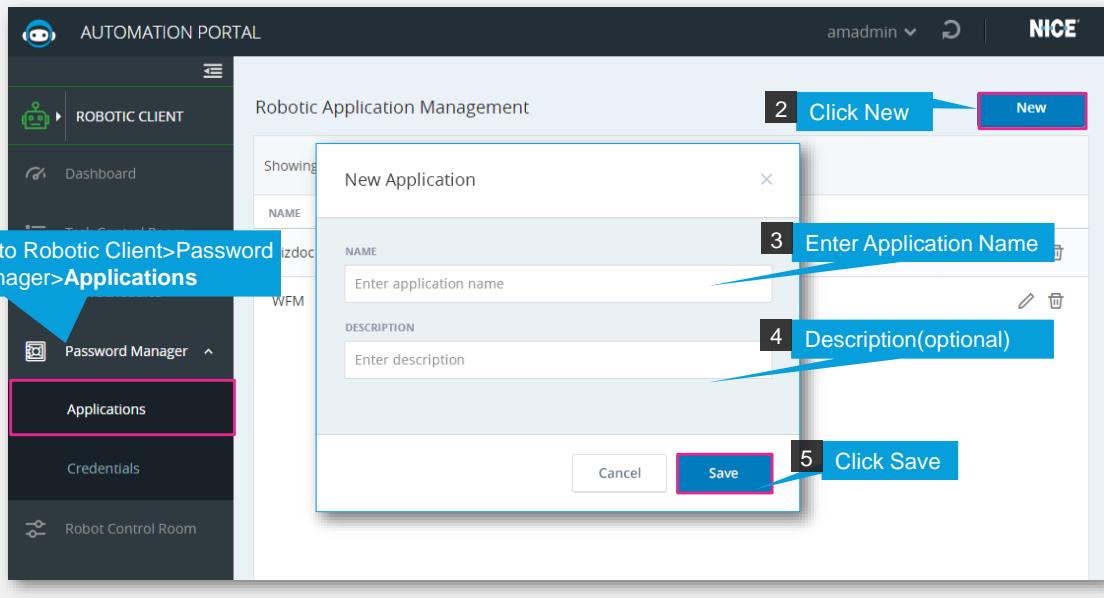
A green callout box highlights the 'Applications' section of the hierarchy tree and the credential table, stating: 'In the Passwords section of the Robotic Client module, you can define all applications and credentials for all robots. Different credentials can be assigned for each team of robots.'

Centralized Credential Management



- To access applications, robots must have valid credentials to these resources.
- These applications and credentials can be defined in the Automation Portal Password Manager.
- Different credentials can be assigned to each team of robots or to a specific robot.
- The robot uses the new CCM microservice to pull the credentials on demand for the applications from the Centralized Password Management repository.
- The passwords are encrypted and stored in the RT CCM DB and the data is secure both at rest and in transit.

Adding Applications for Password Control



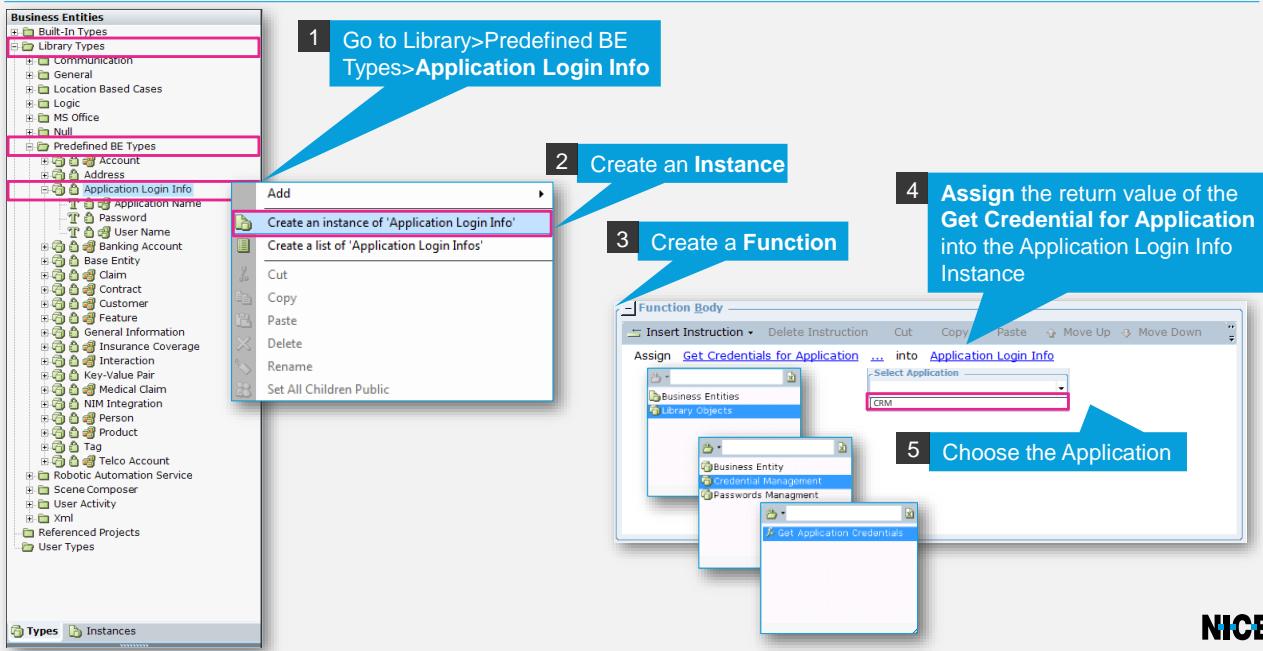
NICE®

Adding Applications for Password Control

The screenshot shows the 'AUTOMATION PORTAL' interface. On the left, a sidebar has 'ROBOTIC CLIENT' selected. Under 'ROBOTIC CLIENT', 'Passwords', 'Applications', and 'Credentials' are listed, with 'Credentials' highlighted by a pink box and a callout. A callout also points to 'Select a team' in the 'Organizational Hierarchy' dropdown, which lists 'All Teams' and several specific teams like 'QA\QA_GROUP', 'RTRAIRTRA_GROUP', 'RTRATEAM4', 'RTRATEAMS', and 'RTRATEAM6'. A callout points to 'Click New' in the top right of the main window. The main window title is 'Robotic Credentials Management'. It shows a 'New Credential' dialog box with fields: 'User' (set to 'All Users'), 'Application' (set to 'Bizdoc'), 'User Name' (empty), 'Password' (empty), and 'Expires' (set to '04/22/2018'). A callout points to 'Click Save' at the bottom right of the dialog. Numbered callouts 1 through 9 point to each step of the process: 1. Go to Robotic Client>Password Manager>Credentials; 2. Select a team; 3. Click New; 4. Select one robot or all users; 5. Select an application; 6. Enter a valid user name; 7. Enter a valid password; 8. Enter an expiration date for the password; 9. Click Save.

- **User** - You will see only the robots who belong to the team selected in the Organizational Hierarchy. Select one robot or All Users (all the robots in the team that was selected in the Organizational Hierarchy).
- **Application** - Select one application.
- **User Name** - Enter a valid user name that will be recognized by the application.
- **Password** - Enter a valid password. Automation Portal does not validate passwords.
- **Expires** - Enter an expiration date for the password

How to access Password Management in the Designer



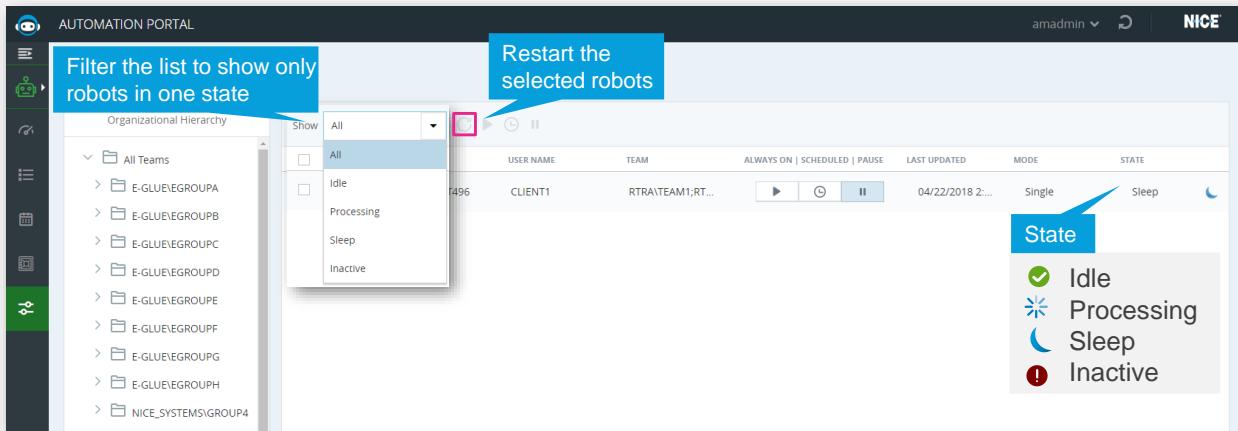
Robot Control Room

The screenshot shows the NICE Automation Portal interface. The top navigation bar includes 'AUTOMATION PORTAL' and 'ROBOTIC CLIENT'. On the right, there's a user dropdown for 'amadmin' and the 'NICE' logo. The left sidebar has a 'Robot Control Room' section highlighted in green, containing options like 'Dashboard', 'Task Control Room', 'Task Schedules', 'Password Manager', and 'Robot Control Room'. Below this is a tree view with nodes such as 'E-GLUE\EGROUPG', 'E-GLUE\EGROUPH', and 'NICE_SYSTEMS\GROUP4'. The main content area is titled 'Robot Control Room' and displays a table of robot activities. A callout box points to the 'Robot Control Room' section in the sidebar with the text: 'In the Robot Control Room you monitor and manage your robots. Here is where you will see your robots and their current activities'.

RESTART	HOST NAME	USER NAME	TEAM	ALWAYS ON SCHEDULED PAUSE	LAST UPDATED	MODE
<input type="checkbox"/>	W7X64RACLINT496	CLIENT1	RTRA\TEAM1;RT...		04/22/2018 2:...	Sing

NICE®

Robot Control Room- Managing Robots



- **Restarting a Robot:** Robots can be restarted from the Robot Control room.
- If a robot is in the middle of a task when restarted, the task is not completed and appears as a technical failure in the control room.
- When you restart a robot, all new solutions and updates to existing solutions are automatically downloaded to the robot
- **Robot States:**
 - **Idle** - The robot sends a 'keep alive' notification and listens to the queue but is not processing any solution currently.
 - **Processing** - The robot sends a 'keep alive' notification and listens to the queue and is currently processing a solution.
 - **Sleep** -The robot sends a 'keep alive' notification but is not listening to the queue. This indicator will also be displayed when a robot is inactive based on a schedule profile defined in the Real-Time Designer.
 - **Inactive** - The robot does not send a 'keep alive" notification, indicating that it is currently inactive.

Robot Control Room - Scheduling a Robot to Work

The screenshot shows the 'Robotic Client Control Room' section of the NICE Automation Portal. On the left, there's a sidebar with various icons and the title 'AUTOMATION PORTAL'. The main area displays an 'Organizational Hierarchy' tree under 'All Teams'. A table lists robots with columns for 'RESTART', 'HOST NAME', 'USER NAME', 'TEAM', 'ALWAYS ON | SCHEDULED | PAUSE', 'LAST UPDATED', 'MODE', and 'STATE'. One row is selected, showing 'W7X64RACLINT496' as the host name, 'CLIENT1' as the user name, and 'RTRA\TEAM1;RT...' as the team. The 'ALWAYS ON | SCHEDULED | PAUSE' row has three buttons: a play button (highlighted with a pink border), a pause button (highlighted with a pink border), and a stop button. Three callout boxes explain these states:

- 'Always on – The robot runs continuously' points to the play button.
- 'Pause - The robot enters a sleep state' points to the pause button.
- 'Scheduled- The robot runs according to the schedule defined in its profile' points to the stop button.

NICE®

Task Schedule Vs Robot Schedule



Task Schedule

Robot Schedule

What you schedule?

When the **Task** should run

When a **Robot** should be active

Why you use the schedule?

Invoking tasks that do not require a manual trigger and can be run by an unattended client (robot)

Flexibility to increase or decrease robot activity around important events such as updates, system maintenance and more

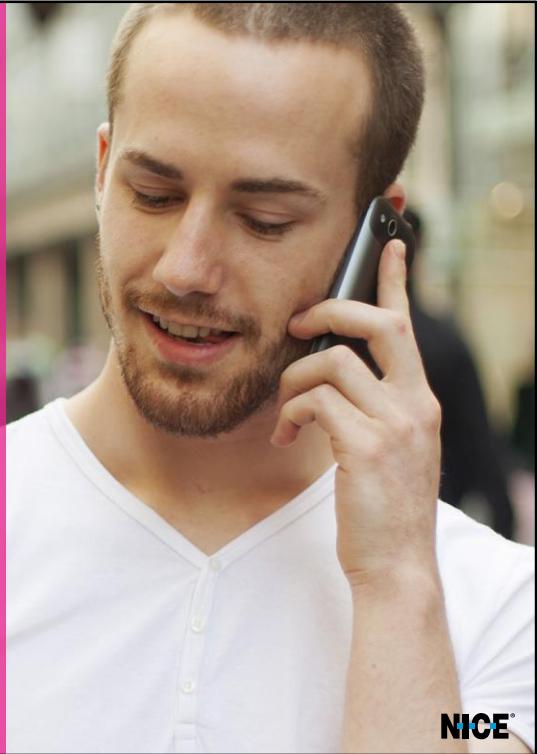
Where you set the schedule?

Task Schedules in the Automation Portal

Real Time Designer and **Robot Control Room**

NICE®

Admin



- The Admin module is where you import users to the server, set up roles and permissions, and also assign solutions to the users by teams. You can also change passwords for the databases, FTP connections, and version management systems installed in the RT Server.

Roles & Policies

The screenshot shows the 'Automation Portal' Admin interface. The left sidebar has a pink header 'ADMIN' with icons for Solution Assignment, User Manager, Roles & Policies (selected), and Password Manager. The main area has a 'PERMISSIONS' section with a 'New' button. It lists four items: 'View Clients' (with edit and delete icons), 'View Robots' (with edit and delete icons), 'Edit Robots' (with edit and delete icons), and 'Restart Robots' (with edit and delete icons). Below these are 'View Tasks' (with edit and delete icons) and 'Edit Tasks' (with edit and delete icons). A purple callout box highlights the 'Automation Portal Authorization is the process of authorizing or limiting user access' text.

- For any admin task, make sure that the OpenAM Administrator has assigned you to the Admin group with the respective authorization policies. The authorization policies in OpenAM set access to every function in the Admin module.

Enabling Authorized Access

- For policies and roles to be enforced, the Configuration Management parameter **Content authorization enabled** on the Real-Time server must be set to **true**.



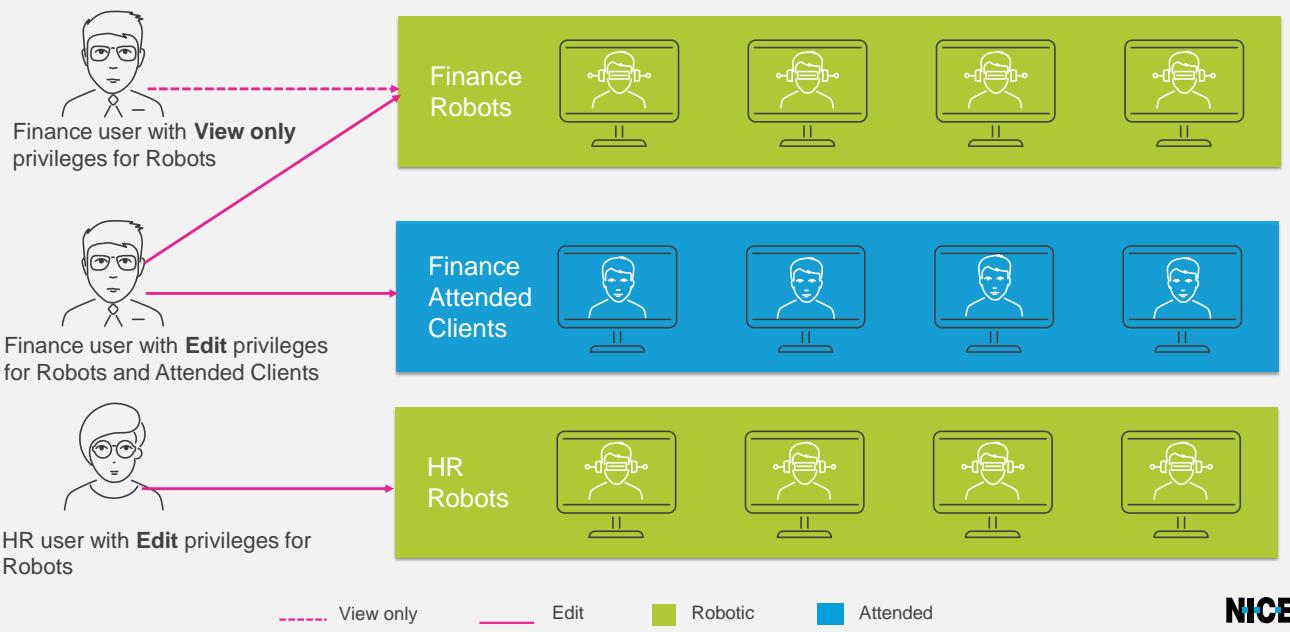
Note

Once Content authorization is enabled, you must set a role for your user, or else no data will be presented in the automation portal

NICE®

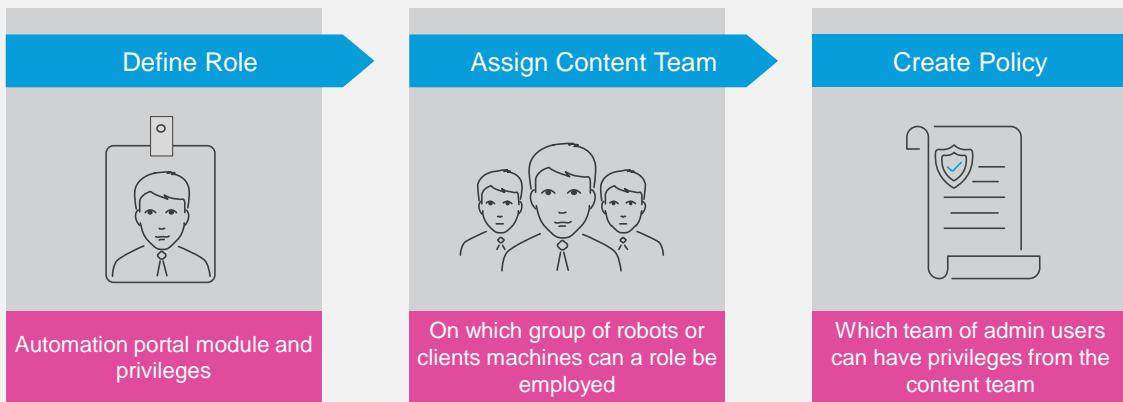
- The default setting is false, in which case you can define roles and polices, but they will not be enforced; any logged in user will have full capabilities.
- Server branch
- False = free access for all**
- True = polices are enforced**

Admin Authorization Overview



- **Automation Portal Authorization** is the process of authorizing or limiting user access. Authorization includes:
- What users are allowed to do - some users can only view control rooms and other users can also perform actions such as creating schedules or removing tasks.
- Which groups each user is allowed to access - viewing and other access privileges are given per team(groups).
- What areas of the Automation Portal are accessible to each user -users can be given limited access to only part of the control room applications.

Admin Authorization - High level workflow



NICE®

- First you create **Roles**-Each role is one or more resource-permission combination. The resources are either clients, tasks, or robots. You define which permission is allowable for each type of resource. For example, one role can give permission to edit tasks, but only to view clients. Another role can give permission to edit tasks and clients.
- Next you assign roles to teams - These are called **Content Teams**. Teams are groups of either attended (agent) or unattended (robot) client machines. Each team can have more than one role and the same role can be assigned to more than one team.
- To complete the security process, you create **Policies** - For each policy you pick a group of admin users and assign one or more Content Teams.

Roles and Authorization Privileges



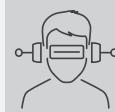
Roles Categories



Clients



Tasks



Robots

Permissions



View



Edit

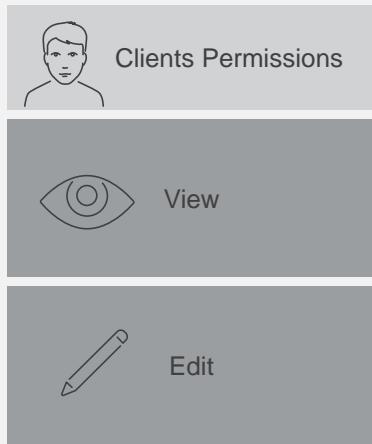


*Restart

NICE

* Restart permission is only relevant for Robots Category

Roles and Authorization Privileges- Example



In the Desktop Client view the:

- Dashboard
- Control Room

This includes searching and filtering the content

In the Desktop Control Room perform all actions such as:

- Set a reference client
- Remove a client

The user must also have the **View** permission



For more information about Roles and Authorization Privileges, refer to the Automation Portal User Guide.

NICE®

Creating Admin Roles



The screenshot shows the 'AUTOMATION PORTAL' interface with a sidebar containing icons for robots, clients, tasks, and categories. The main area is titled 'Roles' and shows a list of existing roles: VIEW CLIENTS, VIEW ROBOTS, EDIT ROBOTS, RESTART ROBOTS, VIEW TASKS, and EDIT TASKS. A modal window titled 'New Role' is open, prompting the user to 'Give the Role a meaningful name' (step 2). The 'NAME' field contains 'eg. Tasks Manager'. Step 3, 'Select a Category', points to the 'CATEGORIES' section which lists 'Clients', 'Tasks', 'Robots', and 'Solution Assignment'. Step 4, 'Select a Permission', points to the 'PERMISSIONS' section where 'View' and 'Edit' checkboxes are available. Step 5, 'Click Save', points to the 'Save' button at the bottom right of the modal. The 'NICE' logo is visible in the top right corner of the portal.

- You can select more **Categories** and more **Permissions**.

Creating Content Teams



AUTOMATION PORTAL

New Content

NAME: []

2 Give the Content Team a meaningful name
e.g. Robots Tasks Manager

1 Click New

New

3 Select one or more Roles

ROLES:

- VIEW CLIENTS
- EDIT CLIENTS
- VIEW ROBOTS
- EDIT ROBOTS
- RESTART ROBOTS
- VIEW TASKS
- EDIT TASKS

4 Select one or more Teams

TEAMS:

- E-GLUE\E-GLUE_GROUP (0)
- E-GLUE\GROUPA (0)
 - DIRECT_USERS (0)
 - DIRECT_USERS (0)
- E-GLUE\GROUPB (0)
 - DIRECT_USERS (0)
 - DIRECT_USERS (0)

5 Click Save

Cancel

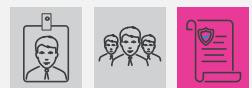
Save

Admin of TEAM 6

TEAM 6

NICE®

Creating Admin Policies



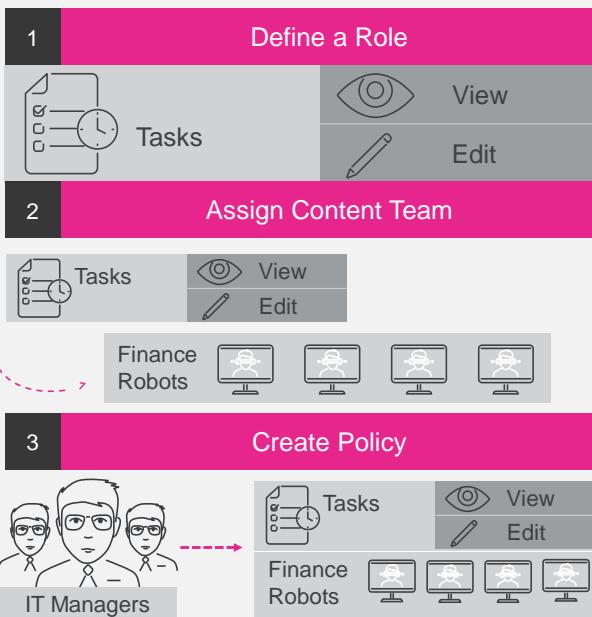
The screenshot shows the 'Policies' section of the Automation Portal. A 'New Policy' dialog is open, containing the following steps:

- 1 Click New
- 2 Give the Policy a meaningful name
- 3 Select a Role Assignment
- 4 Select a Group
- 5 Click Save

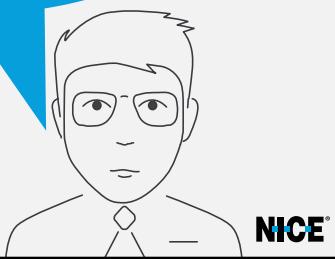
The 'NAME' field contains 'eg_Tasks Manager'. The 'TEAMS' section lists several groups: E-GLUE\E-GLUE_GROUP (0), E-GLUE\GROUPA (0), E-GLUE\GROUPAA (0) which includes DIRECT_USERS (0), E-GLUE\GROUPB (0), and E-GLUE\GROUPBB (0) which includes DIRECT_USERS (0). The 'CONTENT TEAM ASSIGNMENTS' section lists various permissions: View Clients Of CTA RTRA-TEAM2, Edit Clients Of CTA RTRA-TEAM2, View Robots Of CTA RTRA-TEAM1, Edit Robots Of CTA RTRA-TEAM1, Restart Robots Of CTA RTRA-TEAM1, View Tasks Of CTA RTRA-TEAM1, Edit Tasks Of CTA RTRA-TEAM1, and Admin of TEAM 6.

NICE®

Admin Authorization Example



David, IT Manager in a Bank
Needs to have permissions
to edit the Task Schedule of
the Finance robots team



- In order to meet David's needs we first need to define a Role: David needs to edit the Task Schedule so the Role category will be **Tasks** and the permissions will be view and edit(the user must have view permissions in order to have edit permissions)[‘What?’]
- After we defined the Role, we need to assign it to a content team-David needs to be able to edit the tasks of the **Finance robots**, therefore we assign the Schedule Tasks role to the Finance robots group.[‘On who?’]
- Finally, we need to give David permissions to edit scheduled tasks : David is part of the **IT Managers group** so we need to create a policy that enables the IT Managers group to perform those actions and we do it by assigning the content team to the IT Managers group.[‘Who is?’]

Solution Assignment by Team

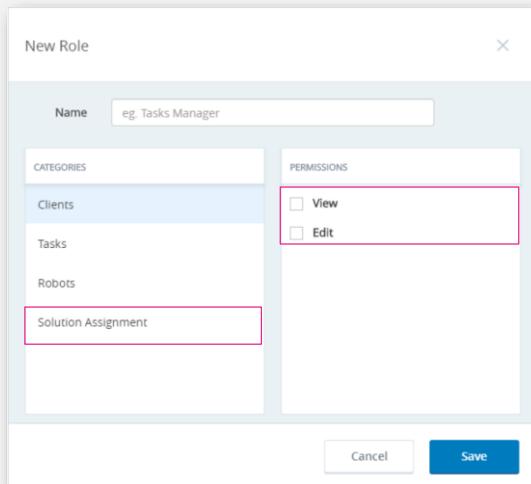
A screenshot of the Automation Portal interface. The top navigation bar includes the 'AUTOMATION PORTAL' logo, user 'amadmin', and the 'NICE' logo. The left sidebar has a dark header 'ADMIN' with icons for 'Solution Assignment', 'User Manager', 'Roles & Policies', and 'Password Manager'. Under 'Solution Assignment', there's a tree view with 'GroupCA', 'GroupCB', 'GroupD' (which is expanded to show 'GroupDA' and 'GroupDB'), and 'GroupE'. The main content area shows a table titled 'Parent Solutions' with columns: SOLUTION NAME, SOLUTION VERSION, CONTEXT, and LOAD ON STARTUP. A message at the top right says 'No items to display'. A red callout box points to the 'Solution Assignment' menu item in the sidebar.

Assigning Published solutions that were created in in Real-Time Designer or Automation Studio to the clients(teams)

SOLUTION NAME	SOLUTION VERSION	CONTEXT	LOAD ON STARTUP
No items to display			

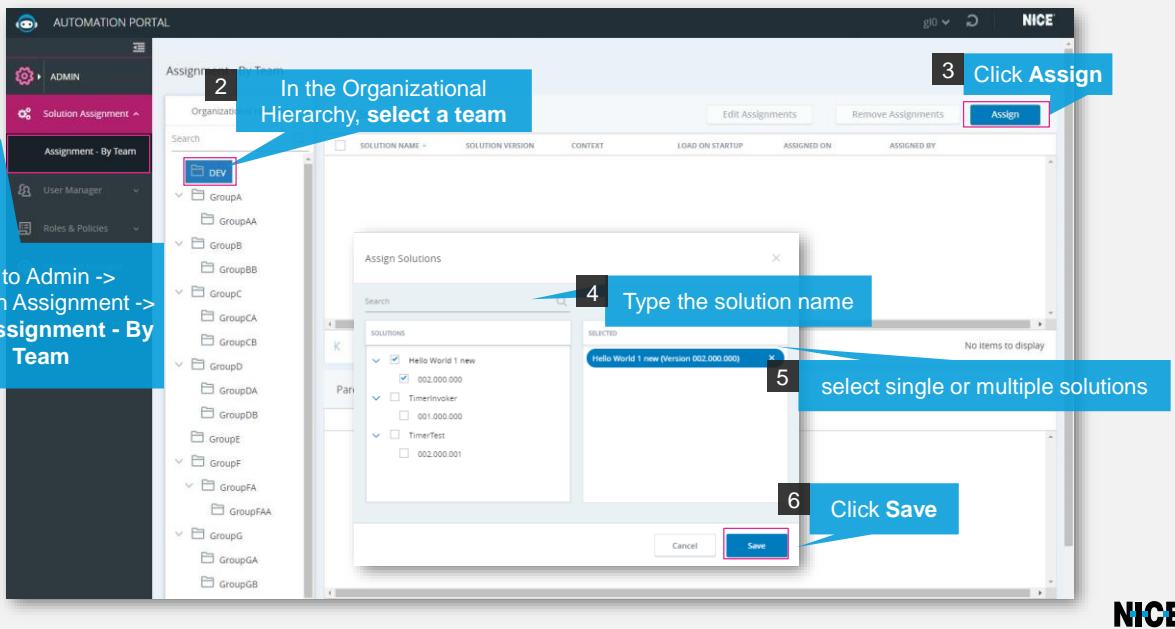
Assigning Solutions by Team

- Make sure that you have the required permission to view or assign solutions to a team:



NICE®

Assigning Solution by Team



NICE®

Removing Assignment

Multiple Solutions: Select one or more solutions and click Remove Assignments

Single Solution: Click the Remove Assignment icon on each solution row

SOLUTION NAME	SOLUTION VERSION	CONTEXT	LOAD ON STARTUP	ASSIGNED ON	ASSIGNED BY
Hello World 1 new	002.000.000		Yes	08/02/2018 6:48 PM	QA\GL0
TimerTest	002.000.001		Yes	08/02/2018 6:48 PM	QA\GL0

NICE®

Editing Assignment

The screenshot shows the Automation Portal interface. On the left, there's a sidebar with icons for assignments, organization, search, and other functions. The main area has a modal window titled 'Edit Solution' over a list of solutions. The 'Edit Solution' window contains fields for 'SOLUTION NAME' (Calculate Interest), 'ASSIGNED ON' (07/25/2018 1:54 PM), 'VERSION' (001.000.001), 'CONTEXT' (empty), and 'LOAD ON STARTUP' (Yes). Below the modal is a table titled 'Edit Assignments' with two rows. Each row has columns for 'LOAD ON STARTUP', 'ASSIGNED ON', and 'ASSIGNED BY'. The first row has 'Yes', '08/02/2018 6:48 PM', and 'QA\GLO'. The second row also has 'Yes', '08/02/2018 6:48 PM', and 'QA\GLO'. To the right of the table is a blue callout box with the text 'Multiple Solutions: Select more than one solution, and then click Edit Assignments'. Another blue callout box points to the edit icon in the second row of the table with the text 'Single Solutions: Click the Edit Assignment icon on each solution row'. In the bottom right corner of the screenshot, there's a 'NICE' logo.

- **When you edit single solution, you can change:**
 - **Version:** Changes the version of the solution that is assigned to the team
 - **Context:** Description of the solution, which is visible to the Real-Time Client users
 - **Load on Startup:** Determines whether or not the solution loads when the Real-Time Client is launched
- **When you edit Multiple solution, you can change:**
 - **Context:** Description of the solution, which is visible to the Real-Time Client users. When you leave this field blank, existing text for each of the individual solutions is deleted
 - **Load on Startup:** Determines whether or not the solution loads when

the Real-Time Client is launched. The setting you choose affects all selected solutions.

Modifying Solution Inheritance

The Parent Solutions list contains solutions assigned to the parent of the team selected in Organizational Hierarchy

1 In the Solution Inheritance list, select a behavior type

2 Click Edit

- You can either add assigned solutions to team from its parent team, or you can replace team solutions by their parent solutions.
- **Solution Inheritance types:**
 - **Default:** Displays solutions as per the inheritance behavior defined in the Real-Time Server properties
 - **Inherit:** Assigns solutions from the parent of the selected team. The solutions that are common in the team and its parent are listed under the Team Solutions list.
 - **Replace:** Retains the solutions only assigned to the team. The solutions in the Parent Solution list appears blank after selecting this behavior.

Modifying Connectivity Watcher Behavior

Select Enabled to activate the Connectivity Watcher for all assigned solutions

Select Enabled and set a Relearn Date to ignore previously gathered data and restart the learning mode

Click Edit

SOLUTION NAME	SOLUTION VERSION	CONTEXT	LOAD ON STARTUP	ASSIGNED ON	ASSIGNED BY
Hello World 1 new	002.000.000		Yes	08/02/2018 6:48 PM	QA\GLD
TimeTest	002.000.001		Yes	08/02/2018 6:48 PM	QA\GLD

SOLUTION NAME	SOLUTION VERSION	CONTEXT	LOAD ON STARTUP
PSolution1	001.001.001		Yes
PSolution10	001.001.010		Yes
PSolution11	001.001.001		Yes

SOLUTION INHERITANCE: Default
CONNECTIVITY WATCHER: Enabled
RELEARN: Enabled
RELEARN DATE: December 09, 2018 10:58 AM

- You can activate Connectivity Watcher and relearn this capability for solutions assigned to a team. (Connectivity Watcher recognizes connectivity breaks between the GUI on the Real Time Client machine and the solution in run-time.)
- Relearn Mode: Each Real-Time client which belongs to the team will compare the Relearn Date to the last time the Connectivity Watcher activated the learning mode and will begin
- The Relearn phase if the Relearn Date is newer.

Importing Users

The screenshot shows the 'User Import' page within the 'ADMIN' section of the 'AUTOMATION PORTAL'. The left sidebar includes 'Solution Assignment', 'User Manager' (which is currently selected), 'Role Assignment', 'Roles & Policies', and 'Password Manager'. The main area displays 'Import Details' with fields for 'Type', 'Start time', 'End time', 'Imported by', and 'Summary'. A large button at the bottom right says 'Import users into the RT Server, along with their team structure, via the staticUsers.txt file'. The top right corner shows the user 'amadmin' and the 'NICE' logo.

NICE®

Importing Users

- Import users into the RT Server, along with their team structure, via the **staticUser.txt** file

Must contain these three main sections:
#Users, **#User Groups**, and **#Groups**

The file name must be **staticUser.txt**

After each section, there must be a blank line

```
#Users
user_id,first_name,middle_name,last_name,os_login,domain
user1,First1,,Last1,test1,
user2,First2,,Last2,test2,
user3,First3,,Last3,test3

#User Groups
USER_ID,GROUP_ID,IS_MANAGER
user1,group1,1
user2,group1,0
user3,group2,0

#Groups
ID,NAME,PARENT_ID
group1,Group 1,-1
group2,Group 2,group1
```

Each section in the file has a required header section for the information

After the header section, you enter the user information
(each bit of user information must be separated by a comma)

NICE®

- On the RT Server, the staticUser.txt file is located at this path :
<installation folder>:\nice_systems\RTServer\config\properties
- Important:** Make sure the admin user who is importing new users is mapped to the UserImportActions authorization policy in OpenAM.
- Note: If no information is available, an additional comma is still required

Importing Users Via the Automation Portal

- You can Import Users to the server using one of these methods:

Run Import

The User Import job imports the last uploaded file to the server

Upload to Server

The file is uploaded to the RT Server, and a scheduled import job runs on the server to import the users. The user import Job can be scheduled to run at a specific time.

Upload & Import

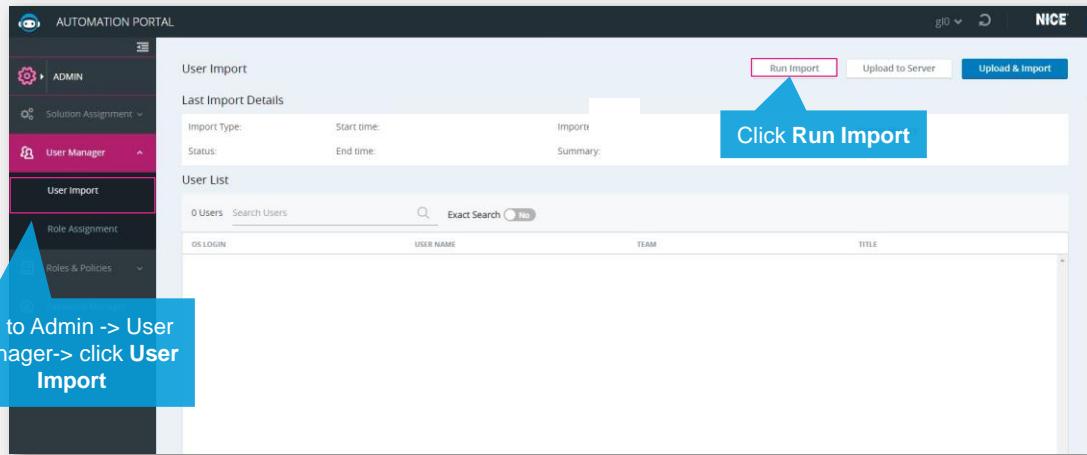
Users in the file are directly uploaded to the RT Server. The new user list is immediately reflected in the portal

NICE®

- Benefits:
 - Ease of use for client
 - Better User Interface
 - Made User Import accessible over Cloud(Automation Studio)

Run Import

- To Run Import:

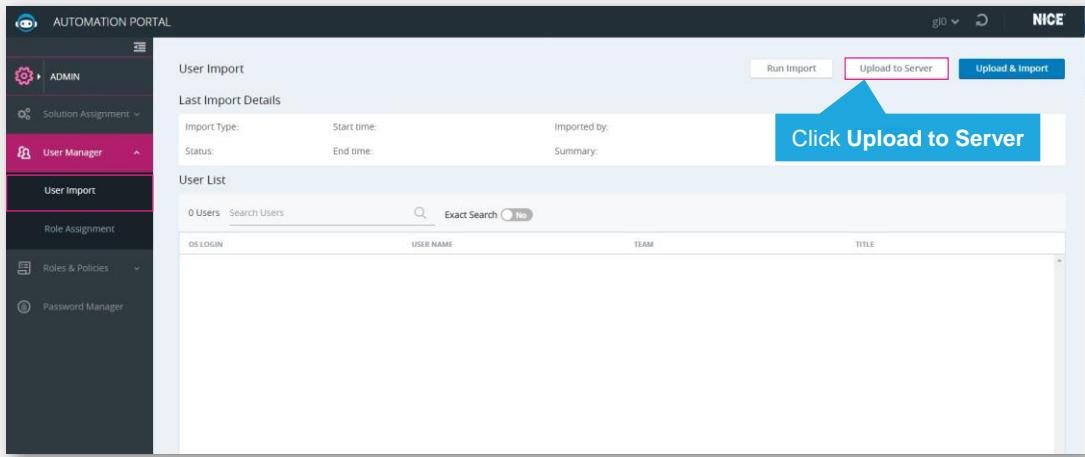


NICE®

- In this method, the User Import job imports the last uploaded file to the server.
- Click Refresh, to view the **User Import summary** of this job.

Upload To Server

- To Upload a user file to the RT Server:

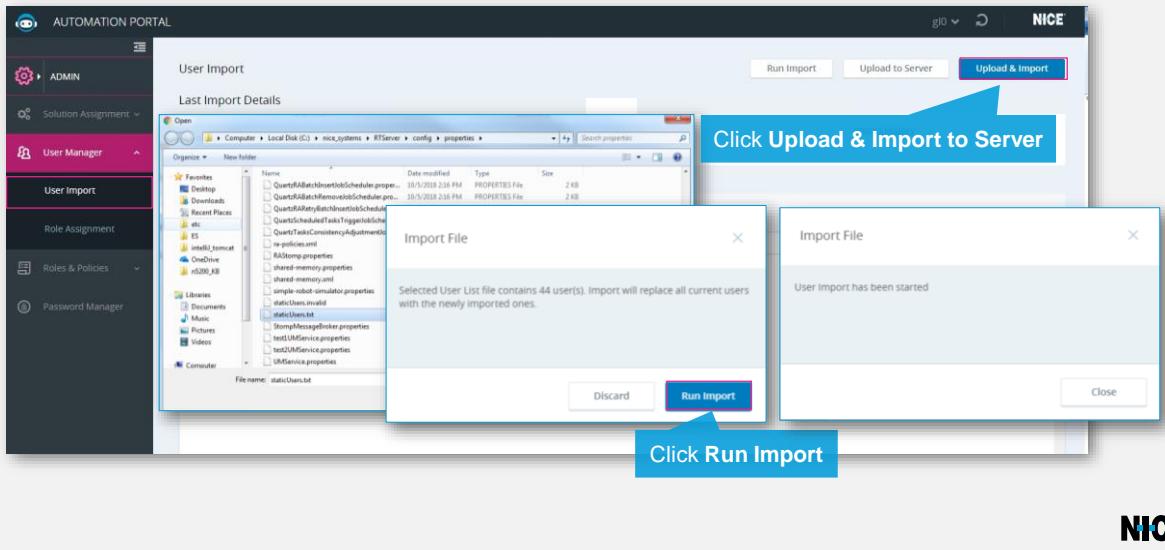


NICE®

- In this method, the file is uploaded to the RT Server, and a scheduled import job runs on the server to import the users.
- The user import Job can be scheduled to run at a specific time. After the job has successfully run on the server, the new user list is available in the Automation Portal.

Upload and Import

- To Upload and Import user file to server:



- In this method, users in the file are directly uploaded to the RT Server.
- The new user list is immediately reflected in the portal.

Import History

- To view a summary of the last import jobs:

The screenshot shows the NICE Automation Portal interface. On the left, there is a sidebar with various administrative options: ADMIN, Solution Assignment, User Manager (selected), User Import (highlighted in pink), Role Assignment, Roles & Policies, and Password Manager. The main area is titled "User Import" and "Last Import Details". It includes fields for "Import Type", "Status", "Start time", and "End time", along with "Run Import", "Upload to Server", and "Upload & Import" buttons. A callout bubble points to the "Import History" button. Below this is a "User List" section with a table titled "Import History (Last 10)". The table has columns: IMPORT TYPE, STATUS, START TIME, COMPLETION TIME, INITIATED BY, and DETAILS. The table lists ten entries, mostly successful full imports, with one failed entry. A blue callout bubble points to the "Import History" link in the table header. The NICE logo is visible in the bottom right corner.

Import History (Last 10)					
IMPORT TYPE	STATUS	START TIME	COMPLETION TIME	INITIATED BY	DETAILS
Full Import	Completed	10/18/2018 4:51:45 PM	10/18/2018 4:17:31 PM (107 sec)	System	Users Imported: 30000, Users Added: ...
Full Import	Completed	10/18/2018 4:12:02 PM	10/18/2018 4:12:06 PM (5 sec)	g0	Users Imported: 100, Users Added: ...
Full Import	Completed	10/18/2018 2:22:37 PM	10/18/2018 2:22:41 PM (5 sec)	g0	Users Imported: 100, Users Added: ...
Full Import	Failed	10/18/2018 2:12:27 PM	10/18/2018 2:12:27 PM (1 sec)	g0	System
Full Import	Completed	10/18/2018 1:57:18 PM	10/18/2018 1:57:23 PM (6 sec)	g0	Users Imported: 100, Users Added: ...
Full Import	Completed	10/18/2018 1:53:02 PM	10/18/2018 1:53:02 PM (1 sec)	g0	Users Imported: 100, Users Added: ...
Full Import	Failed	10/18/2018 1:48:00 PM	10/18/2018 1:48:00 PM (1 sec)	System	StaticUserImporter failed: static us...
Full Import	Failed	10/18/2018 1:45:00 PM	10/18/2018 1:45:00 PM (1 sec)	System	StaticUserImporter failed: static us...
Full Import	Failed	10/18/2018 1:42:00 PM	10/18/2018 1:42:00 PM (1 sec)	System	StaticUserImporter failed: static us...
Full Import	Failed	10/18/2018 1:39:00 PM	10/18/2018 1:39:00 PM (1 sec)	System	StaticUserImporter failed: static us...

Education Services

Role Assignment

The screenshot shows the 'Role Assignment' page within the 'User Manager' section of the Automation Portal. The left sidebar includes options like 'User Import', 'Role Assignment' (which is selected), 'Roles & Policies', and 'Password Manager'. The main area displays a table with columns: DS LOGIN, USER NAME, TEAM, TITLE, and ROLE. A search bar at the top allows for filtering users by name or role. A callout box highlights the 'ROLE' column with the text: 'Assigning Roles to Designer Users to determine which module(s) the user can access'.

NICE®

Assigning Roles to Users

- To assign roles to a user:

1 Go to Admin -> User Manager-> click **Role Assignment**

Type in keywords to search in all the columns

2 Select one or more users to change the role

3 Select the new role, and click **Apply**

NICE®

- The contents in the Roles Assignment page will be visible if you are mapped to the Roles Assignment authorization policy in Open AM.
- You can assign roles to one or more users in the RT Server
- Select Exact Search to see the results that exactly match the keywords.

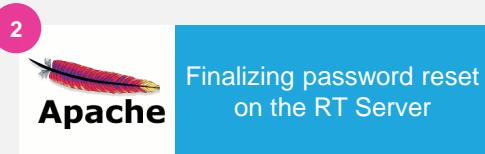
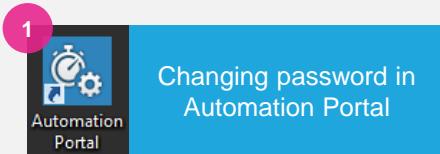
Password Manager

The screenshot shows the Automation Portal (APA) interface. On the left, there's a sidebar with icons for Admin, Solution Assignment, User Manager, Roles & Policies, and Password Manager. The Password Manager icon is highlighted with a dark blue background. The main area displays a grid of icons representing various system components: Operational Database, DataMart, BSF, Total View Import Users, Total View Export Reports, Truststore ActiveMQ, LDAP, and Cognos Admin. A callout box with a pink border and white text is overlaid on the grid, pointing to the first two rows of icons. The text inside the callout box reads: "Changing the passwords of databases, version management systems and FTP connections in the Real-Time Server". To the right of the grid, a text box says: "Select an icon on the left to change its password." The NICE logo is in the bottom right corner.

- Password management for system passwords in APA is done through Vault (a third party tool).
- This method replaces manual password configuration and provides a secure and robust way for system secret management

Changing System Passwords

- You can change the passwords of databases, version management systems and FTP connections in the Real-Time Server from the Portal
- The Password change involves two steps:

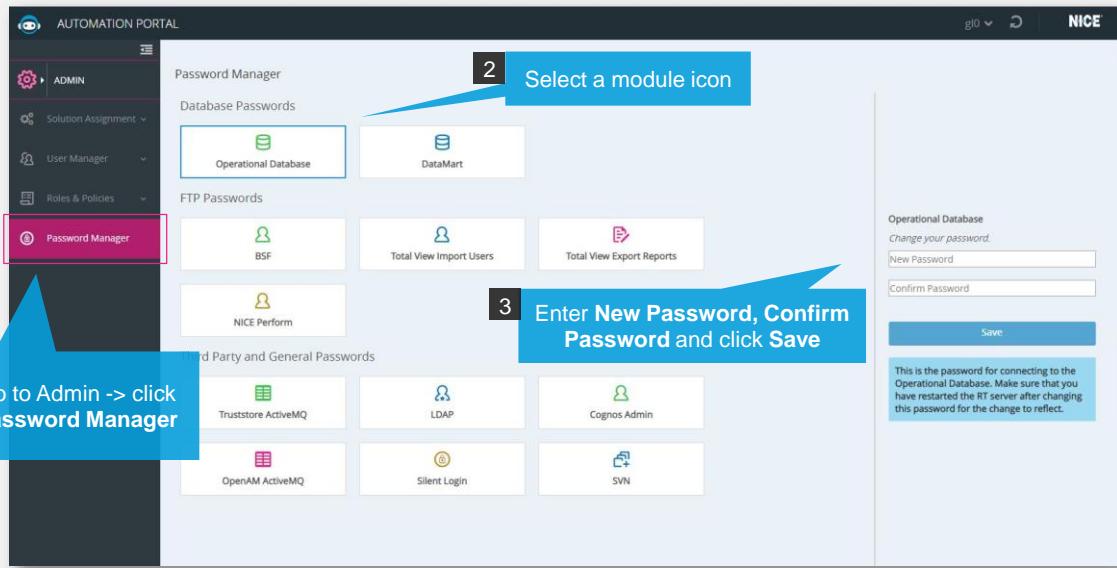


The steps to finalize password change on the RT Server are different for each component
Please refer to the guide for more information

NICE

- For the Operational Database and DataMart passwords, change the password on the RT Server database(before updating it in the portal)
- To change the password, make sure that your log-in ID is mapped to the VaultUpdateActions authorization policy in Open AM by the administrator.
- Important: It is recommended that you change the password during non-business hours as some of the steps involve changing password in the database and restart of the services on the RT Server.

Changing Password in Automation Portal



- Note: Make sure that the password does not have any spaces.

Summary

- Using the Desktop Client Control Room and Dashboard
- Using the Robotic Client Control Rooms and Dashboard
- Scheduling a task for Automation
- Managing Passwords
- Using the Admin Functions

NICE®



Thank You





SOLUTION DEBUGGING



Lesson Objectives

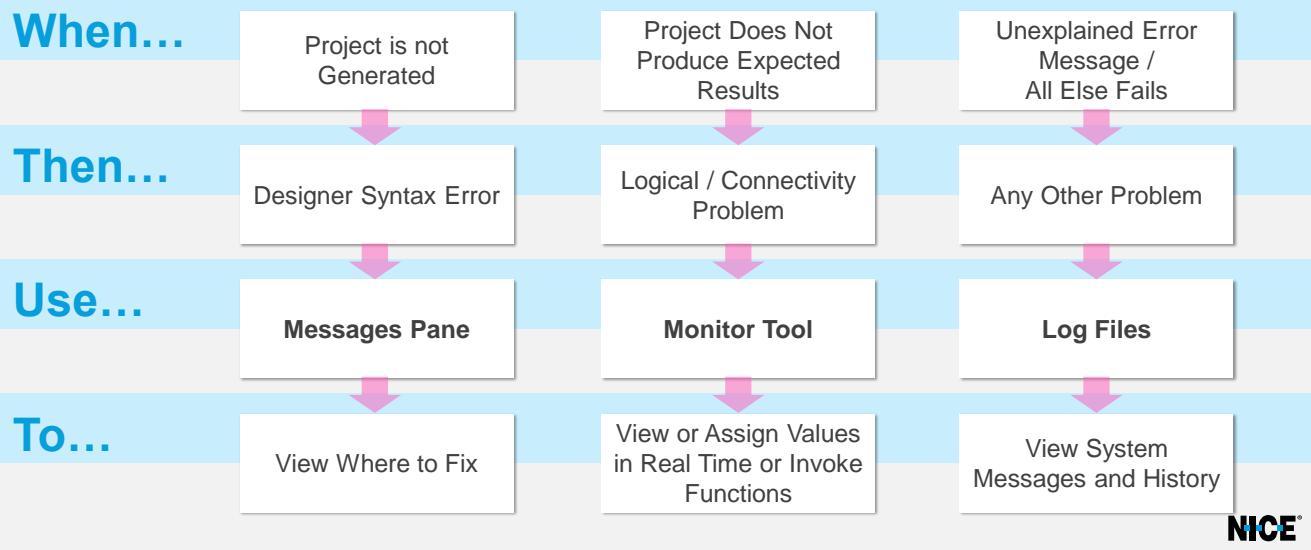
By the end of this lesson you will be able to:

- Debug Solutions
- Add Logging Appenders
- Search/Filter for Logs
- Troubleshoot in the area of Screen Elements
- React to Connectivity Watcher Alerts



Project Debugging Tools

- Designer gives you several tools to debug your Solution:



- When... defines the **symptoms**.
- Then... defines the **actual problem (bug)**.
- Use... defines **which debugging tool to use**.
- To... defines **the debugging tool's usage**.

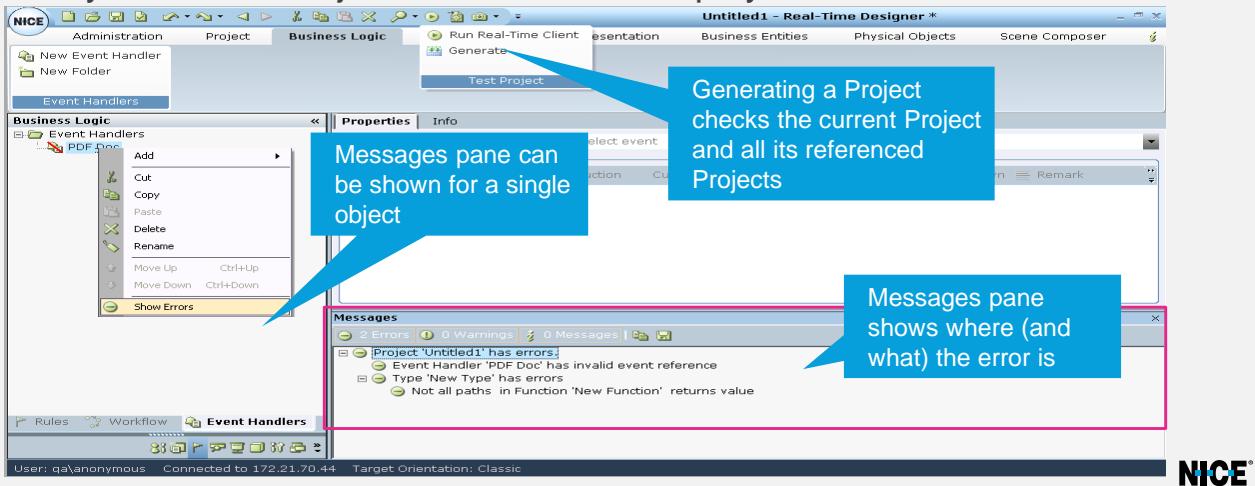
Designer Messages Pane



NICE®

Messages Pane Shows Error Messages

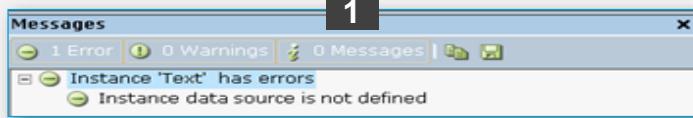
- Designer enables you to validate the correctness of specific objects in your Project and of the entire Project as a whole.
- Only error-free Projects can be run or deployed.



- 2 ways of looking for error messages: either for a single object – if it is red-crossed in the objects tree, you can right click and select Show Errors, or for the whole Project and its referenced Projects, click Generate.
- In the Message pane, click any of the messages to go to the problematic layer.

Common Error Messages

?



For each error message:

- Define the problem
- Offer a solution

NICE®

- **Message 1**

- **The Problem:** an automatically assigned data source was checked in the instance's data source tab, but no data source object has been defined for it.
- **The Solution:** either select a data source or uncheck the auto-assignment box.

- **Message 2**

- **The Problem:** the Function was defined a return parameter of a certain Type, but one of the logical paths in the Function's body does not return anything. Usually happens when using loops ("if-else" for example) and forgetting to return a value in one of the possible outcomes.
- **The Solution:** either define a return value for the problematic path, or remove the return parameter.

Designer Monitor Tool



When to Use the Monitor Tool



Can be Used For...

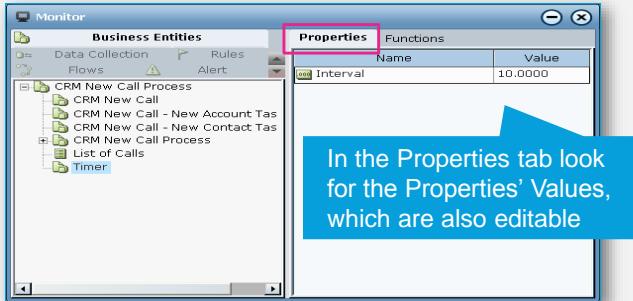
Cannot be Used For...

- **History** Observation
- **Remote** RT Client Testing

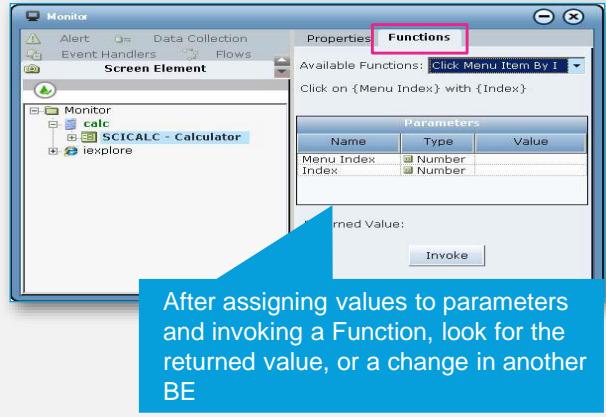
NICE®

How to Use the Monitor Tool

Change Property values to monitor the effect on the RT Client



Invoke Functions to monitor the behavior on the RT Client

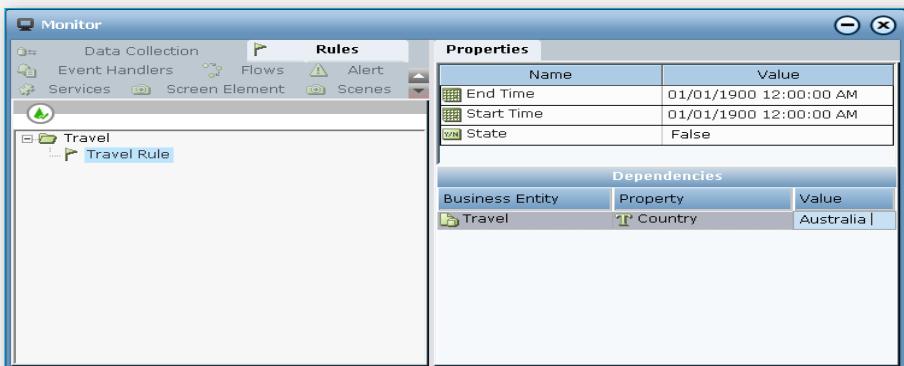


NICE®

- The Monitor will show only directories that relate to that module – for example – only Projects that have Business Entities will appear in the Business Entities window.
- Using the Monitor to observe can answer the following questions:
- Is a certain Screen Element active?
- Was a BE's value modified?

Monitoring Example

Your Solution turns on a Rule when value taken from the screen equals “Australia”. Here’s what the Monitor displays:



The Rule is not turned on when it's supposed to....

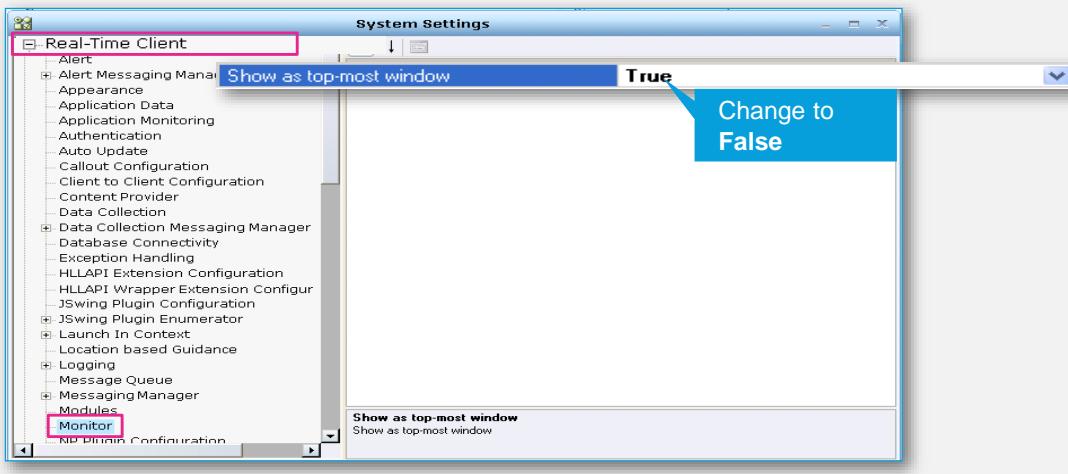
Can you find the problem by viewing the Monitor?

NICE®

- First, identify that the Rule is not turned on: 1. It has not turned Green 2. Its properties have not been changed....
- Second, to debug the problem, look at the value that the Monitor uses as this Rule’s dependencies, you can see the Australia has a space after it.
- Since the Rule’s Condition is not fulfilled (“Australia” does not equal “Australia ”), the Rule is not turned on. We need to either adjust the Condition, or trim the value from spaces before it is saved into the “Travel” Business Entity.

Monitor: Top-Most Window

- For debugging purposes, you may be required to not always have the Monitor as the top-most window:



NICE®

- When this setting is changed to False, the Monitor window will be minimized once started.

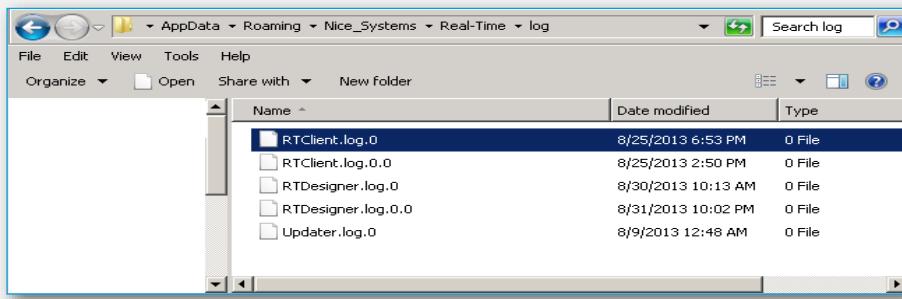
Designer Log Files



NICE®

Log Files Location and Appender Type

- Log files are saved in the installation folder.

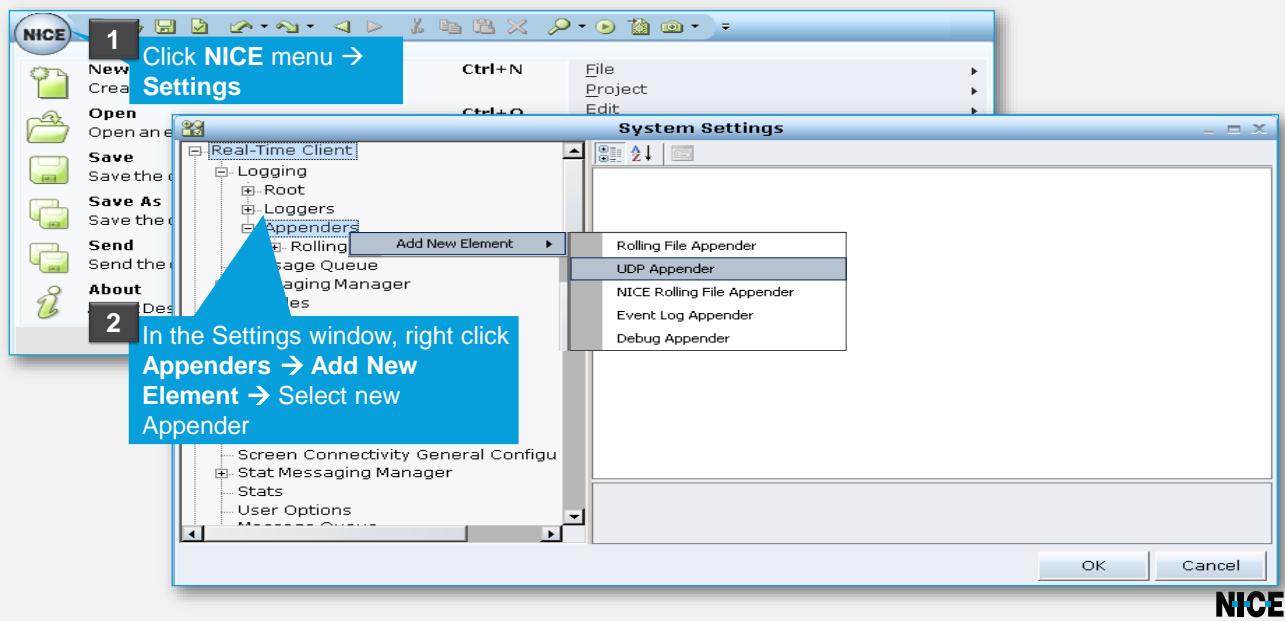


- Rolling File Appender is set by default, producing a simple text file. But other Appenders can be added and configured, such as: UDP, Event log, Debug Appender etc.

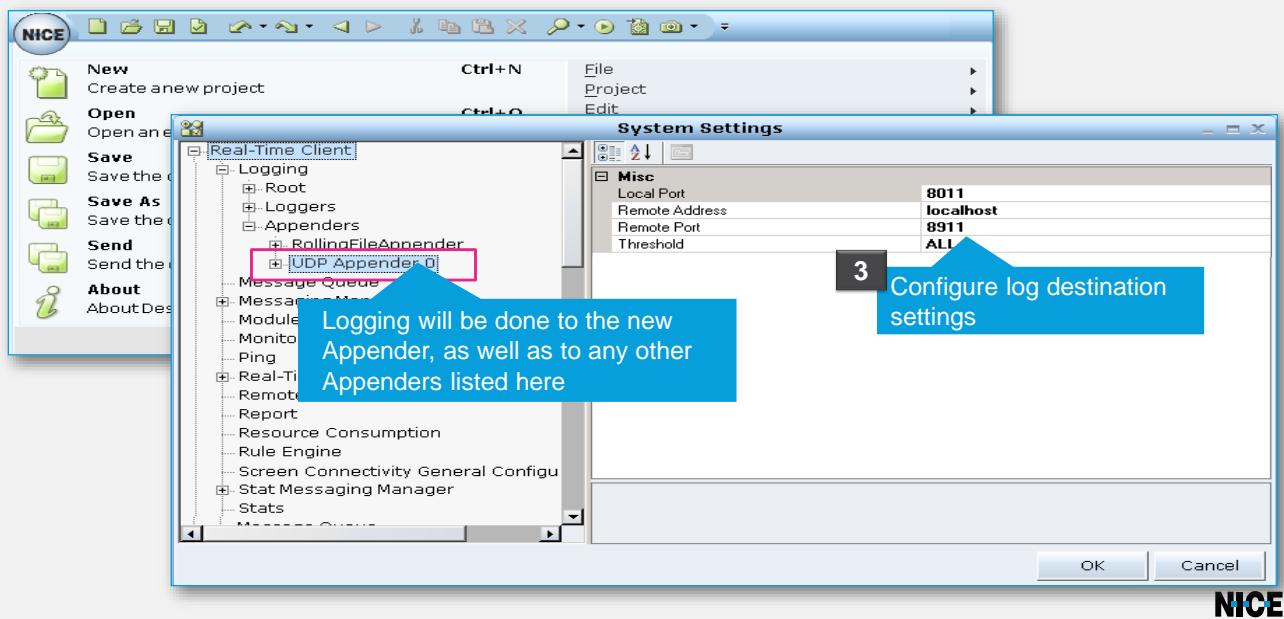
NICE®

- The Installation folder can be either C:\Program Files\NICE Systems\Real-Time Designer or in the AppData folder (depending on what was configured during installation).

How to Add a Logger Appender

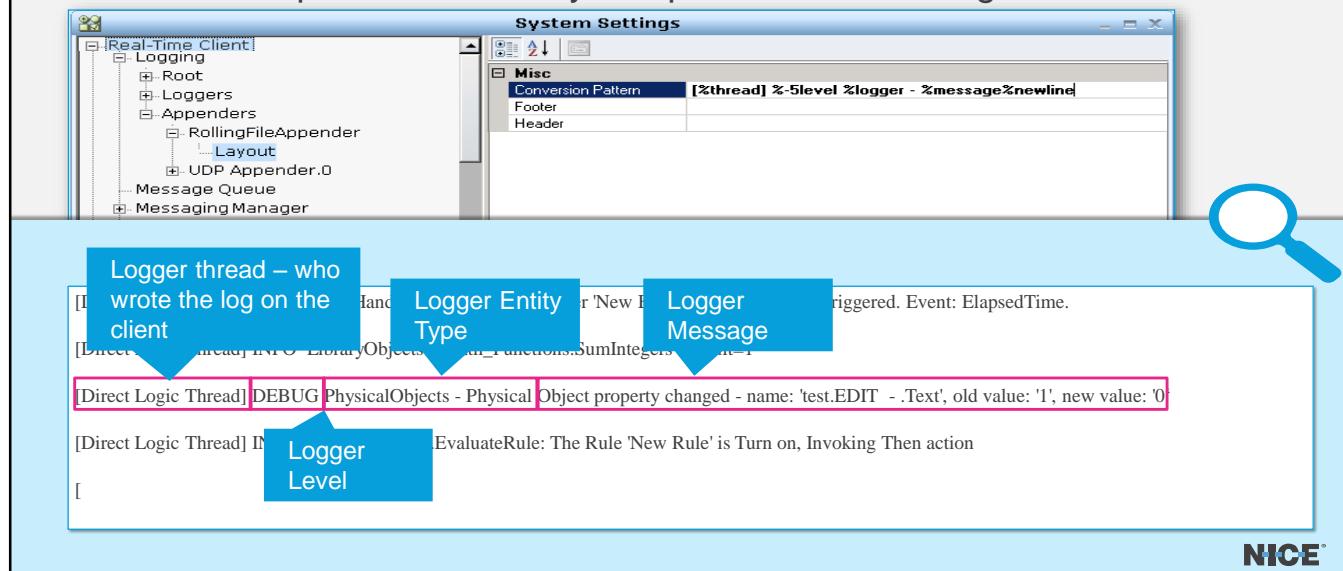


How to Add a Logger Appender



Log Layout

- Conversion pattern under Layout specifies the message structure:



- In the log file there will be also a date and time stamp for every message (not shown here).
- For testing in a multiple Client environment for example, a **%Username** string can be added to the Layout.

Logger Entity Types

- The messaging logging level is configurable by entity type:

Physical Objects	Screen Connectors	Library Objects	Biz Entities	Rules	Event Handlers
POs State & Content	Connector plug-in	Functions State	Modified BE Values	Rules State	EHS State
System		Communication		Developer	
CPU, Memory status etc.		Client-Server		Developing information	

For example, if you have a problem with BE's...

- Increase logging level for Business Entities to view relevant logging messages
- Decrease logging level for all other entities (even turn them off), to avoid unnecessary 'noise'

NICE®

- In order to configure the logging level, you must first select the Entity to configure

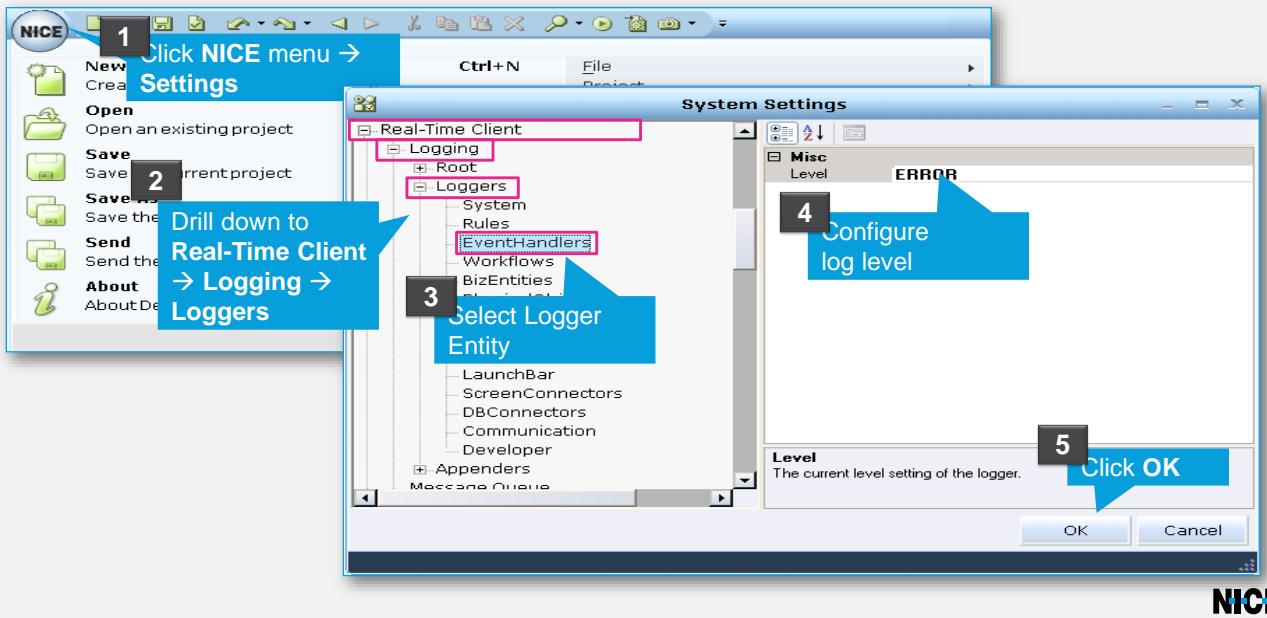
Log Levels

Off	No log data at all
System	System messages only
Fatal	Crucial errors
Error	Only errors
Warning	Warnings and Errors
Info	Info messages (start, end, value changed)
Debug	Errors, warnings and all steps that were performed
All	Errors, warnings, all steps that were performed including extremely detailed messages

NICE®

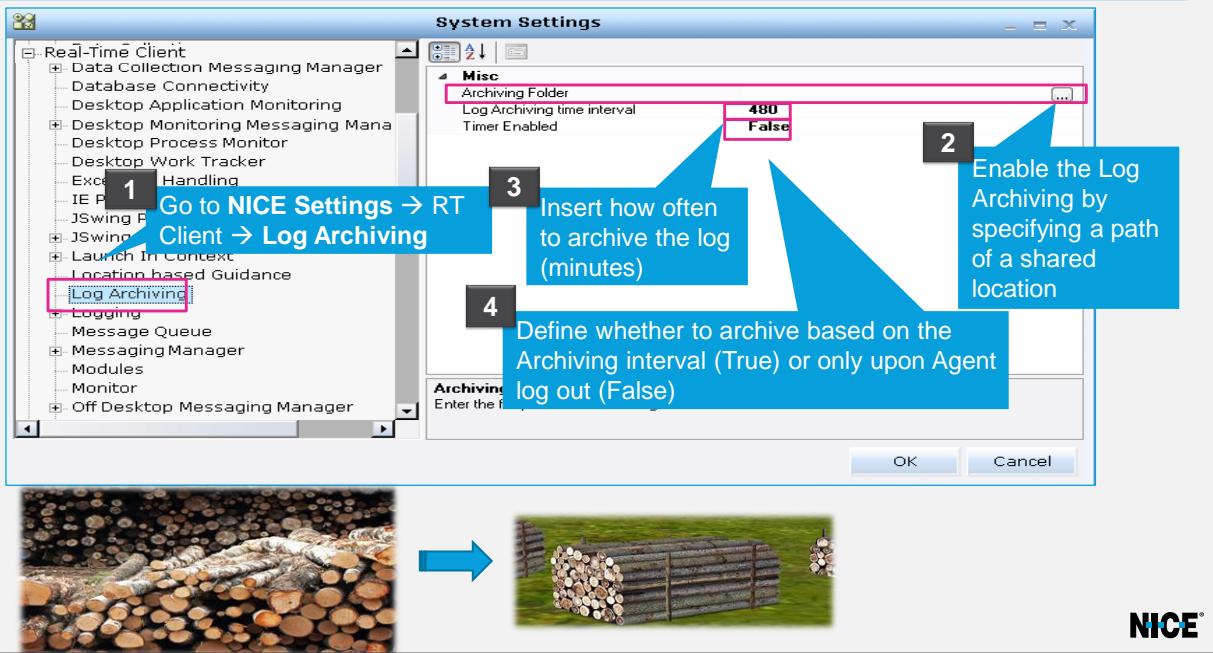
- By default, logging level is Error for all Entities
- For example, when you use a support connector fix, change logging level to All

How to Configure Log Levels



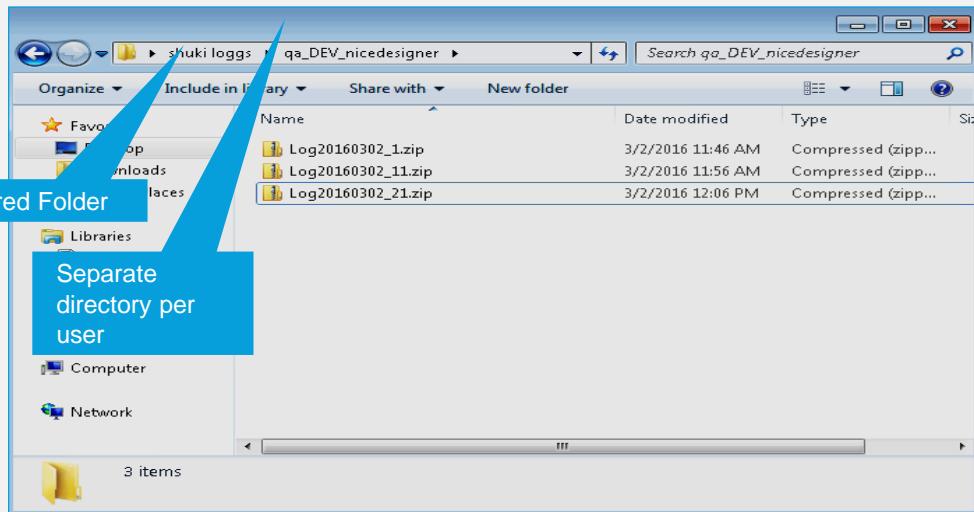
- Logging level can also be configured directly by editing the RTClient.exe.config file, located in the installation folder

RT Client Log Archiving



- The Log Archiving feature allows you to archive log files into a shared folder, which enables the developer to save Real-Time Client logs to a folder and review them whenever necessary.
- This feature is useful in organizations which use Citrix clients, and all application data is deleted once the agent session closes; or in environments with dynamic seating, in which agents move around between workstations.
- The log will be archived as long as you specify an archive path and if the **Timer Enabled** is set to False, the log will only be archived whenever the agent logs out of their workstation.

RT Client Log Archiving

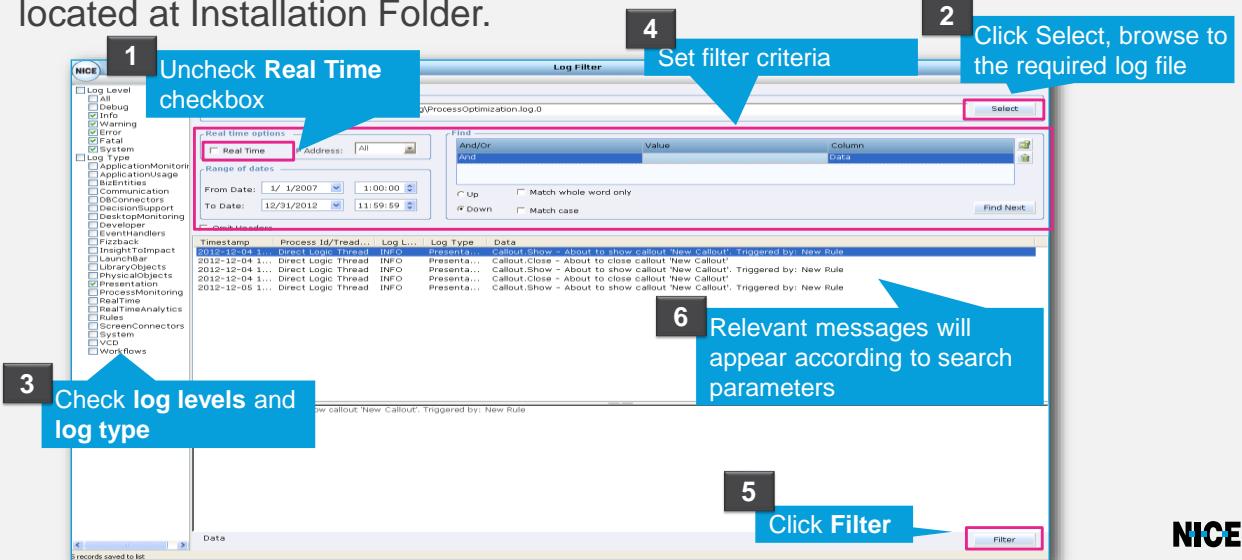


NICE®

- The next time the agent logs out (or if you have set Timer Enabled to True), the archived file will be saved as a ZIP file with year, month, and day of the archive and the index number of the file (for example, if there are several archives a day). The format will look like this: **Log<yyyy><mm><dd>_1.zip**.

Log Filter Tool

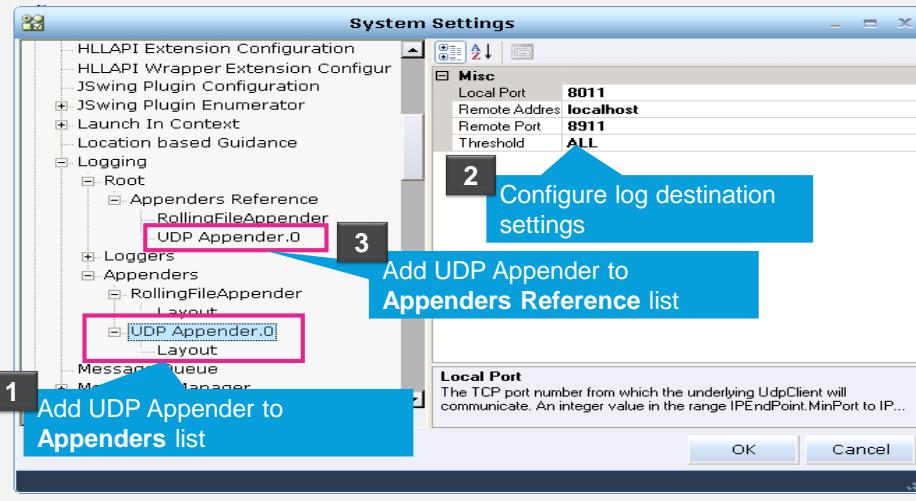
- For searching specific log messages, use the built-in Log Filter tool, located at Installation Folder.



- Log files can also be searched with text editors.
- Log files can be found under the installation folder (either **C:\Program Files\NICE Systems\Real-Time Designer\log** or **%appdata%\Nice_Systems\Real-Time\log**)
- Select one of the Designer logs to view logging during development.
- Select one of the RT Client logs to view logging for the RT Client.

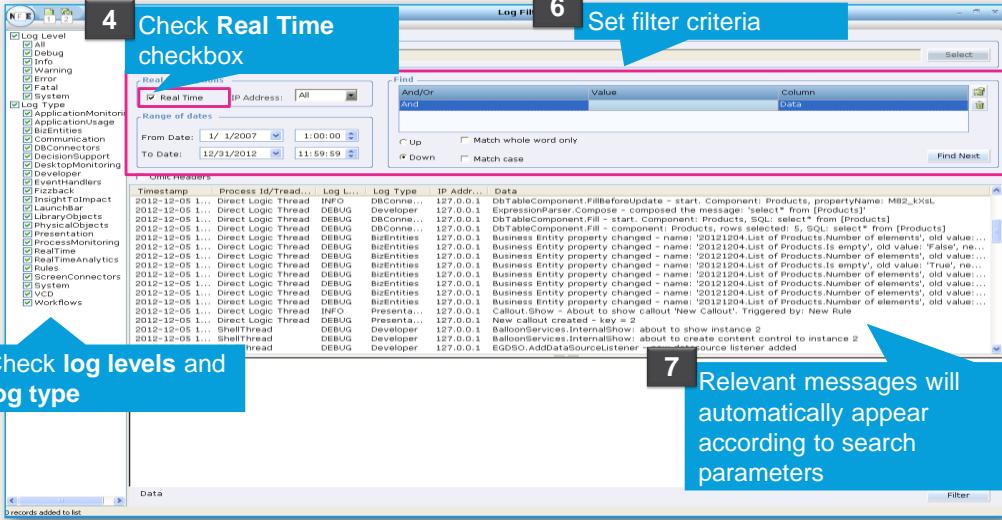
Log Filter Tool with UDP Appender

- Add and configure UDP Appender:



Log Filter Tool with UDP Appender

- Make sure to open the Log Filter tool from the location defined in the UDP Appender settings:



Troubleshooting Screen Elements - Example



How to Troubleshoot Screen Elements

- The process of capturing a Screen Element consists of three primary steps, each of which has a decision point that determines how to proceed



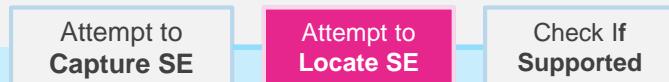
If the Capture process fails...

A message is displayed and the SE is not added to the SE tree as a new branch

Checklist

- Check if the relevant connector is installed and enabled*

How to Troubleshoot Screen Elements



If the Locate process fails...

A message is displayed

Checklist

- Check if the Application in which the SE appears in is open
- Select the top parent in the SE tree and then click the Locate button to locate it
- If the SE is located, then proceed to the next descendant in the tree and repeat

- Work your way down through the tree until you find the SE that has the problem. Go to its Identification tab
- Check SE Type, Relation Types and Self Properties
- Locate Again to test your changes

NICE®

- If the Locate process fails, the goal is to try to understand which of identification parameters of the actual screen element or one of its ancestors is not being identified.
- **SE Type** – if it does not properly describe the SE (for example –a window instead of a button), change its property.
- **Main Relation Type** – If the SE in focus has a parent that is not always displayed on the Agent's screen, change its property.
- **Additional Relation Type** – This by default is None. Try to add more anchoring information so that Designer can find the element.
- **Self Properties** – One of the self-properties for the SE may not be appropriate. For example, it may not be generic enough and may require use of wildcards. This involves testing the SE by typing different values in it to see if one of the Self properties changes.

How to Troubleshoot Screen Elements

Attempt to
Capture SE

Attempt to
Locate SE

Check If
Supported

SE Functionality can fail when the following occur:

- The property is listed in the Properties area, but the Set column does not have a +, which means that a value cannot be assigned to it
- The Function or Event that you want to invoke or capture on this SE is not present
- The Bindable column for a property does not have a +, which indicates that the property is not bindable

Name	Type	Get	Set	Bindable
Bounding Rectangle	Screen Element Rectangle	+		
Enabled	Boolean	+	+	+
Exists	Boolean	+		+
Focused	Boolean	+		+
Process Id	Number	+		+
Visible	Boolean	+	+	+
Window Handle	Number	+		+

Name	Return Type	Name
Activate Context Menu Item	None	Created
Activate Default Action	None	Destroyed
Bring To Front	None	Gained Focus
Click	None	Lost Focus
Do Accessible Default Action	None	Mouse Clicked
Emulate Keyboard	Boolean	Mouse Double Clicked

NICE®

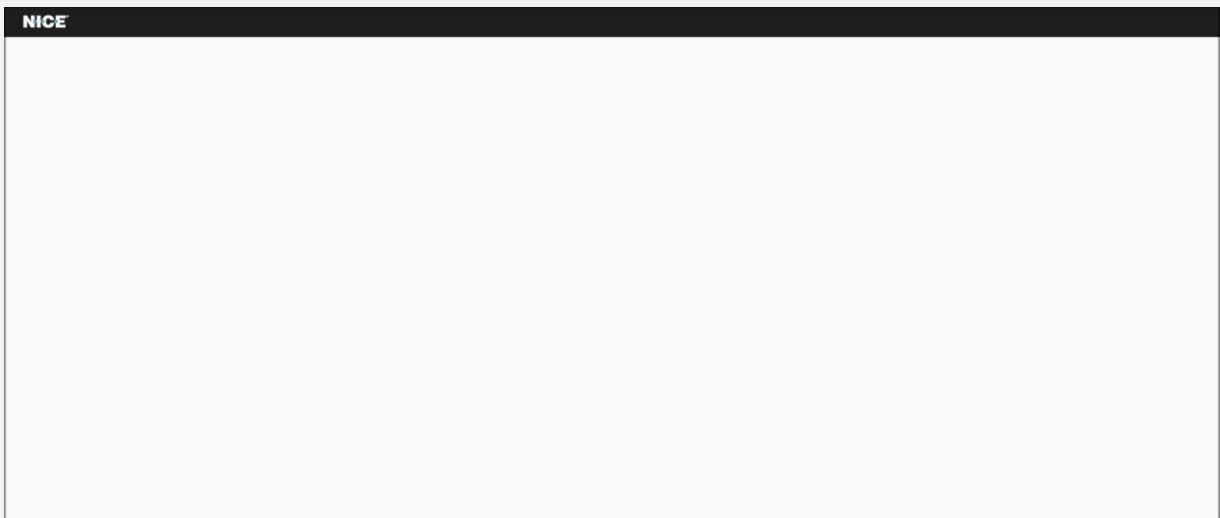
- Supported Functionality for each SE can be viewed at the SE branch at **Functionality** tab.
- If the **Bindable** column is not marked for a property and you do not know what is the specific event that indicates a change in this property, then polling that property constantly does provide a solution, but it may introduce a performance issue. In such situations, for the purpose of screen capturing, the functionality is considered to be not supported. The Bindable setting applies to both Get and Set operations.

Connectivity watcher

29

NICE®

What Is Connectivity Watcher



NICE®

- The **Connectivity Watcher** is part of the Real-Time Solution platform, enabling users to detect connectivity breaks on the client machines while a solution are running.
- The **Connectivity Watcher** automatically alerts when there is a connectivity break, allowing to react by modifying or repairing the project.
- **Main Functionalities:**
 - Recognize connectivity breaks during runtime post deployment (such as GUI changes or application modifications).
 - Use statistic analysis of connectivity to analyze incoming events.
 - Collect and analyze statistical data of the relevant events in order to recognize abnormal behavior of screen components.

Connectivity Watcher – Operating Modes

- The Watcher has 2 operating modes:
- Learning Mode:



- In the *Learning* mode the watcher:
- Gathers information about the solution's Screen Elements on an agent's computer
- Performs statistical analysis on the data collected
- A meta data file is stored in the generated Solution's folder named: "*Connectivity_Watcher.data*"
- A data statistics file is stored in the AppData folder named: {user's name}_dproj_{solution's name}.statistics
- An event is raised when the initial learning phase is completed for the implementer to react to, while the learning continues to run as long as the solution is working.

Connectivity Watcher – Operating Modes

- The Watcher has 2 operating modes:
- Learning Mode:
- Run-Time Mode:



NICE®

When the initial Learning mode completes, i.e. each Screen Element was created and destroyed 120 times, the **Learning Finished** event is raised for each Screen Element in the solution.

During this mode the watcher:

- Utilizes the data compiled when the algorithm was running in Learning Mode to track ongoing screen activity
- Verifies the Screen Elements' activity is within the average distribution of screen events

Connectivity Watcher Events

- The Connectivity Watcher raises 3 events:



Initial learning mode is over and run-time mode turns on



A Screen Element cannot be detected within the average distribution

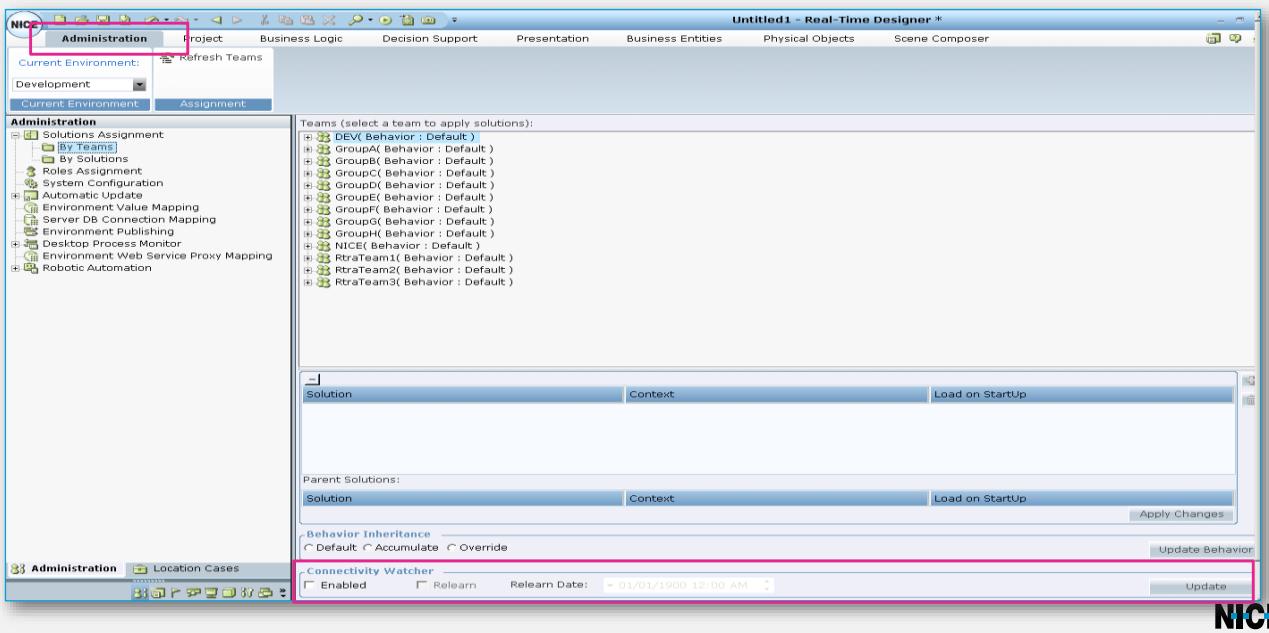


A Screen Element cannot be detected within the maximum distribution

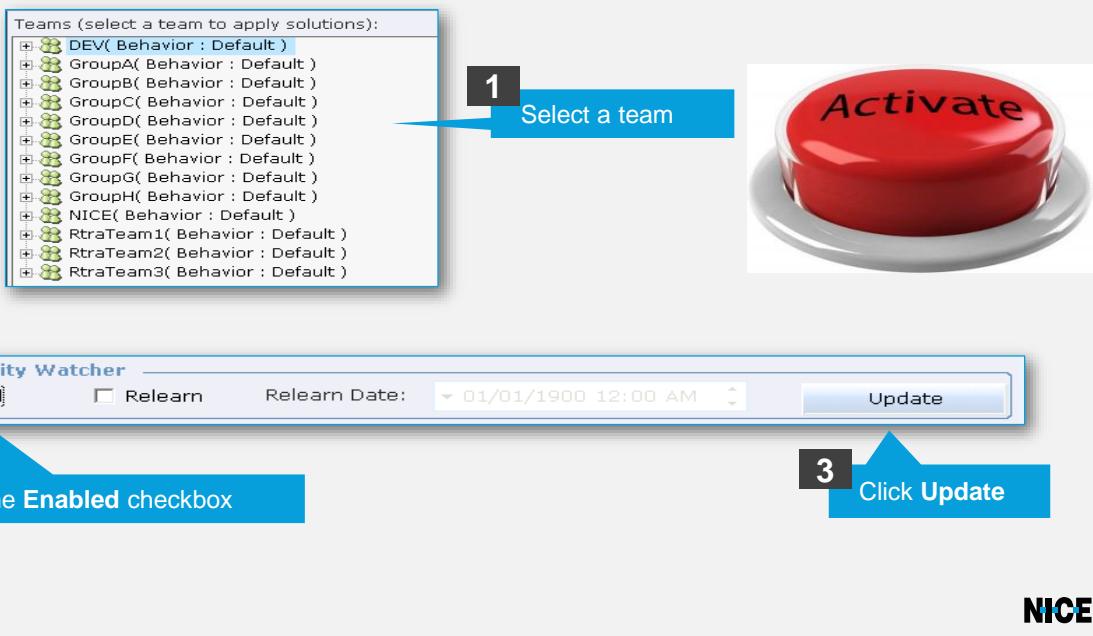
NICE®

- If a connectivity break is detected, the Connectivity Watcher issues the following events:
 - Warning event** - A warning event is displayed when a Screen Element cannot be detected on the agent's computer within the average distribution.
 - Exception event** - An exception event is shown in addition to the **Warning event** when there is no Screen Element event sent at all within the maximum time distribution.

Connectivity Watcher – Menu

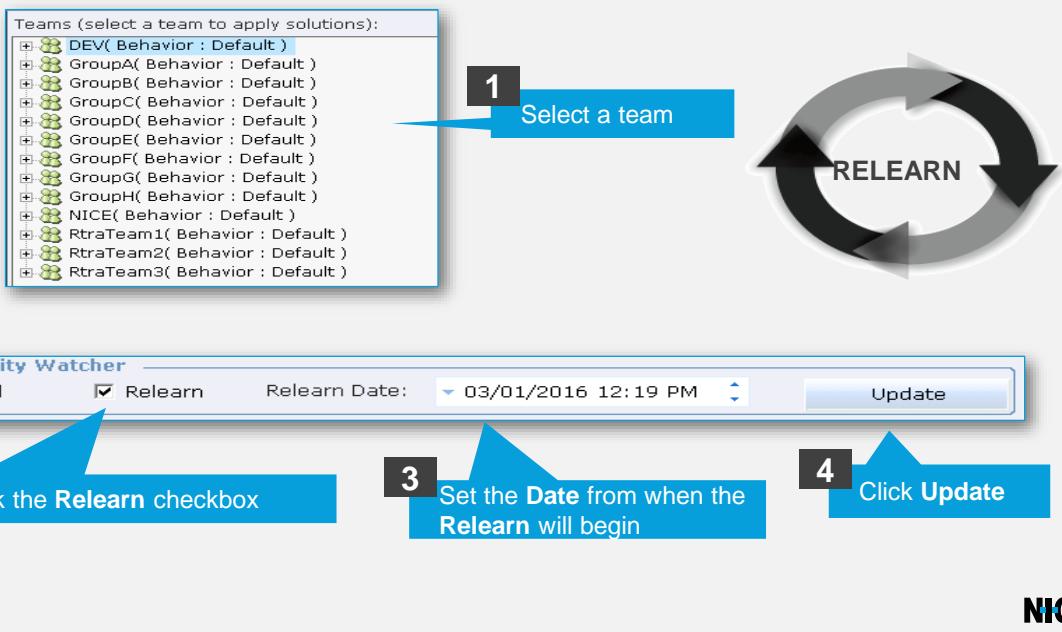


Activating the Connectivity Watcher



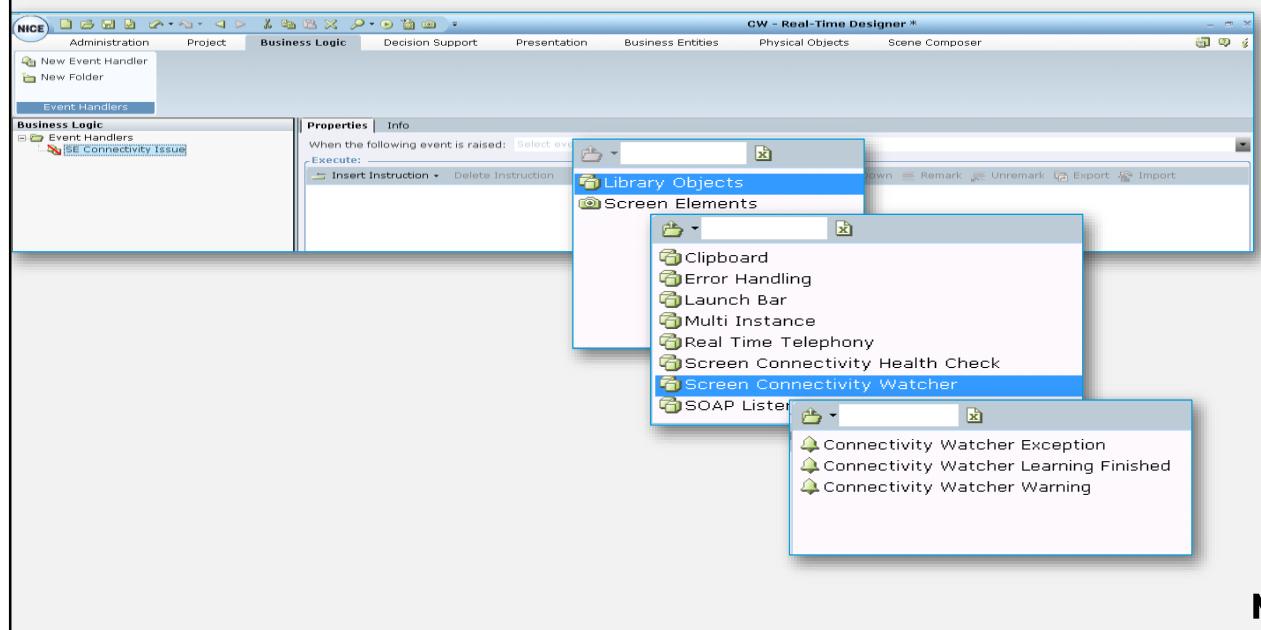
- Since the Connectivity Watcher is a tool which analyzes Screen Elements in a specific solution assigned to a specific team, you must first select a team to which the Watcher will be enabled.

Activating the Relearn Mode



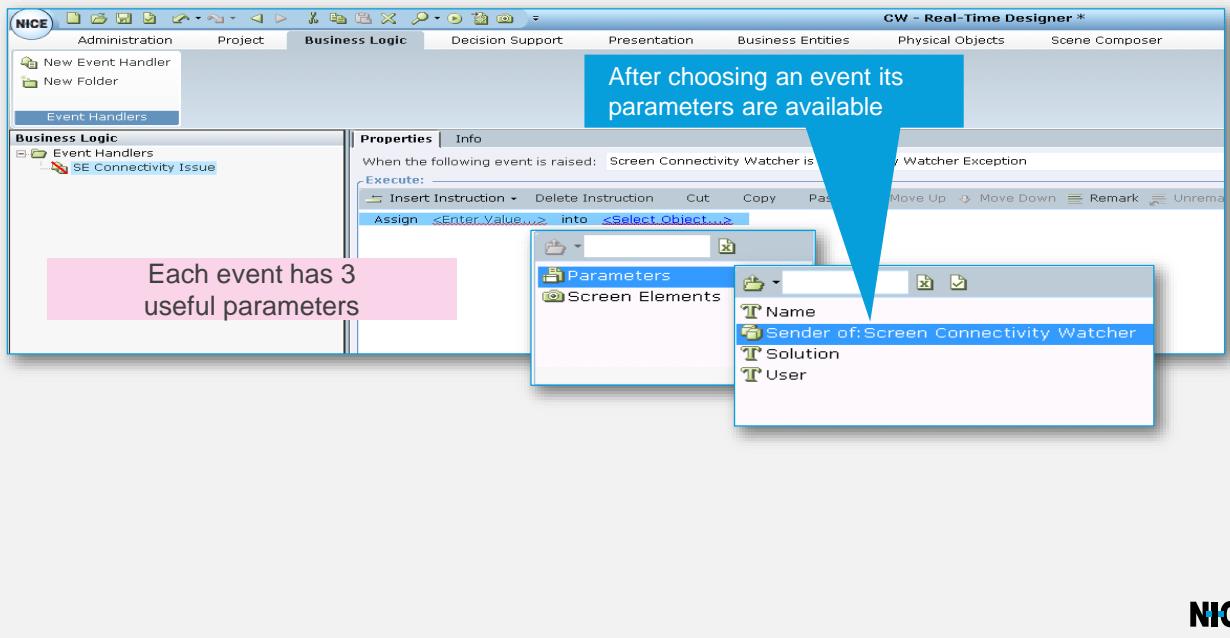
- In some cases a new learning period will be required, such as when an application is changed – affecting the captured Screen Element.

Connectivity Watcher – Handling Events



NICE®

Connectivity Watcher – Handling Events

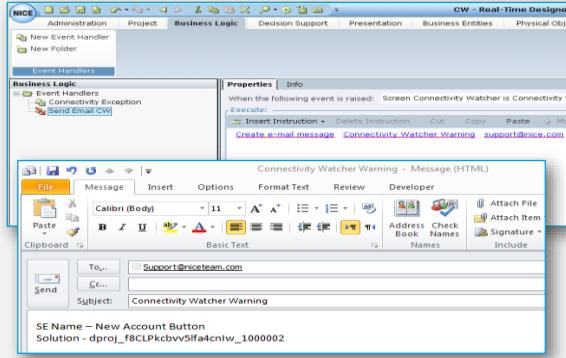


Each event has 4 parameters:

- Name – The name of the Screen Element which raised the event
- Sender of – The sender of the event (Connectivity Watcher)
- Solution – The name of the solution containing the Screen Element
- User – The name of the user who is running the solution

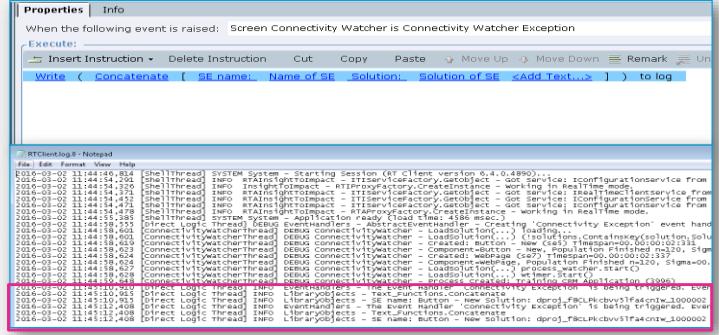
Connectivity Watcher – Common Uses

- Send an Email



Best Practice
Combine Write to Log function with the Log Archiving feature as a complete operation

- Write to Log file



NICE®

Summary

- Debugging Solutions
- Adding Logging Appenders
- Searching/Filtering for Logs
- Troubleshooting in the area of Screen Elements
- Reacting to Connectivity Watcher Alerts

NICE®

- Designer gives you several tools to debug your Solution
 - Messages Pane
 - Monitor Tool
 - Log Files
-
- Messages Pane helps you to find syntax problems – before running the Client.
-
- Monitor displays values and allows you to set values in real time.
-
- Use the Log files when all else fails or to view history.
-
- You can add a Logging Appender to write logs to a different target.



Thank You



NICE®

DECISION SUPPORT: DATA COLLECTION



Lesson Objectives

By the end of this lesson you will be able to:

- Describe the goals of Decision Support features
- Define BEs as collectable
- Create and define Data Collection objects
- Use Data Collection Functions in the Business Logic Layer
- Test Data Collection logic in the Monitor



Decision Support

- Defined in Business Logic Layer
- Collects information from defined sources
- Investigates collected information for reporting and improved guidance and assistance

Business Logic

Decision Support



Data Collection

Business Entities

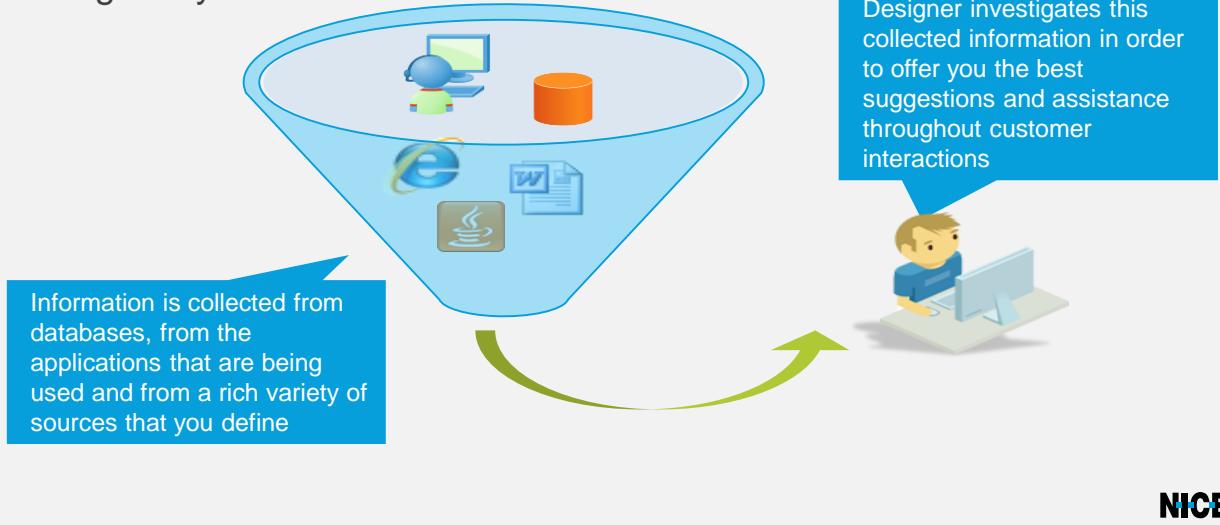
Physical Objects



- Designer can also investigate the behavior of the Agents, bottlenecks in customer handling, most-used parts of applications and where mistakes are typically made. For example: Average Handling Time, Conversion Rate.

Decision Support

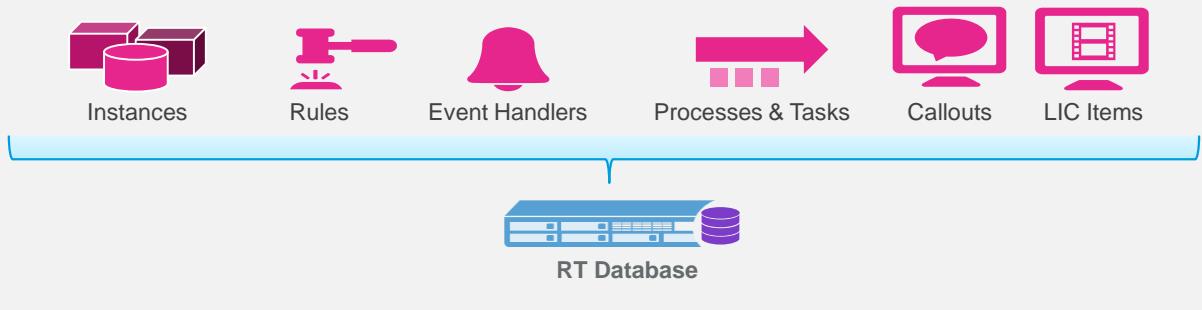
- Designer provides insight and Decision Support based on its data mining analytics.



- Designer also investigates the behavior of the Agents, bottlenecks in customer handling, most-used parts of applications and where mistakes are typically made. For example: Collect data about the interaction, Collect Time spent in each workflow step, Average Handling Time KPI, Conversion Rate KPI

What is Data Collection?

- Data Collection feature enables specifying the data to collect and store in the RT database during runtime
- The idea is simple: track Designer entities so you can learn about agent activity and interactions with the RT Client



- A Data Collection enables you to collect information about the interactions between an Agent and a customer and various kinds of profiling information about the customers themselves, such as the activities that they typically perform.
- This is done by tracking the activities performed in the RT system in order to extract from it information that has relevant business implications.

How to Use Data Collection



- Define future behavior according to the data collected
- Collect data from previous interactions to improve others



- Complete data that is not saved to regular DB



- Select data collected to be presented in reports

NICE®

- RT creates a predictive model that modifies rules to affect RT's behavior and thus the Agents' interaction with the customers. For example, refrain from offering the same product to a client that has refused it in the past. Its self-learning features do not require human interaction and independently modify RT's behavior for the company's benefit.
- Example queries: Recent offers made to the customer, success rate of various business actions, the reason the customer refused an offer and so on.

What Data can be Collected?

- Data can be collected on any RT Entity, for example:

 Rules	 Event Handlers	 Callouts
Rule Duration	Event Handler Start Date	Callout Position
Rule Instances	Event Handler Instances	Callout Shown Duration
More...	More...	More...



- Designer enables you to define for which RT elements information is collected, such as Business Entities, Rules, Event Handlers etc.
- RT collects a variety of information about each element that you select.

Collecting Data From Business Entities

- In contrast to all other RT Entities, which their collectable data is static (i.e. their properties are known and do not change), Business Entities are dynamic.
- They are built each time you publish a solution to the server, and are based on the BE Types defined for collection in that Solution

 **NOTE**
Since Properties' values are dynamic, they must be checked as collectable

Properties	Info
Author:	niceadmin
Created:	1/12/2012 3:42:59 PM
Modified By:	niceadmin
Modified:	1/12/2012 3:55:00 PM
<input type="checkbox"/> Public	
<input checked="" type="checkbox"/> Collectable	
Data Usage In Reports:	Unknown
Data Usage Function:	None

NICE®

- The *Collectable* checkbox is available at the info tab of every Business Entity **Type**, under the *Public* checkbox.

Business Entities Example

- Each **BE Type** has a corresponding table; Its **Properties** are represented as table **Columns**.



When a **BE Instance** of that Type is constructed, property **values** are represented as table **rows**.

The diagram shows three separate instances of the Offer BE type, each represented by a blue box with a pink cube icon above it. The first instance is labeled "Current Offer" and contains one row of data: Client ID 1334756, Name David S., and Price 10,000. The second instance is labeled "Pending Offer" and contains one row of data: Client ID 4064332, Name Tom M., and Price 12,000. The third instance is labeled "Rejected Offer" and contains one row of data: Client ID 6112002, Name Sandy I., and Price 15,000.

NICE®

- The Offer branch contains Properties, each of which corresponds to a column in a database table. Thus, each column represents a Business Entity Property.
- These columns are created dynamically in the database according to the Properties that are defined for collection in the Project where the Type is declared. Each row in the table represents a specific offer Instance.

Primitive BE's Data Usage

- For primitive BE properties, data usage is defined to specify how their data will be used in data collection reports.
- There are 4 optional usage definitions:

Unknown	Identifier	Fact	Attribute
Default, no indication to what the data represents	Data represents the unique ID	Data represents an amount	Data represents a segment
Examples:	SSN, customer ID etc.	Balance, Age etc.	Gender, City etc.

Facts can be aggregated by function: average, minimum, median etc.

Attributes can be used to group by...



Data Collection Configuration - Workflow

- In general, there are 3 steps to configure data collection:



- However, since Business Entities are dynamic, there is an additional preparation step for them:



NICE®

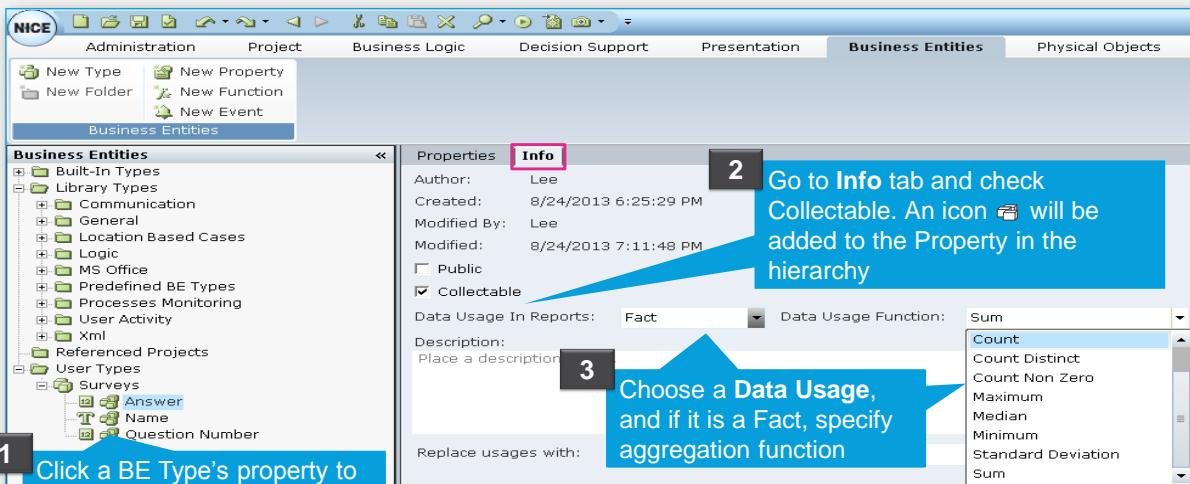
- This step is done only if there are BE to collect data on.
- Only once the Property is defined as collectible in the *Business Entities* layer, it will appear available to be selected in the *Data Collection* tab, where you can specify that RT starts collecting it.
- When you click on a Type, its child can be marked as collectable.

How to Define Business Entities as Collectable



NICE®

How to Define BE Properties as Collectable



1 Click a BE Type's property to collect data on

Define BE as collectable

Create a DC Object

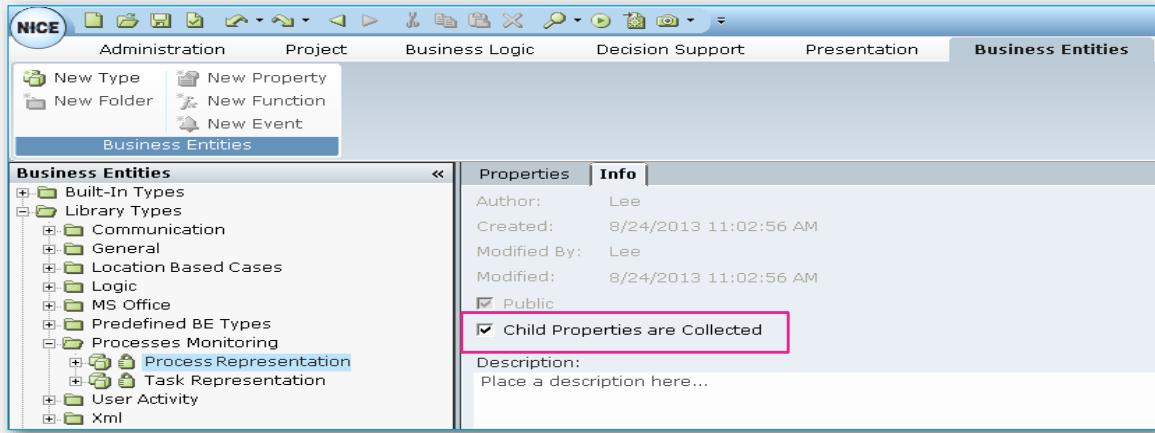
Invoke DC Functions

Test DC

NICE®

Process Monitoring Business Entities

- Process and Task representation BE's are collectible by default:



LIVE DEMO

Defining BE Data as Collectable

- Create a Boolean Instance called 'Trigger' and initialize it to FALSE
- Create a Type 'Customer' with the following properties:
 - Age (Number)
 - Gender (Text)
- Mark these properties as 'Collectable':
 - Age: Fact (Average)
 - Gender: Attribute
- Create an Instance of 'Customer'

NICE

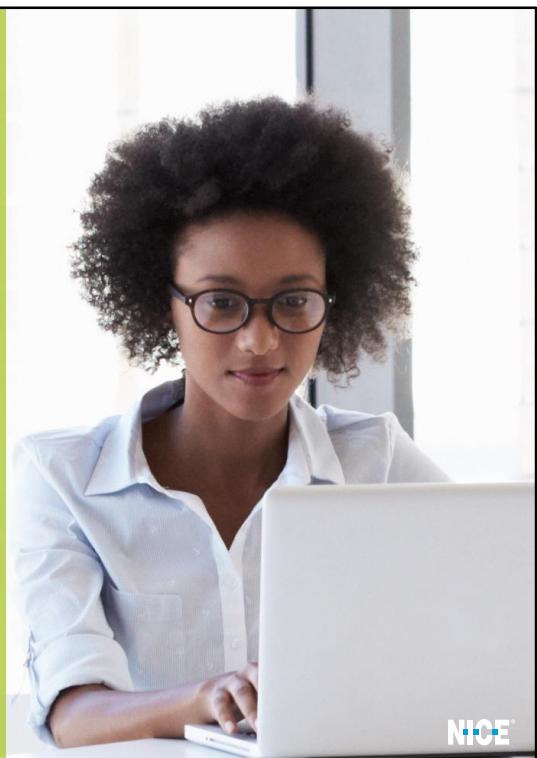
DO IT YOURSELF

Defining BE Data as Collectable

- Create a Boolean Instance called 'Trigger' and initialize it to FALSE
- Create a Type 'Customer' with the following properties:
 - Age (Number)
 - Gender (Text)
- Mark these properties as 'Collectable':
 - Age: Fact (Average)
 - Gender: Attribute
- Create an Instance of 'Customer'

NICE

How to Create Data Collection Objects



When to Create Separate Data Collection Objects

- Different collectable processes use the same Entities



NICE®

- Product Name (Business Entity Property) is used both for collecting data during a sale process AND for collecting data about products that were cancelled.
- The number of processes that use this entity = the number of Data Collections to create.
- An example for **process A contains process B**: Process B collects data about opening an account. Process A collect data of selling a product, which requires opening an account for new customers.

When to Create Separate Data Collection Objects

- A process that may have more than one instance in a single interaction



NICE®

- Offers of different products may occur in a single interaction.
- The number of occurrences we would like to collect = the number of Data Collections to create.

How to Create a Data Collection Object

The screenshot shows the NICE software interface with the 'Decision Support' tab selected. A 'Data Collection' window is open, displaying a tree view of 'Collectable objects' and a list of 'Selected objects'. The 'Selected objects' list includes items like 'Number of surveys', 'text', 'Survey Complete', etc. A pink box highlights the 'Data Collection' tab in the bottom navigation bar.

1 Go to Decision Support → Data Collection tab

2 Click New Data Collection

3 Give DC object a meaningful name

4 Click entity type to drill down the hierarchy and check the entities you wish to collect data for

Selected entities will appear here

Define BE as collectable → Create a DC Object → Invoke DC Functions → Test DC

NICE®

- Each time you check the checkbox of a branch, it appears in the **Selected objects** area at the bottom of the window.

LIVE DEMO

Creating a Data Collection Object

- Create a Data Collection Object called ‘Customer Data Collection’
- Define only the ‘Customer’ Business Entity to be collected

NICE®

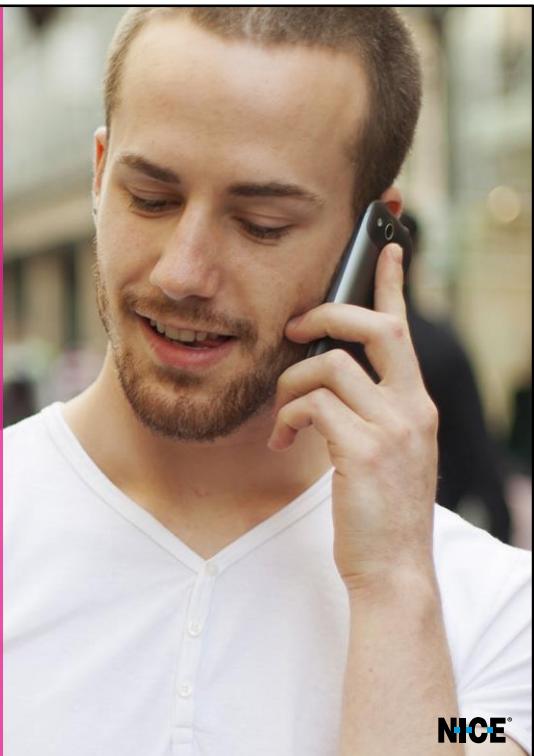
DO IT YOURSELF

Creating a Data Collection Object

- Create a Data Collection Object called ‘Customer Data Collection’
- Define only the ‘Customer’ Business Entity to be collected

NICE®

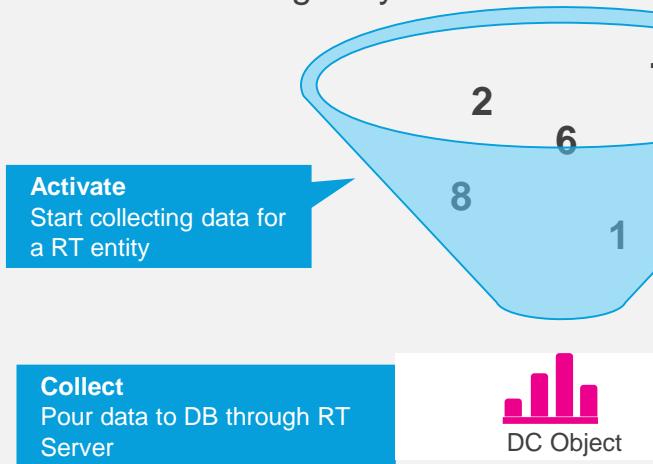
How to Activate and Collect Data



NICE®

Data Collection Functions

- In order to control Data Collection, two Functions are used in the Action Editor of the Business Logic layer:

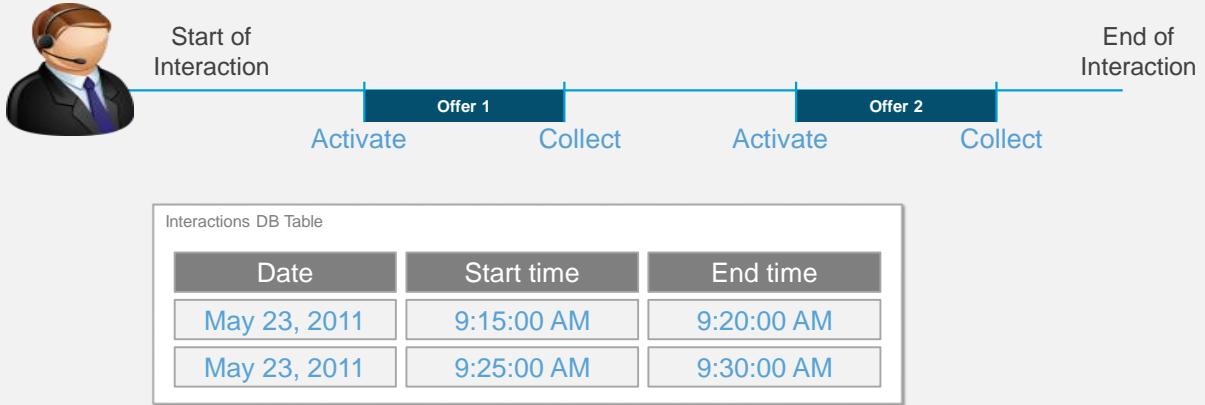


NICE®

- Each time you invoke the Activate and Collect Functions for a specific Data Collection activity, its data is collected as a transaction in the database.

Data Collection Functions

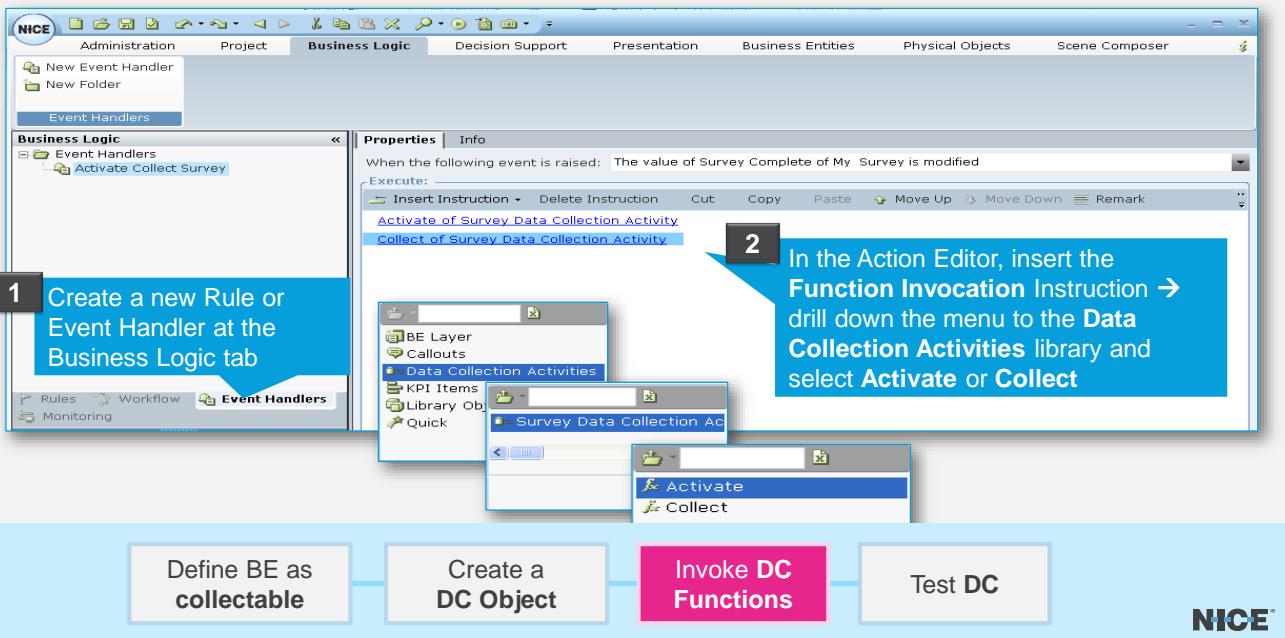
- Each start collecting (Activate) and stop collecting (Collect) pair constitutes an instance of one interaction



NICE®

- Interactions are also an Entity in the Designer, for which data is collected according to the DC Functions “Activate” and “Collect”. If, for example, in one interaction we have 2 different DC processes, then we will have 2 instances for that interaction.

How to Call Data Collection Functions



- DC Functions can be created in the Action Editor of every Business Logic Layer object.

LIVE DEMO

Calling Data Collection Functions

- Create a Rule called 'Activate DC', which:
 - Turns on when the value of 'Trigger' is set to TRUE
 - Runs the 'Activate' function when Turned On
 - Runs the 'Collect' function when Turned Off

NICE®

DO IT YOURSELF

Calling Data Collection Functions

- Create a Rule called 'Activate DC', which:
 - Turns on when the value of 'Trigger' is set to TRUE
 - Runs the 'Activate' function when Turned On
 - Runs the 'Collect' function when Turned Off

NICE®

How to Test Data Collection



How to Test DC Objects

The screenshot shows the Data Collection Monitor interface. On the left, there's a tree view with 'CRM New Call Process' expanded, showing 'CRM New Call DC'. On the right, there are two windows:

- Properties Window:** Shows a table with two rows:

Name	Value
Active	True
Last Collected Time	08/25/2013 09:10:18

A callout box points to the 'Active' row with the text: "The **Active** property will test the **Activate DC** Function. The **Last Collected Time** will test the **Collect DC** Function".
- Functions Window:** Shows a list of available functions: "Activate", "Collect", and "Initiates the data co...". The "Activate" function is selected. A callout box points to the "Parameters" section with the text: "The functions tab allows us to invoke the DC functions at will". Below the parameters is a button labeled "Invoke".

Below the monitor windows, a flowchart provides a step-by-step guide:

```
graph LR; A[Define BE as collectable] --> B[Create a DC Object]; B --> C[Invoke DC Functions]; C --> D[Test DC]
```

NICE®

- It is highly recommended that before publishing, you monitor the Rules or Event Handlers that trigger the DC Functions, make sure they “go on” when expected.
- In order to view real DC Entities values, query the database or run a Report.

LIVE DEMO

Testing Data Collection

- Run the RT Client and open the Monitor Application
- Observe the status of the Data Collection object
- Change the 'Trigger' value to TRUE
- Verify that the Data Collection object has become Active
- Change the 'Trigger' value to FALSE
- Verify that the Data Collection object has become Inactive, and that the 'Last Collected Time' has been updated

NICE®

DO IT YOURSELF

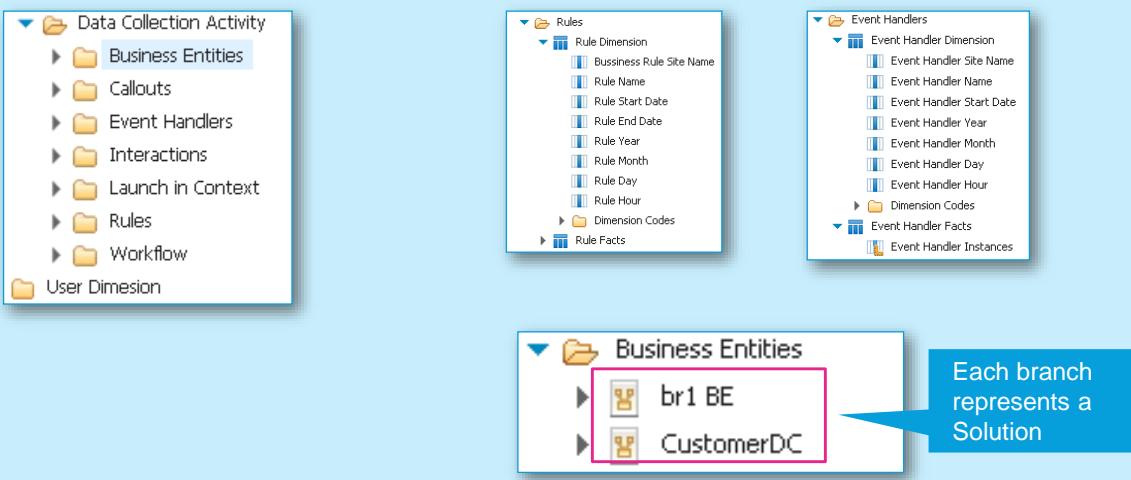
Testing Data Collection

- Run the RT Client and open the Monitor Application
- Observe the status of the Data Collection object
- Change the 'Trigger' value to TRUE
- Verify that the Data Collection object has become Active
- Change the 'Trigger' value to FALSE
- Verify that the Data Collection object has become Inactive, and that the 'Last Collected Time' has been updated

NICE®

Data Collection Reports

- The Data Collection Activity data model represents the various objects for which RT collects data, such as Rules, Event Handlers and so on



NICE®

Summary

- The goals of Decision Support features
- Defining BEs as collectable
- Creating and defining Data Collection objects
- Using Data Collection Functions at the Business Logic Layer
- Testing Data Collection logic in the Monitoring

NICE®

- Designer provides insight and Decision Support based on its data mining analytics.
- One of these features are Data Collection objects, which enable specifying the data that RT should collect and store in the RT database during runtime.
- This data can then be presented in runtime, queried for or displayed in reports.
- The following cases require creating more than one DC object:
 - Different collectable processes use the same Entities.
 - Same process may happen more than once in a single interaction.



Thank You



NICE®

REPORTS



Lesson Objectives

By the end of this lesson you will be able to:

- Create a Report with static and dynamic Data Models
- Produce Robotic Automation Reports
- Produce Server Dashboard Reports





Notes from Demo

DO IT YOURSELF

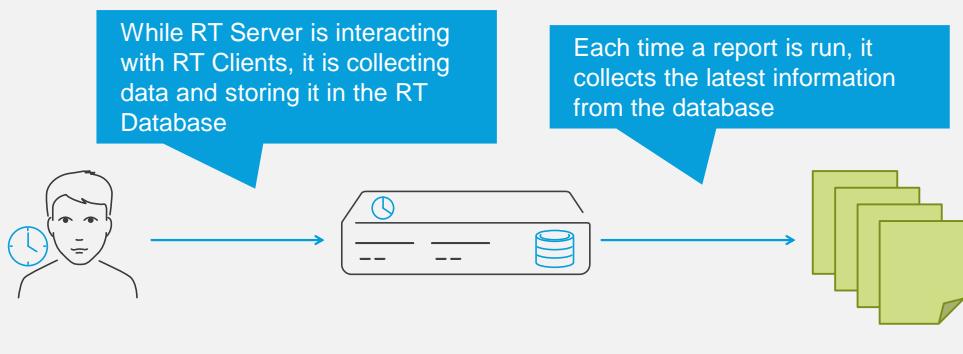
Lesson Preparation

- Publish and Run a solution which contains the following:
 - Composite user defined Type, marked as Collectable
 - Callout
 - Data Collection Activity for the above items
 - Have the callout shown while the data collection is on
 - Invoke the *Activate* and *Collect* functions from the Monitor

NICE®

High Level Flow

- APA provides a set of useful Cognos-based default reports and also enables you to define a variety of additional reports. These reports have different uses such as Attainment of goals and Performance statistics



NICE®

- Cognos is IBM's Business Intelligence (BI) and performance management software suite.
- The software is designed to enable business users without technical knowledge to extract corporate data, analyze it and assemble reports.

Static Data Models

- Static Data Models are permanent data structures, from which you can drag fields. Every field represents data that the RT Server collects information on



Report Area

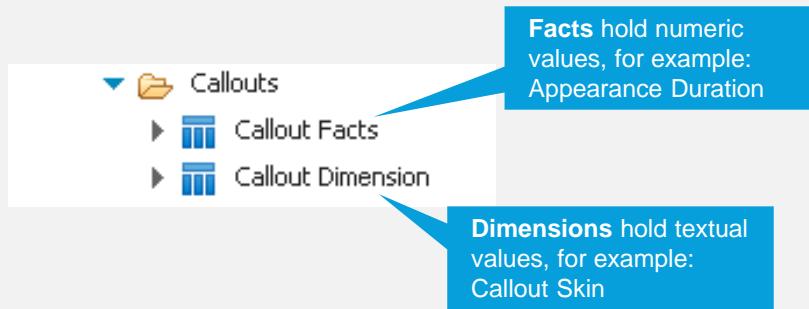
Alison	Sale Attempt	10	9
Jeff	Sale Attempt	7	9

NICE®

- Each Data Model consists of a hierarchy of data that is collected by RT.
- Every Data Model has a Users branch.
- **Static Data Models** means that the type of information collected by these models is predefined and cannot change.
- All data items within a given data model can be used in a report.
- You cannot create a report that combines data from different data models.

Dimensions and Facts

- The data within each data model is broken down into two data types: Dimensions and Facts

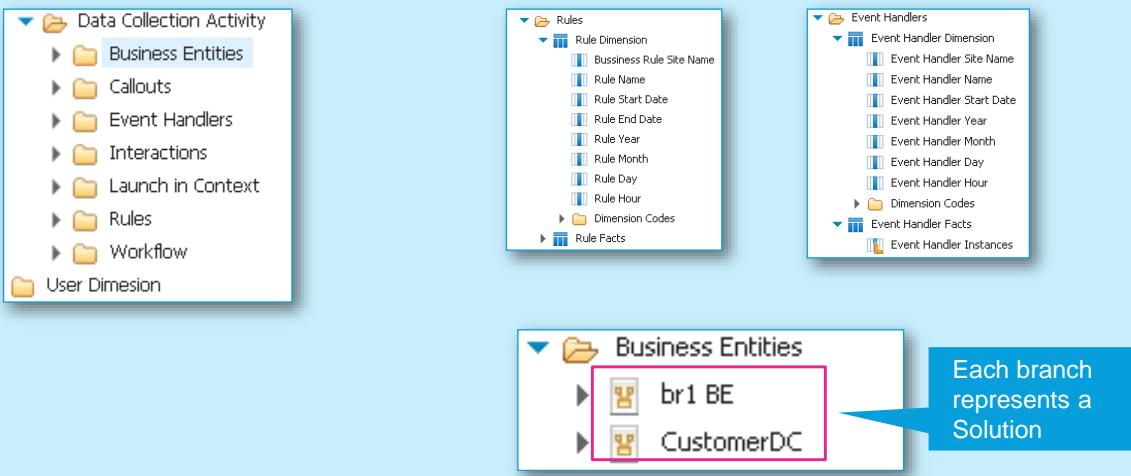


NICE®

- Some items in a data model contain both **Dimension** branches and **Dimension Code** branches. Code-related branches contain unique identifiers.
- They are more efficient when running internal queries for a report, since query searches are performed for an entity based on its index and not on its name.

Data Collection Reports

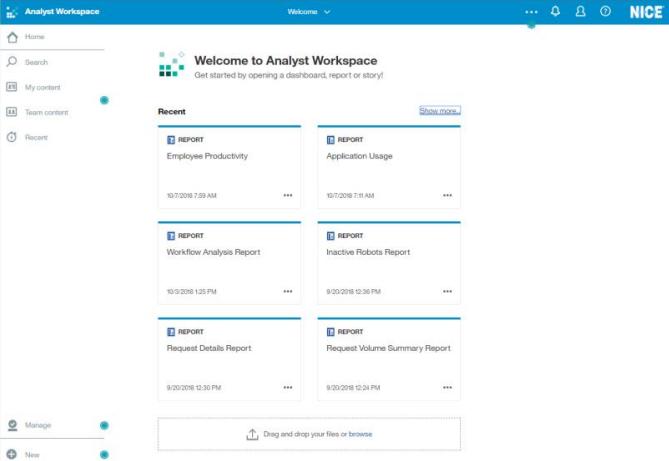
- The Data Collection Activity data model represents the various objects for which RT collects data, such as Rules, Event Handlers and so on



NICE®

How to Create a Report

- Reports are accessed via the web using the following URL:
<http://localhost/ibmcognos/bi/?perspective=home>



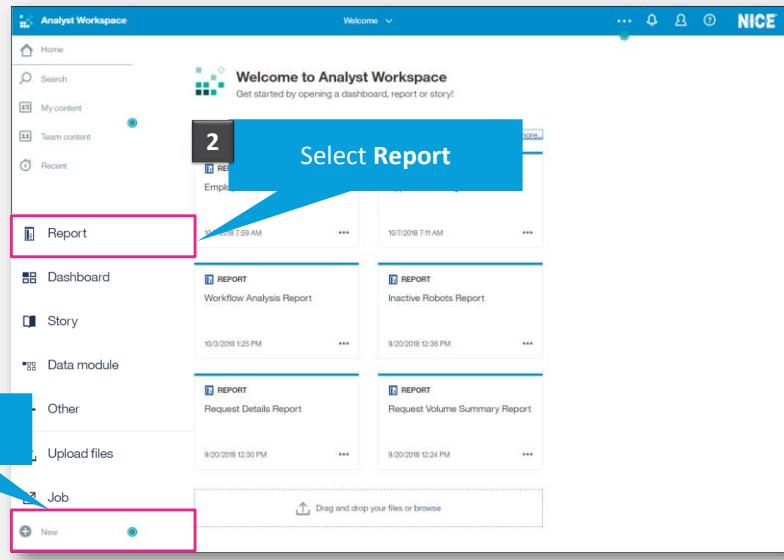
The screenshot shows the 'Analyst Workspace' interface. At the top, there's a navigation bar with 'Home', 'Search', 'My content', 'Team content', and 'Recent'. Below the navigation is a 'Welcome to Analyst Workspace' message with a 'Show more...' link. The main area is titled 'Recent' and displays a grid of six report cards. Each card includes a report icon, the report name, and a timestamp. A 'Show more...' link is located at the top right of the grid. At the bottom left, there are 'Manage' and 'New' buttons, and a central area for dragging files.

Report Name	Timestamp
Employee Productivity	10/7/2018 7:59 AM
Application Usage	10/7/2018 7:11 AM
Workflow Analysis Report	10/9/2018 12:25 PM
Inactive Robots Report	9/20/2018 12:36 PM
Request Details Report	9/20/2018 12:30 PM
Request Volume Summary Report	9/20/2018 12:24 PM

NICE®

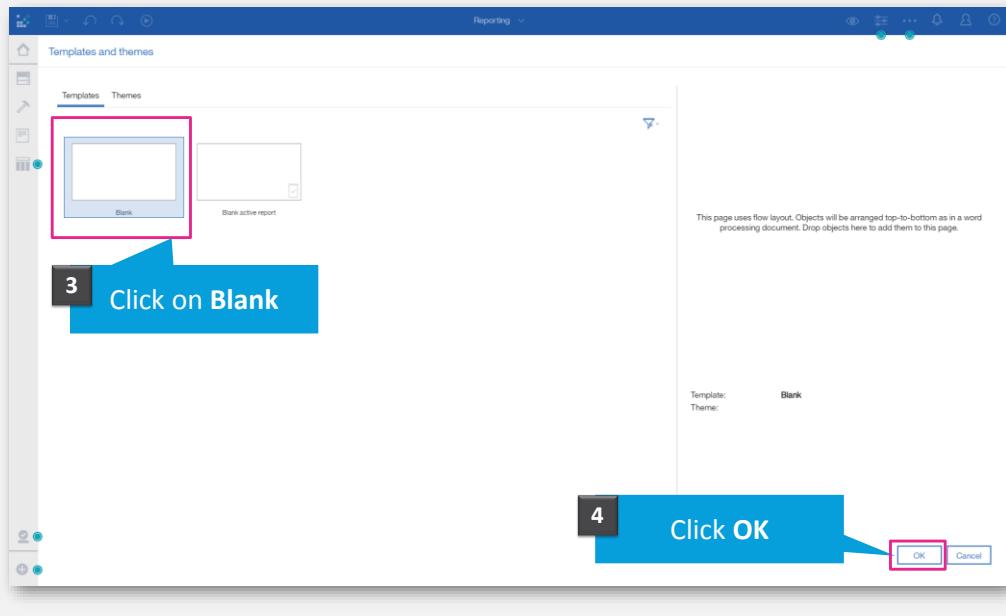
How to Create a Report

- Connect to <http://localhost/ibmcognos/bi/?perspective=home>



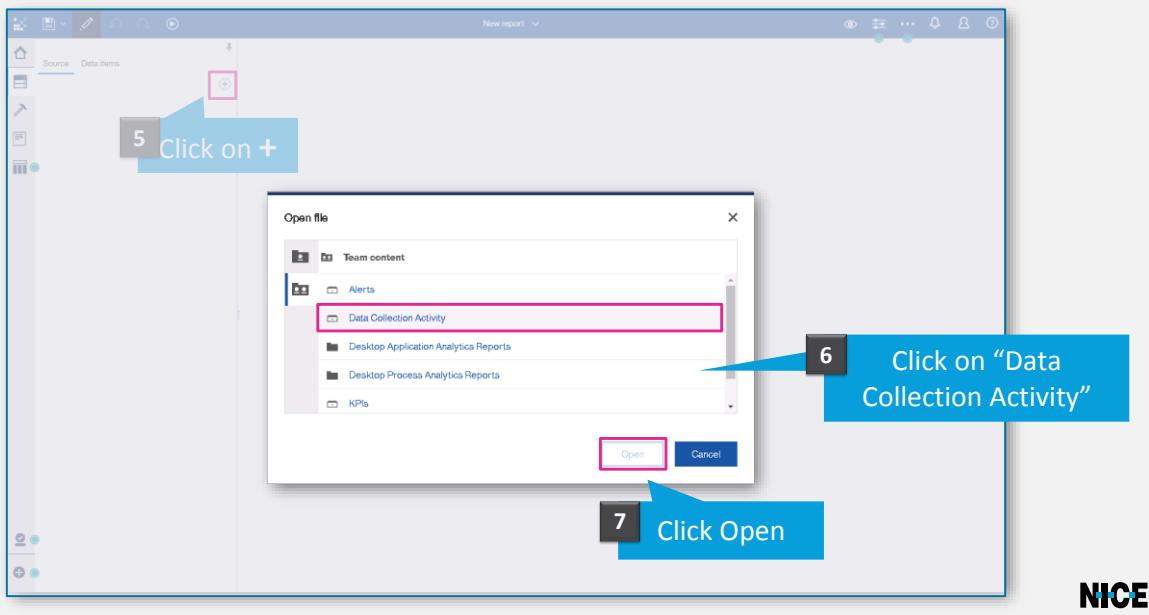
NICE®

Creating a Report

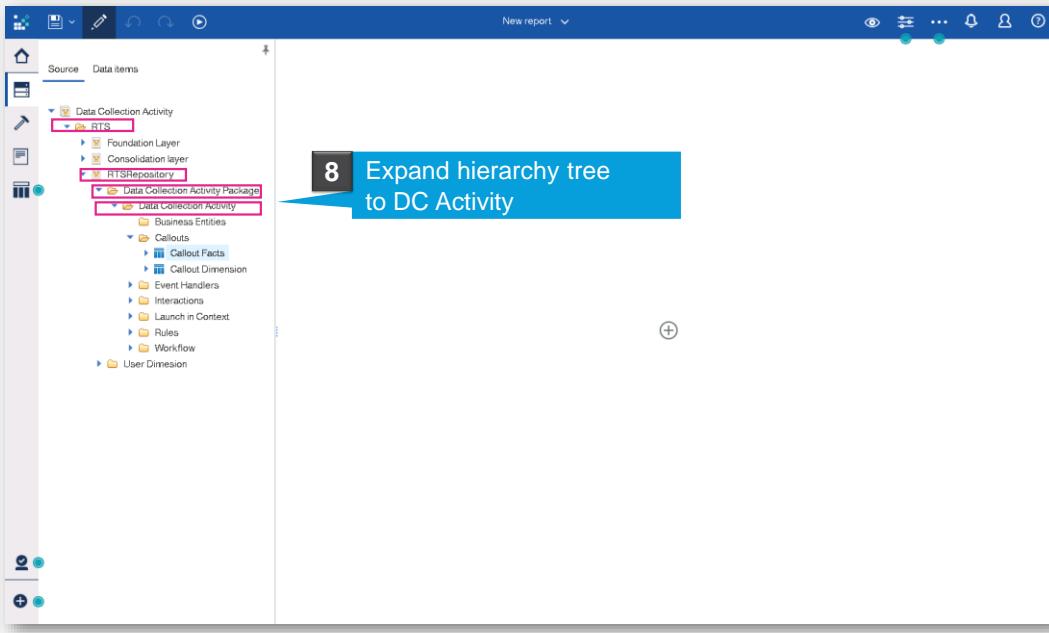


NICE®

Creating a Report



How to Create a Report



NICE®

- Expand **Data Collection Activity > OLD RTI > eGlueRepository > Data Collection Activity**.
- The different data collection entities will be shown.

Creating a Report

The screenshot shows the software interface for creating a report. On the left, there's a navigation pane with sections like Source, Data items, Data Collection Activity, RTS, Foundation Layer, Consolidation layer, RTSRepository, Data Collection Activity Package, Callouts, Callout Facts, Event Handler, Interactions, Launch in, Rules, Workflows, and User Dimension. A blue callout labeled '9' points to the 'Callout Facts' section under 'Callouts'. Another blue callout labeled '10' points to the 'Drag the Report to the Canvas' area. A third blue callout labeled '11' points to the 'Run Report' button at the top left. The main workspace on the right displays a report structure with columns: Callout Appearance Duration, Callout Instances, Callout Number Of Blinks, and Link Use Quantity. The rows show various summary and detail levels for these metrics.

11 Run Report

9 Select the required Report

10 Drag the Report to the Canvas

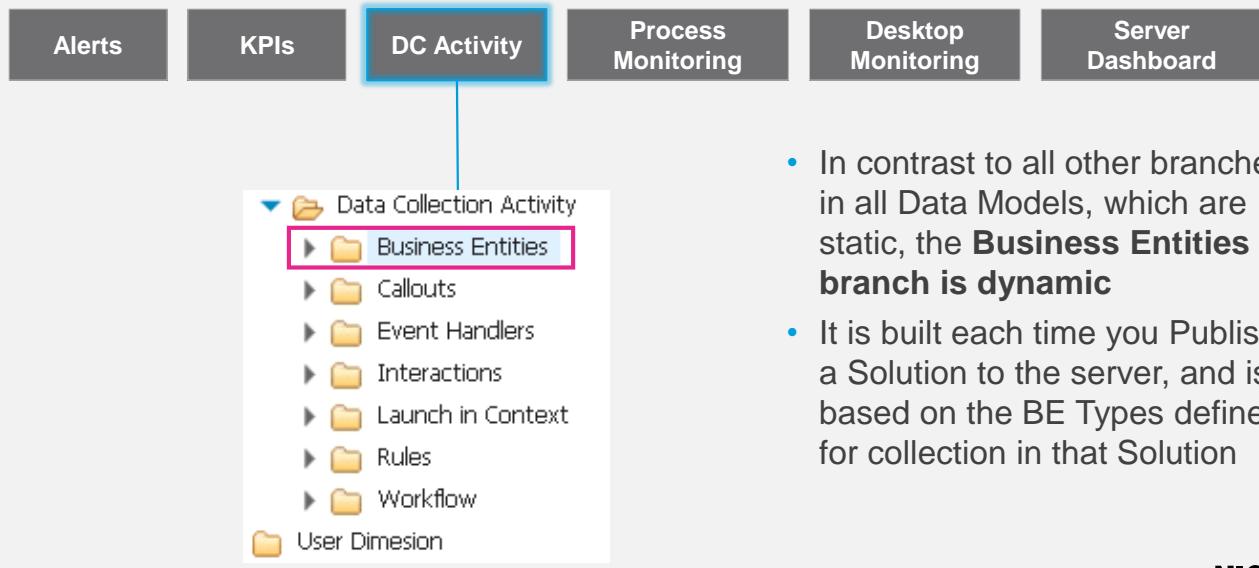
11 Run Report

New report* ▾

Callout Appearance Duration	Callout Instances	Callout Number Of Blinks	Link Use Quantity
<Callout Appearance Duration>	<Callout Instances>	<Callout Number Of Blinks>	<Link Use Quantity>
<Callout Appearance Duration>	<Callout Instances>	<Callout Number Of Blinks>	<Link Use Quantity>
<Callout Appearance Duration>	<Callout Instances>	<Callout Number Of Blinks>	<Link Use Quantity>
<Summary(Callout Appearance Duration)>	<Summary(Callout Instances)>	<Summary(Callout Number Of Blinks)>	<Summary(Link Use Quantity)>
Overall - Summary			

NICE®

Business Entities Branch

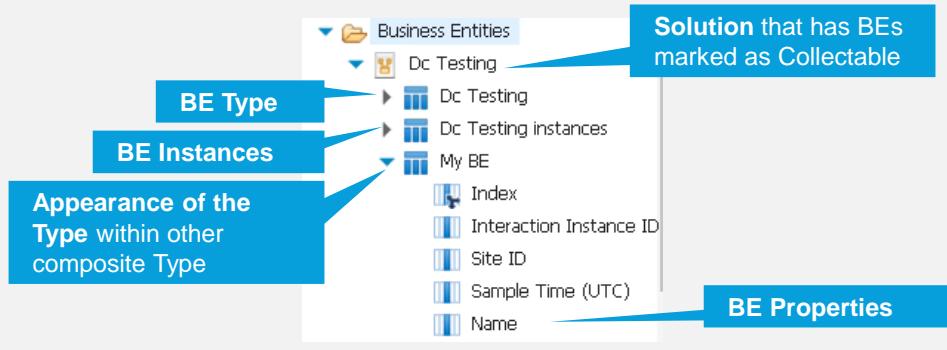


NICE®

- The Data Collection Activity data model represents the various objects for which the RT Server collects data, such as callouts, rules, event handlers and so on.
- For each branch in the hierarchy, there will be Dimensions and Facts.

Business Entities Branch

- Every Solution that you Publish is represented as a sub branch under the Business Entities branch
- In addition, the objects under each Solution also change each time the Solution is Published



NICE®

- Offer of Sales Attempt is the “Offer” Type when it appears as part of “Sales Attempt” Type.
- It is important to remember that there can be multiple instances of different contexts that are defined for collection within a given solution, such as Current Offer, Previous Offer, Last Offer and so on.
- Because of this possibility, all business entity type branches (such as Offer in our example) have a companion branch that is named **X instances**, where X is the name of the business entity type branch. For example: **Offer instances**.

Business Entities Branch

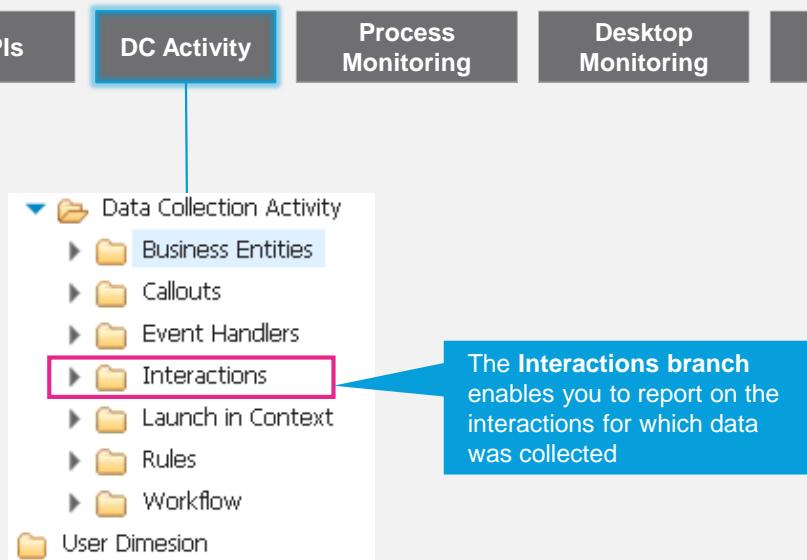
- All Business Entity Types have the same 3 properties in common:
 - Index
 - Interaction Instance ID
 - Site ID
- For Instances of that Type, the Instance will have its own Index, plus the Type's Index to differentiate between different Instances

Offer	Current Offer	Last Year's Offer	Refused Offer
Index =10	Index =21 Offer Index =10	Index =22 Offer Index =10	Index =23 Offer Index =10

NICE®

- **Index:** The index of the Type/Instance.
- **Interaction Instance ID:** The ID of the interaction Instance for which the data was collected.
- **User ID:** The ID of the site from which the data was collected.
- In order to differentiate between the Instances, you will have to add the Instance's Index to the Report as a column.

Interactions Branch



NICE®

- Each time you invoke the Activate and Collect Functions for a specific Data Collection activity, its data is collected as a transaction in the database.
- The Activate Function invokes the collection of data and the Collect Function stops that collection of data. Thus, each start (Activate) and stop (Collect) pair constitutes an instance of one interaction.



Notes from Demo

DO IT YOURSELF

Creating a Report: Business Entities

- Select the **Data Collection** data model
- Run Query Studio
- Insert all the fields from the **Business Entities** branch into the Report Area
- Create a report based on the callout data collected

NICE®

Robotic Automation Reports

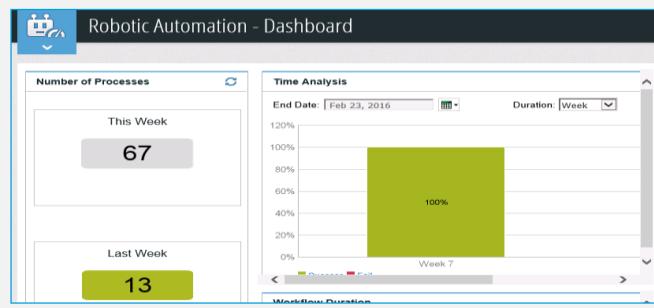


Robotic Automation Reports

- There are 2 sources for reports:
 - Web Portal - Dashboard
 - Cognos

The screenshot shows the 'Public Folders' section of the NICE Analyst Workspace. At the top, there are tabs for 'Public Folders' and 'My Folders'. Below the tabs, there is a search bar and a column header 'Name'. Underneath, there is a list of items:

- Server Dashboard
- Server Dashboard Reports
- Robotic Automation
- Robotic Automation Reports



NICE®

- The Robotic Automation solution includes predefined reports. Some can be accessed through the Automation Portal, and others through the Cognos.

Robotic Automation Reports

- These 3 reports cannot be accessed from the Dashboard

The screenshot shows the NICE Analyst Workspace interface. At the top, there's a navigation bar with 'Public Folders' and 'My Folders' tabs, and a breadcrumb path 'Public Folders > Robotic Automation Reports'. Below the navigation is a toolbar with various icons. The main area displays a list of reports under the heading 'Name'. There are ten items in the list, each with a checkbox and a preview icon. Four specific reports are highlighted with a pink border: 'Inactive Robots Report', 'Number of Processes Report', 'Request Details Report', and 'Request Volume Summary Report'. To the right of the report list is a column titled 'Modified' showing the last update time for each report. At the bottom right of the workspace is the 'NICE' logo.

Name	Modified
Inactive Robots Report	December 22, 2015 8:53:18 AM
Number of Processes Report	December 22, 2015 8:53:38 AM
Request Details Report	January 11, 2016 8:28:45 AM
Request Details Report - For Scheduling	December 22, 2015 8:54:10 AM
Request Details Report Drill	December 22, 2015 8:54:17 AM
Request Volume Summary Report	December 22, 2015 8:54:27 AM
Time Analysis Report	January 14, 2016 7:05:14 AM
Workflow Analysis Report	January 14, 2016 7:18:30 AM
Workflow Duration Report	January 14, 2016 7:09:16 AM

- The Robotic Automation solution includes predefined reports. Some are used in the Automation Portal, and others can be generated using Cognos.
- We will now focus on those that can be generated using Cognos, and elaborate on the insights that can be gained from each one.
- The Request Details Report for Scheduling can only be presented using the Web Portal.

Robotic Automation Reports

- Inactive Robots Report

The screenshot illustrates the NICE Analyst Workspace interface. On the left, the 'Inactive Robots Report Results' page is displayed with a table showing inactive durations for various robots. A callout points to the 'Robot ID (the VM hostname)' column, which highlights 'Server0' with a blue box. Another callout points to the 'Total duration of the inactivity for the selected timeframe * Drill in by clicking' row, which also highlights 'Server0' with a blue box. On the right, a separate window titled 'Inactive Robots Report Drill Results' shows detailed information for the selected robot, with a callout pointing to 'Detailed information for the selected robot'.

Robot	Inactive Duration (min)
Server0	31,287,060
Server1	16,432,860
Server10	30,461,520
Server100	30,785,520
Server1000	18,314,580
Server1001	13,545,480

Estimated Start Time	Estimated End Time	Inactive Duration (min)
Jun 26, 2014		
Jun 26, 2014 11:49:09 AM	Jun 26, 2014 3:48:28 PM	861,540
Jun 26, 2014 3:48:28 PM	Jun 26, 2014 4:16:15 PM	100,020
Jun 26, 2014 4:16:15 PM	Jul 2, 2014 12:40:00 PM	30,325,500
Total		31,287,060

- The first report is the Inactive Robots Report. The purpose of this report is to provide targeted information for the IT users/Admins.
- The report can be filtered by date or by Robot, and allows you to view:
 - The Robot ID, which is the hostname of the VM on which it is running
 - The Total duration of the Robot's inactivity over the selected timeframe.

To drill-down into the details of an inactive Robot, click its Inactive Duration (min) link. The inactive durations are broken down more easily in a way that is simple to understand.

The main insights from this report include:

- Being able to track malfunctioning Robots
- Being able to obtain information about the length of time during which a Robot was inactive

Robotic Automation Reports

- Request Details Report

NICE Analyst Workspace

Request Details Report Results

Report Filters

Date Range: NICE Internal Request ID

External (Customer) Request ID

Workflow Priority

Date/Time Invocation Entered the Queue

Request Duration

Robot Name = VM Hostname

Error Details Information

Internal Request ID	External Request ID	Workflow Name	Priority	Entered Into Queue (date)	Duration (sec)	Workflow Result	Robot	Error Type	Error Details
2339	EXT-REQ-ID	RA2.WF2		Aug 13, 2015 1:19:45 PM	11		W7X64RACLINT496_8435b3a434ee4919b97134284df51b01	System Error	i
2338	EXT-REQ-ID	RA2.WF2		Aug 13, 2015 1:19:44 PM	8		W7X64RACLINT496_8435b3a434ee4919b97134284df51b01	System Error	i
2313	EXT-REQ-ID	RA2.WF2	High	Aug 10, 2015 1:14:14 PM	2,764			System Error	i
2312	EXT-REQ-ID	RA2.WF2	High	Aug 10, 2015 1:14:13 PM	2,781			System Error	i
2311	EXT-REQ-ID	RA2.WF2	High	Aug 10, 2015 1:14:12 PM	2,774			System Error	i
2310	EXT-REQ-ID	RA2.WF2	High	Aug 10, 2015 1:14:11 PM	2,769			System Error	i

Invoked Workflow Name

Free Text Value Assigned within the Workflow

NICE

- The second report is the Request Details Report. The purpose of this report is to determine which failed requests to rerun.
- The report can be filtered by Date, Error Type, Workflow and Business Data values, and allows you to view:
 - The various request IDs
 - The request invocation time and duration
 - The Automation Workflow name
 - The Robot's VM hostname
 - The Workflow result and other business data values.

Clicking the icon next to each row opens the Request Details Report Drill window.

Robotic Automation Reports

- Request Details Report

The screenshot shows the NICE Analyst Workspace interface with the 'Request Details Report Drill' window open. The window is divided into several sections:

- General request Information:** This section contains basic details about the request, such as 'Entered Into Queue: Feb 23, 2016 12:57:41 PM', 'Request Status: Success', and 'Workflow Name: robotic1_gabyRobot'. It also lists various Business Data fields (Business Data 1 through Business Data 9) and an OS Login ID.
- XML:** This section displays an XML fragment for each internal request ID. A callout points to this area with the text 'XML results for each internal request ID'.
- Request Parameters:** This section shows a table of request parameters and their values. A callout points to this area with the text 'Request parameters as assigned during invocation'.

- The report includes the following information:
- Request Information
- XML results for each internal request ID (you can export the details of each report result to an external file)
- Request parameters (displayed at the bottom of the Request Details Report Drill window)

Robotic Automation Reports

- Request Volume Summary Report

NICE Analyst Workspace

Request Volume Summary Report Results

Report Filters

Date Range: Dec 06, 2017 to Jan 6, 2018

Error Type: All

Workflow Status: All

Date

of successful and failed workflows

Date	# of Requests	# of Success	# of Failures	% Success	% Failures
Dec 6, 2017	0	0	0		
Dec 7, 2017	0	0	0		
Dec 8, 2017	0	0	0		
Dec 9, 2017	0	0	0		
Dec 10, 2017	0	0	0		
Dec 11, 2017	0	0	0		

of requests at date

% of successful and failed workflows

NICE

- The Request Volume Summary Report shows a breakdown of the number and results of workflows requests by date



Notes from Demo

DO IT YOURSELF

Reports

- Generate the Robotic Automation reports

NICE®

Server Dashboard Reports



Server Dashboard

- The Server Dashboard consists of reports that provide you with information across the entire population of users.

Automatic Update

Displays information about recent automatic update downloads across the users of the current environment

Auto Update Directory	Agent Team	Computer Name	Revision	Update Time
rfq4	GroupC	QA1XP1	30	27 Dec 2010 14:28:17
	GroupC	QA1XP1	33	27 Dec 2010 14:33:44
	GroupC	QA1XP1	35	28 Dec 2010 15:27:53
	GroupC	QA4XP2	37	2 Jan 2011 09:33:01

Logged-in Users

Displays information about users currently logged in to the current environment

Supervisor	Agent Team	Agent	Login Time	Logout Time	Unexpected Logout?
NICE_A	nicesuper	30 Aug 2013 09:46:24	30 Aug 2013 09:51:24		Yes
NICE_A	nicesuper	3 Sep 2013 18:18:00	3 Sep 2013 18:18:37		No
NICE_A	nicesuper	3 Sep 2013 18:42:50	3 Sep 2013 18:44:13		No
NICE_A	nicesuper	3 Sep 2013 18:54:40	3 Sep 2013 18:54:41		No
NICE_A	nicesuper	3 Sep 2013 18:54:46	3 Sep 2013 18:59:56		No
NICE_A	nicesuper	3 Sep 2013 19:06:08	3 Sep 2013 19:06:46		No

Solution Downloads

Displays information about recent solution downloads across the users of the current environment

Solution Name	Version	Team	Employee	Computer Name	Download Time
Offer	1000001	NICE_A	niceagent2	23POD1	16 Sep 2013 10:45:07
20130909_DC	1000001	NICE_A	niceagent	23POD1	12 Sep 2013 15:56:33
20130909_DC	1000000	NICE	niceexec	23POC1	9 Sep 2013 17:39:43
BL Layer	1000005	NICE_A	niceagent4	23POD1	5 Sep 2013 01:28:35
BL Layer	1000004	NICE_A	niceagent2	23POD1	4 Sep 2013 14:18:57
BL Layer	1000002	NICE_A	niceagent2	23POD1	4 Sep 2013 09:38:38

- The Server Dashboard relies on the availability of the RT reports server (Cognos). If the reports server is not available, the automatic update and Solution download auditing data reside in the database, but is not accessible via the Server Dashboard.

Callout Display Report

- Number of Callout instances
- Number of callouts closed within 2 seconds
- Number of callouts moved on screen

Callout Name	Display Count	Closed < 2 Seconds		Moved From Original Location	
		Total	Percentage	Total	Percentage
Callout 3	28	19	67.86%	1	3.57%
Callout replace	33	13	39.39%	6	18.18%
New Callout 1	8	2	25.00%	1	12.50%
select 4	6	2	33.33%	3	50.00%
Overall	75	36	48.00%	11	14.67%



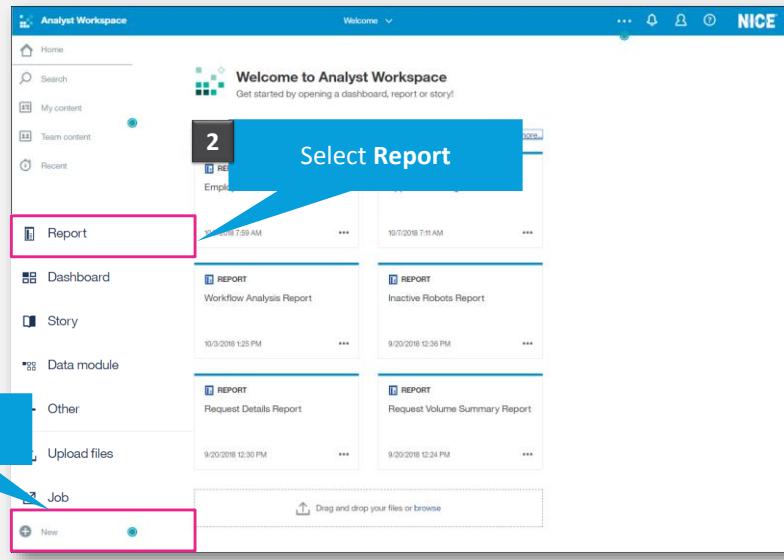
Note

This report will be relevant only when the APA license includes use of Callouts

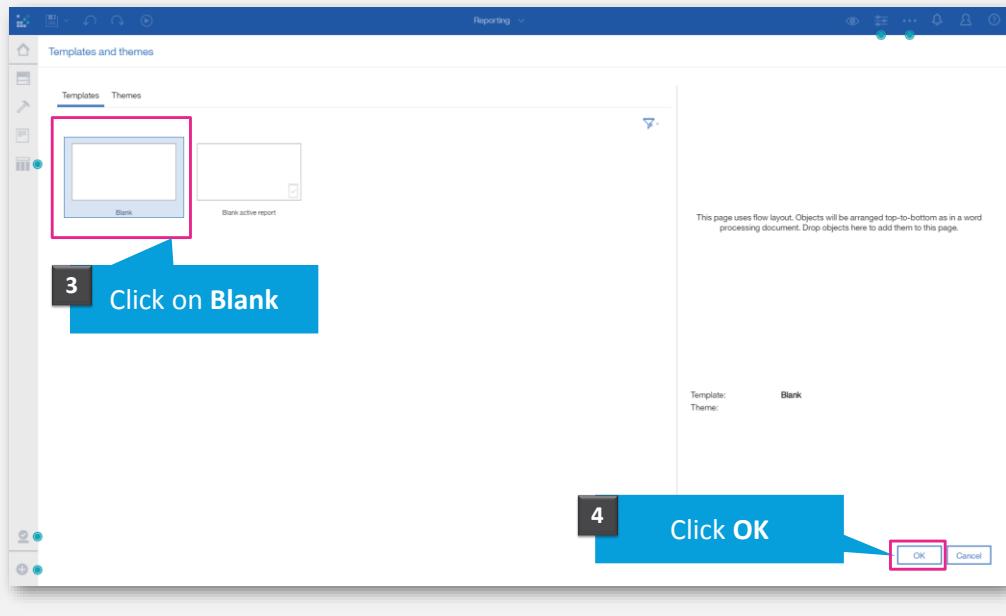
NICE®

Access The RTS Cognos Reports Server

- Connect to <http://localhost/ibmcognos/bi/?perspective=home>

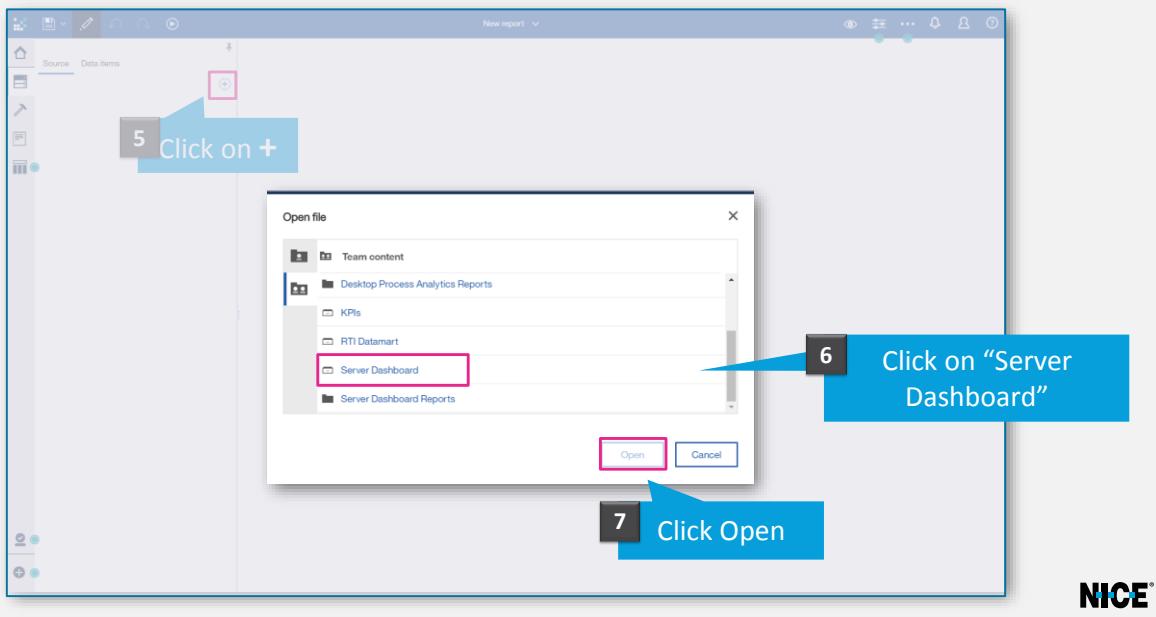


Creating a Report

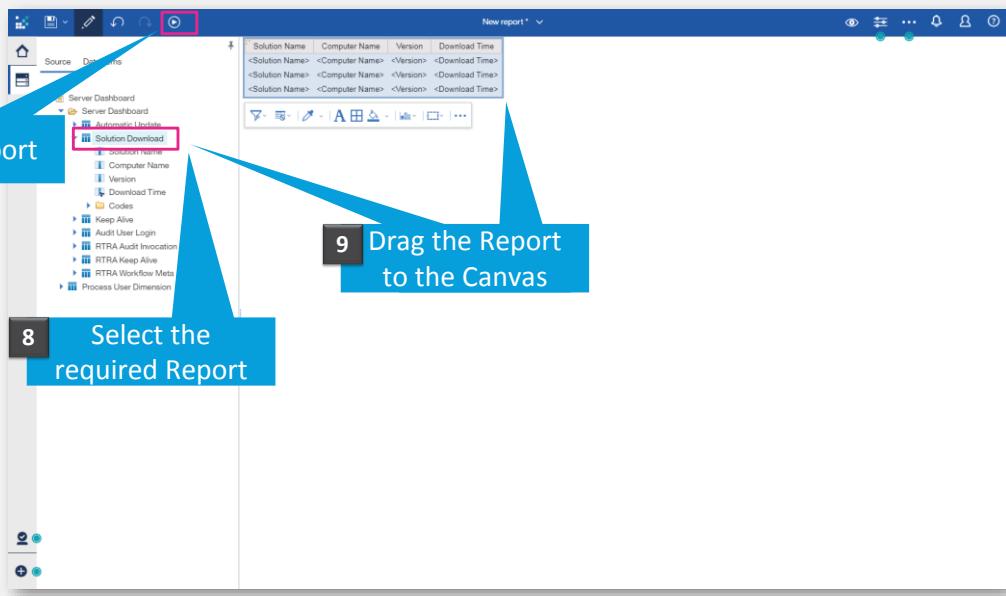


NICE®

Creating a Report



Creating a Report



NICE®

- **Solution Download Report** displays information about recent solution downloads across the users of the current environment.

Solution Download Report Results

The screenshot shows a web-based application titled "Solution Download - NICE Analyst Viewer". The interface includes a search bar at the top with fields for "Solution", "Version", "Company", and "Team", and buttons for "Search Again", "Close Search Section", and "Deselect". Below the search area is a table with columns: Solution Name, Version, Team, Employee, Computer Name, and Download Time. The table lists 20 entries of solution downloads, primarily from the GROUPA team, with download times ranging from April 14, 2015, to April 26, 2015. The table has sorting options for "Sort By" (Download Time) and "Sort Order" (Descending). At the bottom of the page, there are navigation links for "Apr 16, 2015" and "2:13:11 PM". The NICE logo is visible in the bottom right corner.

Solution Name	Version	Team	Employee	Computer Name	Download Time
6.3stabProject	1000003	GROUPDA	GL19 E-GLUE	STAB2CLENSITE1	Apr 14, 2015 9:41:50 AM
6.3stabProject	1000003	GROUPA	GL19 E-GLUE	STAB2CLENSITE1	Apr 14, 2015 9:42:14 AM
6.3stabProject	1000003	GROUPA	GL0 E-GLUE	STAB2CLENSITE1	Apr 14, 2015 9:52:31 AM
6.3stabProject	1000003	GROUPD	GL10 E-GLUE	STAB2CLENSITE1	Apr 13, 2015 7:10:21 AM
6.3stabProject	1000003	GROUPDA	GL15 E-GLUE	STAB2CLENSITE1	Apr 13, 2015 7:13:08 AM
6.3stabProject	1000003	GROUPCB	GL13 E-GLUE	STAB2CLENSITE1	Apr 13, 2015 6:44:37 AM
6.3stabProject	1000003	GROUPA	GL0 E-GLUE	KSD090141323	Apr 8, 2015 12:55:29 PM
6.3stabProject	1000003	GROUPA	GL0 E-GLUE	STAB2CLENSITE1	Apr 13, 2015 12:55:41 PM
6.3stabProject	1000003	GROUPA	GL2 E-GLUE	STAB2CLENSITE1	Apr 2, 2015 5:19:17 PM
6.3stabProject	1000003	GROUPC	GL8 E-GLUE	STAB2CLENSITE1	Mar 26, 2015 8:16:28 AM
6.3stabProject	1000003	GROUPAA	GL5 E-GLUE	STAB2CLENSITE1	Mar 25, 2015 5:40:30 PM
6.3stabProject	1000003	GROUPB	GL7 E-GLUE	STAB2CLENSITE1	Mar 25, 2015 5:46:30 PM
6.3stabProject	1000003	GROUPA	GL0 E-GLUE	STAB2CLENSITE1	Mar 25, 2015 5:49:41 PM
6.3stabProject	1000002	GROUPB	GL7 E-GLUE	STAB2CLENSITE1	Mar 25, 2015 1:52:15 PM



Notes from Demo

DO IT YOURSELF

Solution Download Report

- Run the RT Client. The solution you published earlier should download to the Client.
- Run a Solution Download report with the appropriate filters to verify the Solution was downloaded.

NICE®

Summary

- Creating a Report with static and dynamic Data Models
- Producing Robotic Automation Reports
- Producing Server Dashboard Reports

NICE®

- The Cognos-based reports enable you to define a variety of reports.
- Static Data Models are permanent data structures, from which you can insert fields.
- The data within each data model is broken down into two data types: Dimensions and Facts, which represent textual and numeric values of the Entity, respectively.
- In contrast to all other branches in all Data Models, which are static, the **Business Entities branch is dynamic**. It is built each time you Publish a Solution to the server, and is based on the BE Types defined for collection in that Solution.
- The Robotic Automation solution includes predefined reports that allow you to track malfunctioning Robots, obtain information about the length of time during which a Robot was inactive, determine which failed requests to rerun and etc.
- Producing Server Dashboard Reports
- **Server Dashboard** allows you to produce reports that provide you with information about automatic updates and Solution downloads across the entire population of users.



Thank You



NICE[®]

CLIENT-TO-CLIENT



Lesson Objectives

By the end of this lesson you will be able to:

- Describe the Client to Client concept
- Use Client to Client Functions

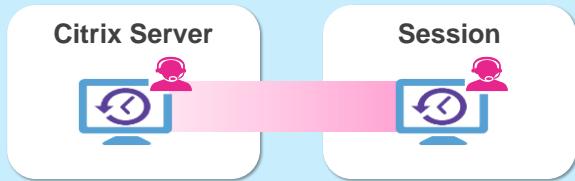


Client to Client Concept

- Client to Client feature enables two RT Clients to communicate with each other, whether or not they are running on the same machine
- Two possible implementations:

Citrix Channel

Citrix Server sends values to Citrix Client and vice versa



Remote Channel

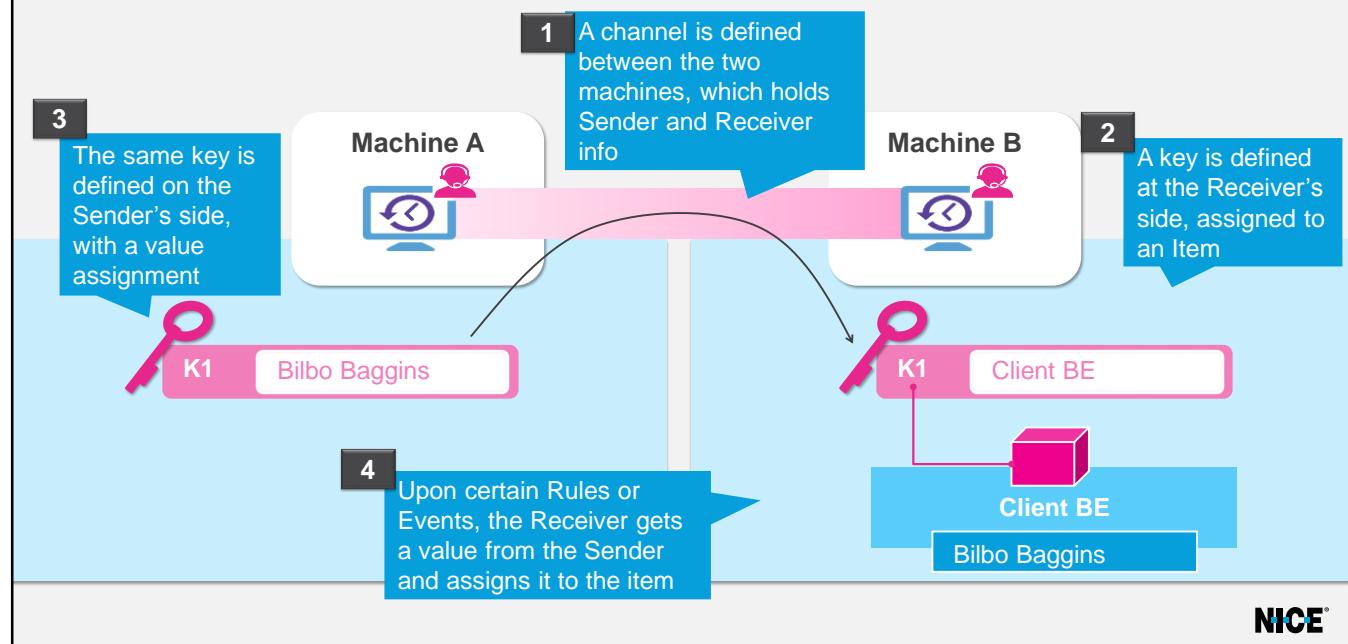
Remote machines send values to one another



NICE®

- The RT Client must run on the same machine as the running application. So when there are different applications running on two different machines, Clients on these machines must communicate with each other to send Business Entity values.

How Does it Work?



- **Citrix example** - Citrix session provides numerous screens that trigger the Presentation of Callouts in the RT Client and the application running in the desktop handles background processes and communicates with a business application database server.
- **Remote example** - when using virtual desktop – one Client will run on the real desktop and the other on the virtual one. Another example is when both a bank teller and a supervisor in another location are using the RT Client. The bank teller can forward a transaction to the supervisor for authorization, receive authorization and then proceed.

How to Create a Citrix Channel



Create a
Channel

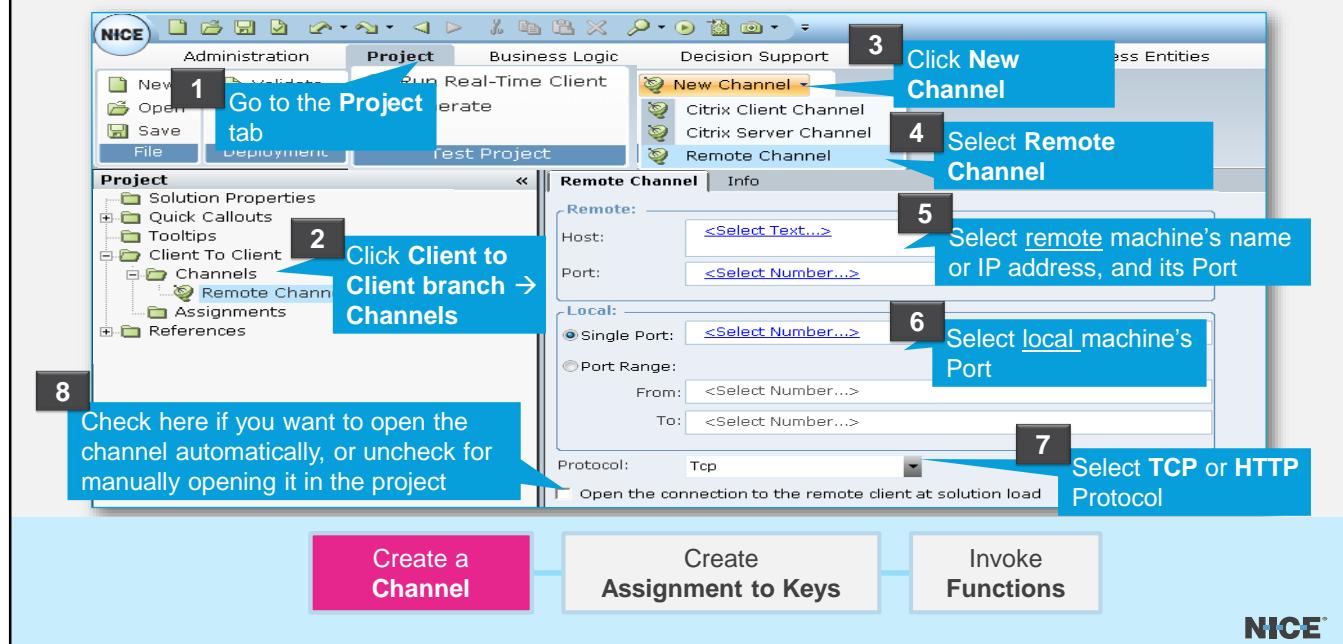
Create
Assignment to Keys

Invoke
Functions

NICE®

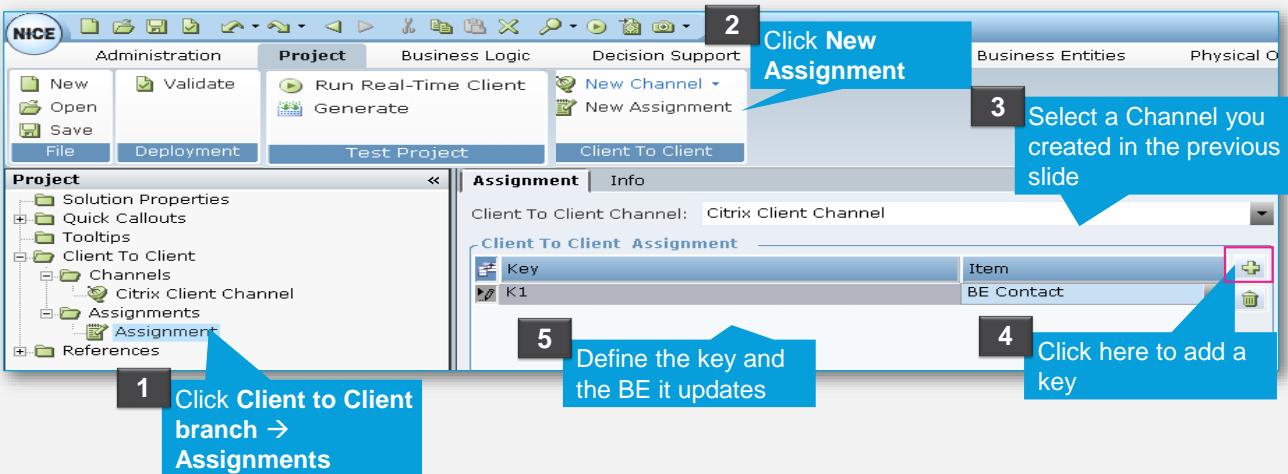
- A Channel must be defined on both machines, the Citrix Server and the Citrix Client.
- **Channel name** – Should be the same String on both machines, Server and Client. It is possible to select a Function, or a BE Text property, or just type a meaningful name.

How to Create a Remote Channel



- A Remote channel allows you to define the TCP connection between 2 Projects residing on different hosts.
- Step 6 notes:
 - **Single Port:** For a pre-defined connection.
 - **Port Range:** For dynamic connections; Will scan the port range for first free port

How to Create Assignment to Keys



Create a
Channel

Create
Assignment to Keys

Invoke
Functions

NICE®

- **Key Name** - must be unique and identical on both sides.

Client to Client Functions Invoke Assignments

- Citrix Channels have one Function



- Remote Channels have 3 additional Functions

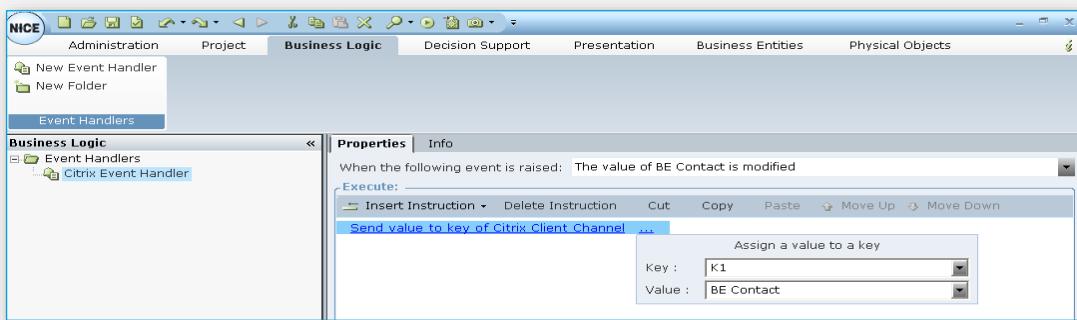


NICE®

- **Reconnect to host / local port** – for fail safe and manually start connections.
- **Send value to key asynchronous** – send messages asynchronously identical on both sides.

Invoking Functions in the Designer

- Create a Rule or an Event Handler to define when the key is sent and what its value is.



Create a
Channel

Create
Assignment to Keys

Invoke
Functions

NICE®

- An Integration Expert will create the necessary Business Entities for storing the values to be sent across / received from the Client to Client Channel.

Summary

- The Client to Client concept
- How to use Client to Client Functions

NICE®

- Client to Client feature enables two RT Clients to communicate with each other, whether or not they are running on the same machine



Thank You



NICE®

ADMINISTRATION



Lesson Objectives

By the end of this lesson you will be able to:

- Work with different Environments
- Map different DB connections per Environment
- Map different text variables per Environment



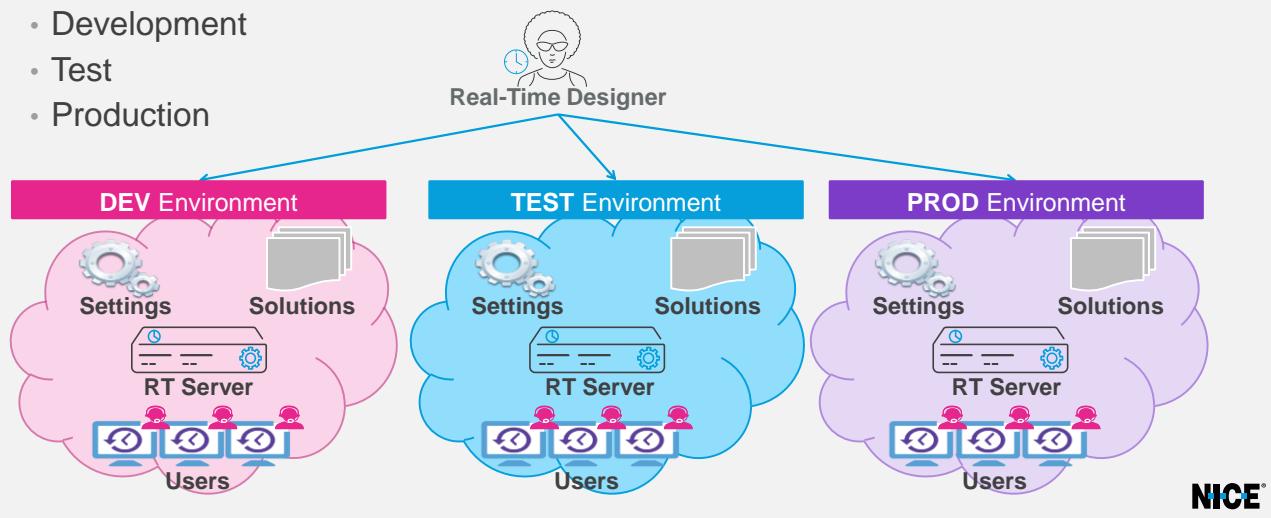
Environments



NICE®

Working with Multiple Environments

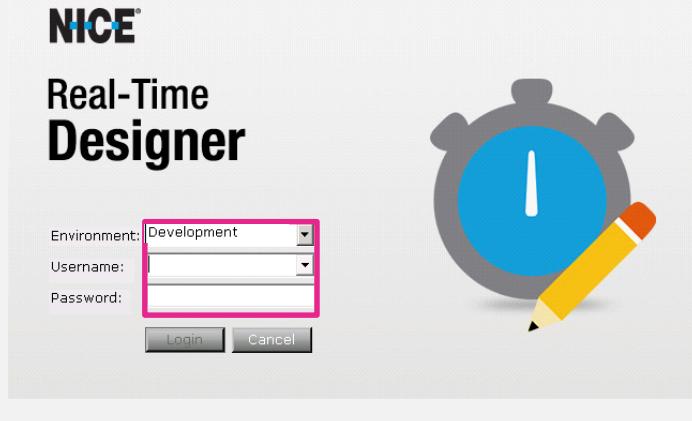
- The published Solution supports multiple environments. Designer provides three default environments:
 - Development
 - Test
 - Production



- Each Environment is associated with its own RT Server. So each Environment has its own solutions published to it, its own groups of users, and its own settings.
- Each environment has access only to the display names database that were defined to it. For example, the Development environment will only contain display names that were applied to the Development environment. It will not have access to display names applied to the Test environment.

Working with Multiple Environments

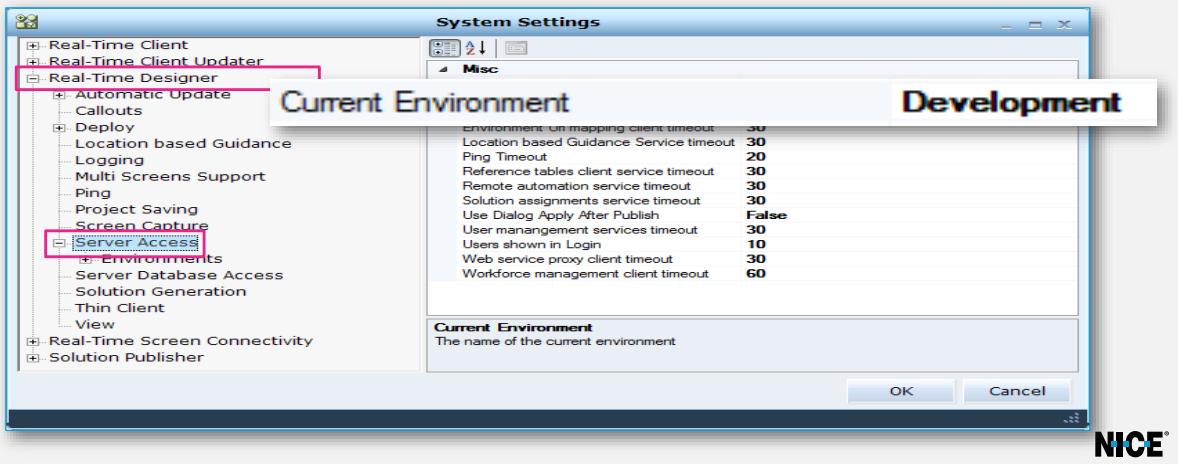
- Default Environment:
 - Chosen at Login...



- The default environment that appears in the Administration tab is configured at the **RTDesigner.exe.config** file at the **currentEnvironment** property.
- When logging into RT Designer, you must enter the login credentials for the current environment.
- When you want to change to another environment within the Administration module, RT Designer automatically attempts to log into the new environment and perform authentication using the current credentials.
- If RT Designer cannot log in using these credentials, an error message is displayed. Click OK and then type in the new credentials, and click on the Key icon to login.

Working with Multiple Environments

- Default Environment:
- Chosen at Login...
- ... and seen in Designer Settings



- The default environment that appears in the Administration tab is chosen when logging into the Designer, before entering the login credentials for a user in that environment.

Working with Multiple Environments

- You can also move from one environment to another directly from the Designer



NICE®

- When you want to change to another environment within the Administration module, RT Designer automatically attempts to log into the new environment and perform authentication using the current credentials.
- If RT Designer cannot log in using these credentials, an error message is displayed. Click OK and then type in the new credentials, and click on the Key icon to login.

Working with Multiple Environments

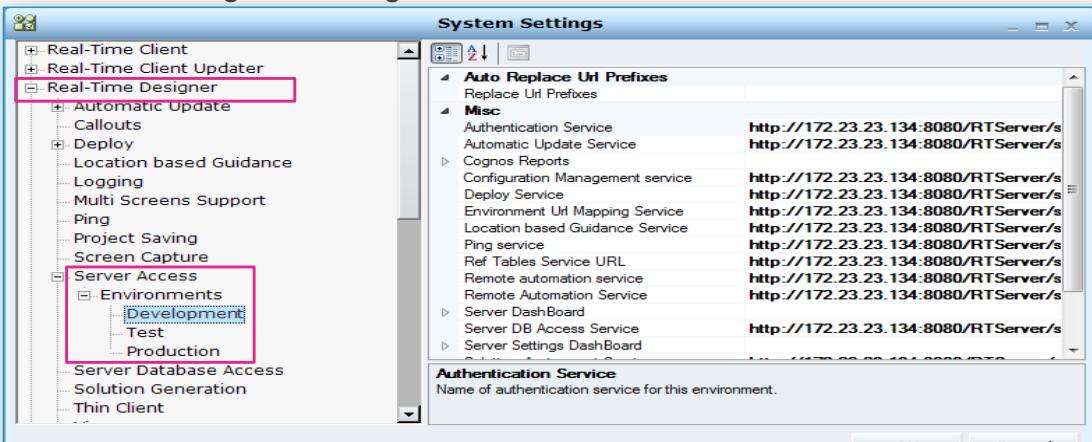
- Existing Environments:
 - Configured in RTDesigner.exe.config...

```
<ServerAccess currentEnvironment="Development" usediald>
  <Environments>
    <add name="Development" displayName="Development" DeployService="true">
      <svn active="true" url="http://LTCG3POS:3691/eGluon/development">
        <ServerDashBoard url="http://LTCG3POS/cognos8/cgibin/ServerDashBoard">
          <ReportConfiguration url="http://LTCG3POS/cognos8/cgibin/ReportConfiguration">
        </ReportConfiguration>
      </ServerDashBoard>
    </add>
    <add name="Test" displayName="Test" DeployService="true">
      <svn active="true" url="http://LTCG3POS:3691/eGluon/test">
        <ServerDashBoard url="http://LTCG3POS/cognos8/cgibin/ServerDashBoard">
          <ReportConfiguration url="http://LTCG3POS/cognos8/cgibin/ReportConfiguration">
        </ReportConfiguration>
      </ServerDashBoard>
    </add>
    <add name="Production" displayName="Production" DeployService="true">
      <svn active="true" url="http://LTCG3POS:3691/eGluon/production">
        <ServerDashBoard url="http://LTCG3POS/cognos8/cgibin/ServerDashBoard">
          <ReportConfiguration url="http://LTCG3POS/cognos8/cgibin/ReportConfiguration">
        </ReportConfiguration>
      </ServerDashBoard>
    </add>
  </Environments>
</ServerAccess>
```



Working with Multiple Environments

- Existing Environments:
 - Configured in RTDesigner.exe.config...
... and in Designer Settings



Working with Multiple Environments

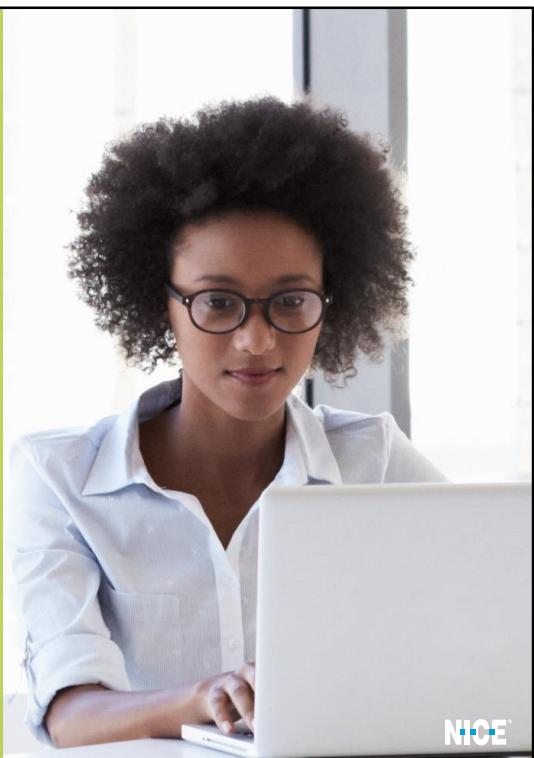
- New Environments:
 - Can be added in RTDesigner.exe.config, using the same structure

```
<ServerAccess currentEnvironment="Development" usedialog="true">
  <Environments>
    <add name="Development" displayName="Development" DeployService="true" DeployServiceType="Web">
      <svn active="true" url="http://LTCG3POS:3691/eGluonSVN" />
      <ServerDashBoard url="http://LTCG3POS/cognos8/cgibin/ReportServer" />
      <ReportConfiguration url="http://LTCG3POS/cognos8/cgibin/ReportServer" />
    </add>
    <add name="Test" displayName="Test" DeployService="true" DeployServiceType="Web">
      <svn active="true" url="http://LTCG3POS:3691/eGluonSVN" />
      <ServerDashBoard url="http://LTCG3POS/cognos8/cgibin/ReportServer" />
      <ReportConfiguration url="http://LTCG3POS/cognos8/cgibin/ReportServer" />
    </add>
    <add name="Production" displayName="Production" DeployService="true" DeployServiceType="Web">
      <svn active="true" url="http://LTCG3POS:3691/eGluonSVN" />
      <ServerDashBoard url="http://LTCG3POS/cognos8/cgibin/ReportServer" />
      <ReportConfiguration url="http://LTCG3POS/cognos8/cgibin/ReportServer" />
    </add>
  </Environments>
</ServerAccess>
```

NICE®

- You can also define additional environments. In order to add a new environment, the RT Designer configuration file (**RTDesigner.exe.config**) must be changed:
- The new environment entry must be added under **ServerAccess/Environments**, and must use the same format as the existing entries.
- The URL of the services and the repository provider must be changed, based on the type of server that is being added.

Users Administration



Users Import

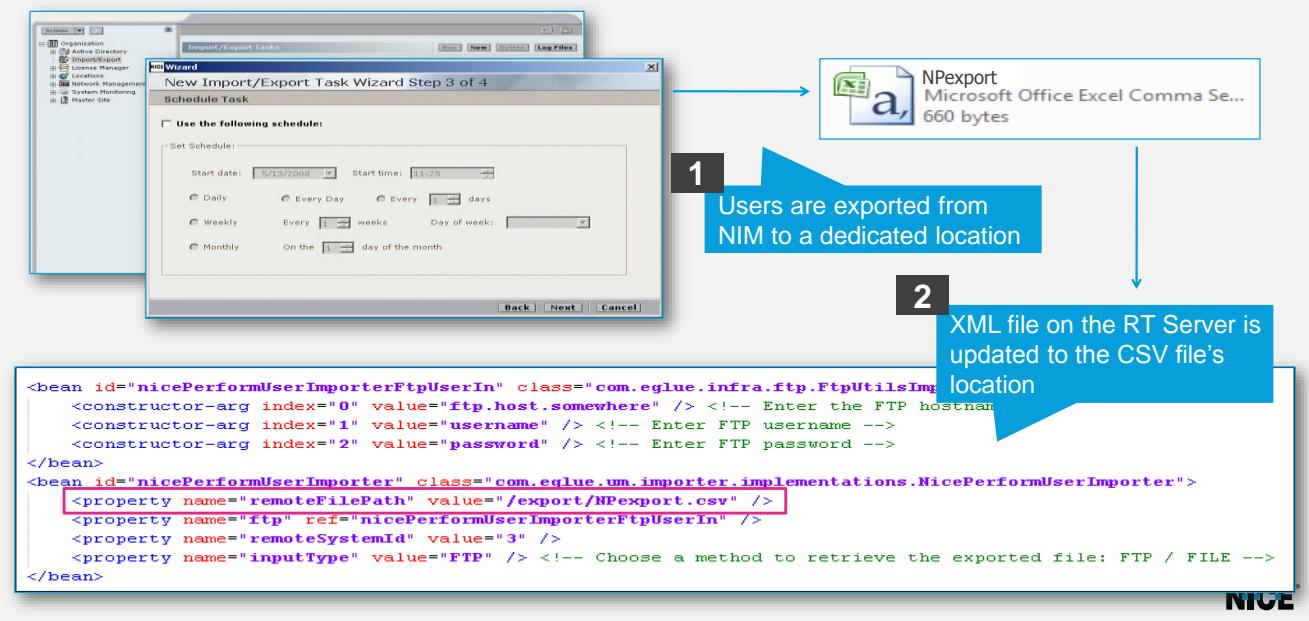
- Users are imported from Engage, WFM or from Active Directory used in the organization, by CSV files
- Import/export process is done only once during installation and scheduled by a task to update the files



NICE®

- After importing users, the administrator view will hold all teams and users that were exported from the dedicated destination.

Exporting Engage Users; Importing to RT Server



- XML filename is **user-management-beans.xml**.
- It's default location on the RT Server is **D:\nice_systems\RTServer\config\spring**.
- If the CSV file is stored on **NIM FTP**:
- Search for bean ID **nicePerformUserImporterFtpUserIn** to change FTP host name, username and password.
- Search for bean ID **remoteFilePath** to change URL to CSV file.
- If the CSV file is stored in the **RT Server local file system**:
- Search for bean ID **remoteFilePath** to change URL to CSV file.
- Refer to the **System Administration Guide** for more details.

Designer User Roles

- The role assigned to a user determines which module(s) the user can access

	Administration	Project	Business Logic	Decision Support	Presentation	Business Entities	Physical Objects
Integration Expert						✓	✓
Business Analyst	✓	✓	✓	✓	✓		
Technical Engineer	✓	✓	✓	✓	✓	✓	✓
Administrator Roles	✓						

NICE®

- Designer User Roles are assigned to power-users.
- Each role indicates which modules are visible to and editable by the users assigned that role.
- When starting RT Designer, only the modules permitted by the user's assigned role are visible. All other modules are loaded, but are not visible.

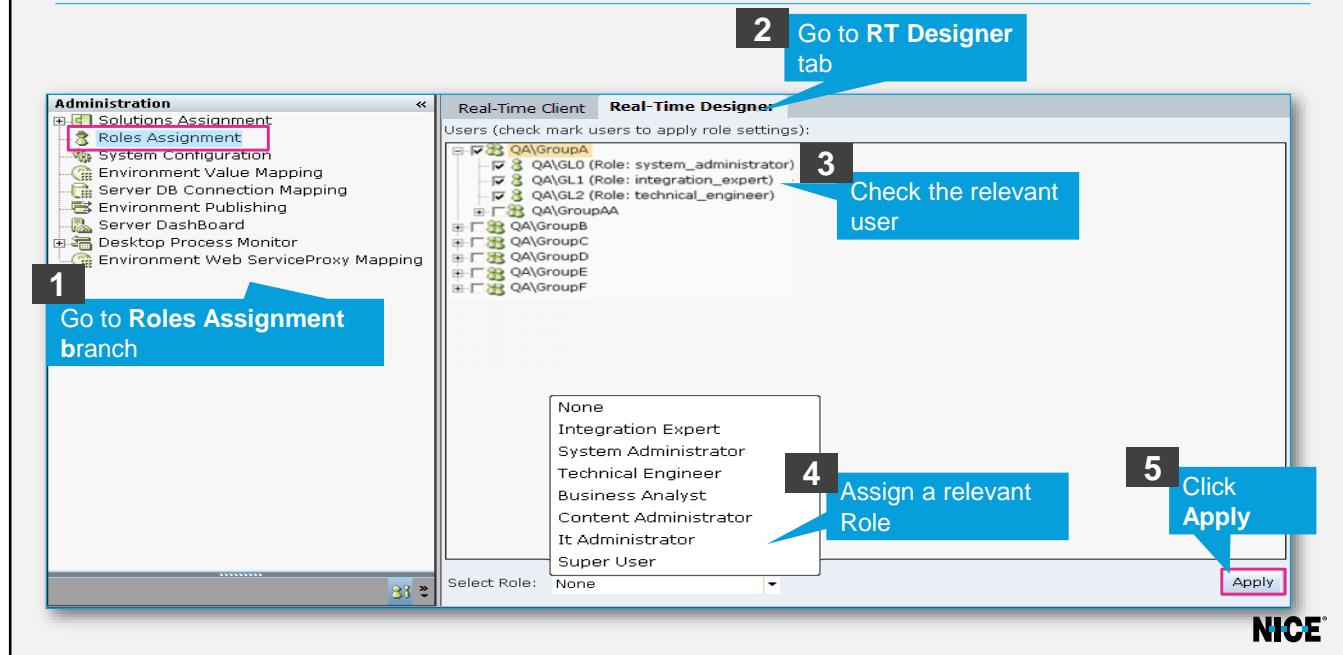
Designer User Roles: Administrator Roles

- Different Administrator roles can view and edit different branches in the Administrator module

	IT Admin	Content Admin	System Admin	Super User
Solutions Assignment		✓	✓	✓
Roles Assignment	✓		✓	✓
System Configuration	✓		✓	✓
Automatic Update		✓	✓	✓
Environment Value Mapping		✓	✓	✓
Server DB Connection Mapping	✓		✓	✓
Environment Publishing		✓	✓	✓
Server Dashboard		✓	✓	✓
Desktop Process Monitor		✓	✓	✓
Environment Web Service Proxy Mapping	✓		✓	✓
Location Cases Tab		✓		✓

- The IT Administrator, Content Administrator, System Administrator and Super User roles can view and edit different branches in the Administration module, as described in this table.
- Only the branches/tabs permitted by the user's assigned role are visible in Real-Time Designer.

How to Assign Designer User Roles



- The Role selected for a Team applies to all Users in the Team.

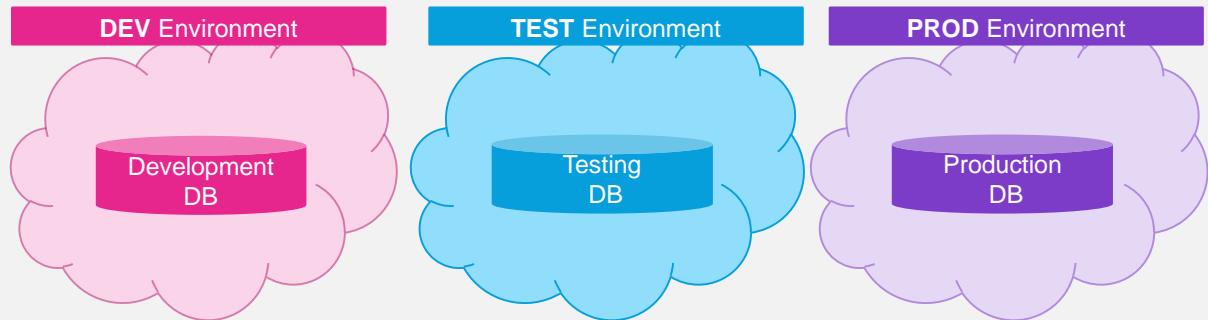
Assigning User Roles

17

The NICE logo, featuring the word "NICE" in a bold, sans-serif font with a registered trademark symbol.

DB Connection Mapping

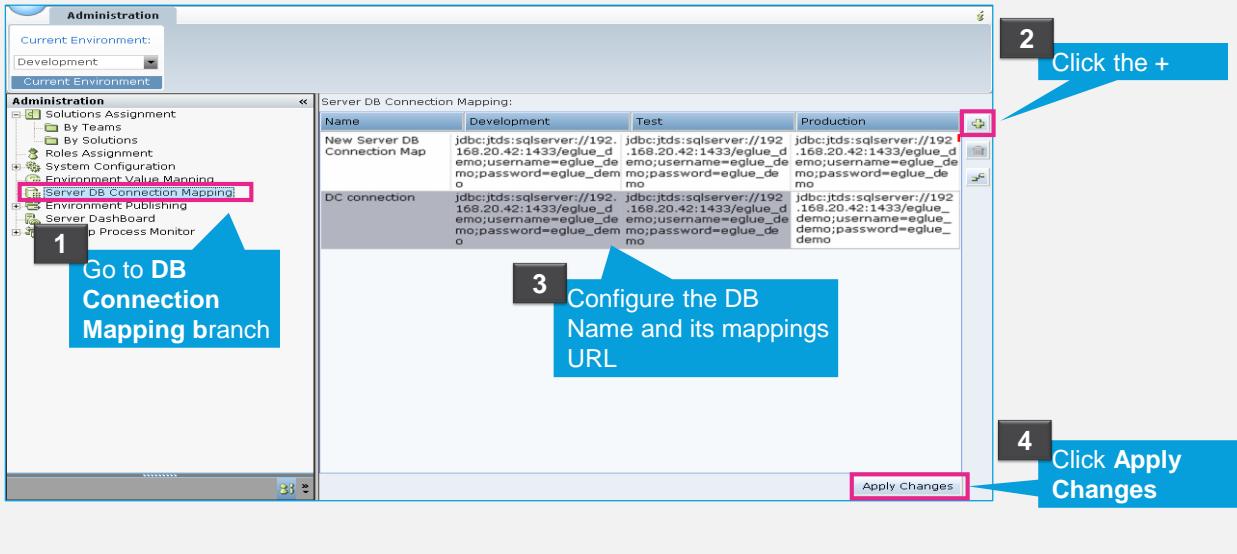
- Allows you to define a different DB connection value for every environment used



NICE®

- Database connection mappings are used in the Physical Objects Database module. There, in the Connection object, rather than defining a concrete connection string, you reference one of the mappings.
- Therefore, when a Designer user is ready to deploy a solution to another environment (for example, from Test to Production), the connection string in the project's DB Connection object does not need to be changed to specify the database details for the Production environment, as the database connection strings for all environments are defined in the database connection mapping.

How to Map a DB Connection



NICE®

- The values that you assign are maintained in separate databases, with one database per environment. Each row in the Server DB Connection Mapping area constitutes a record in the database.
- **DB Connection Configuration (Step 3)** - Note that the pane on the right contains four columns: **Name**, **Development**, **Test** and **Production**. The Name column specifies the name of the database connection. The three adjacent columns (Development, Test and Production) specify the names of the environment to which the database connection value applies.
- Note that if you have additional environments, the window also displays a column for each of them as well.

Environment Value Mapping



Environment Value Mapping Feature

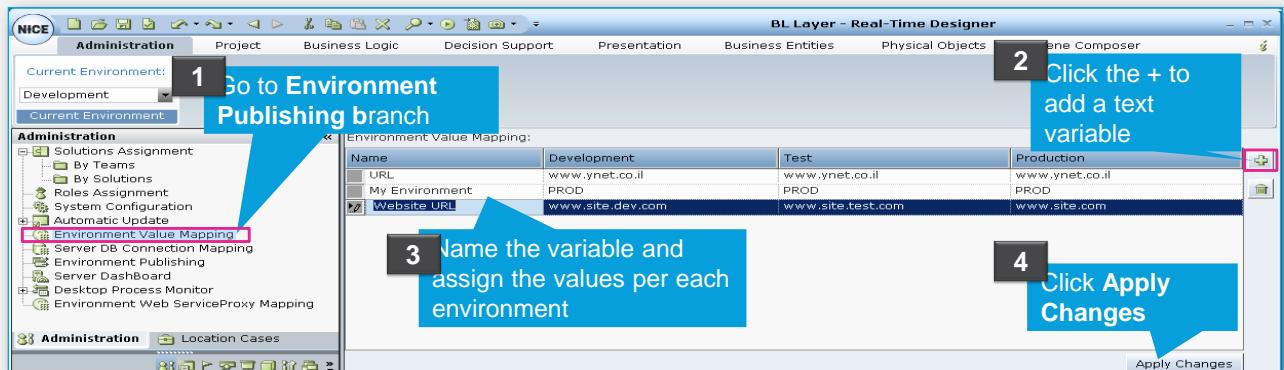
- Defines parameters that have different values, depending on the environment in use.



NICE®

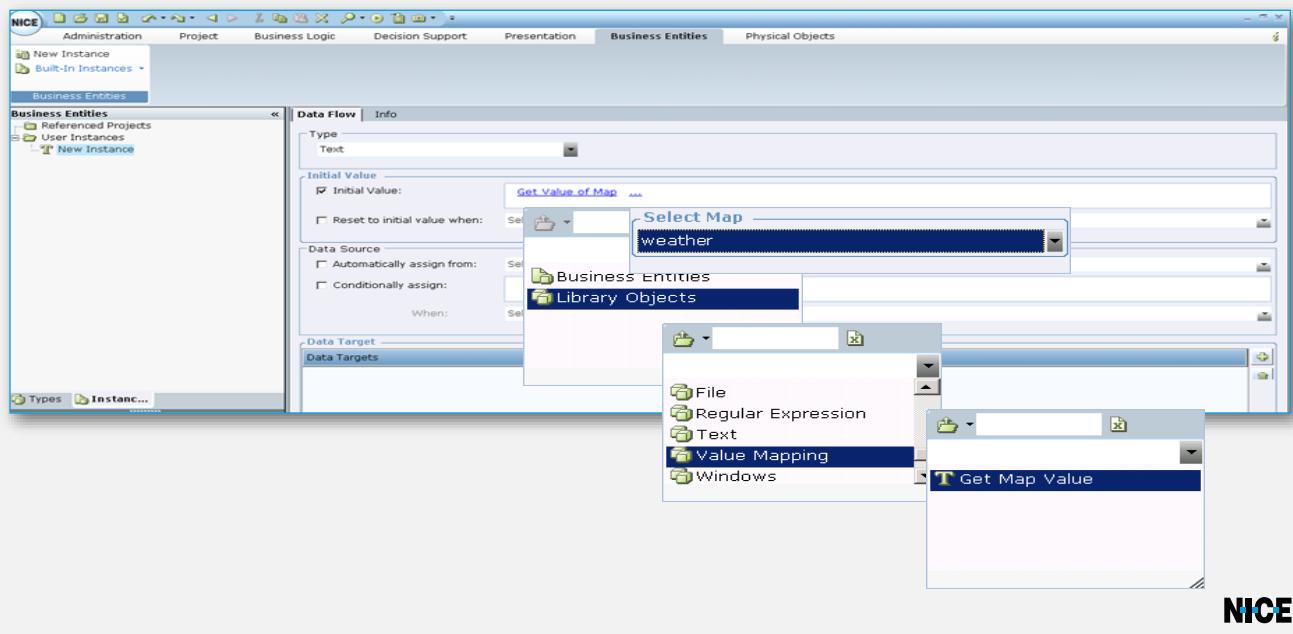
- Environment value mapping can be used for any kind of text value that you want to have distinct values in different environments.
- In effect, it maps the various values for a parameter that are used in separate environments. Because you want to be sure to access the correct URL of the third-party environment, you need to create a placeholder variable for this URL.
- This variable will get the value defined for the specific environment in use.

How to Add a Variable in the Designer



NICE®

How to Map an Environment Value



- The Environment-dependent value can be used/stored in a regular business entity.
- The Designer must be connected to the RT Server in order to use the Get Map Value function.
- During runtime on the Client, the value returned by this function is determined by the environment in which the Client is currently running.



Notes from Demo

DO IT YOURSELF

Environment Value Mapping

- Create an environment variable and allocate it token values for each environment.

NICE®

Summary

- Working with different Environments
- Mapping different DB Connections per Environment
- Mapping different text variables per Environment

NICE®

- The RT Solution supports multiple environments, and provides three default environments: Development, Test and Production.
- **Environment Publishing Feature** enables to migrate a ready-for-production Solution from the testing environment to a production environment .
- **DB Connection mapping** allows you to define a separate DB connection string for every environment used.
- **Environment Value mapping** allows you to define a variable that receives different values for every environment used.



Thank You





AUTOMATIC UPDATE



Lesson Objectives

By the end of this lesson you will be able to:

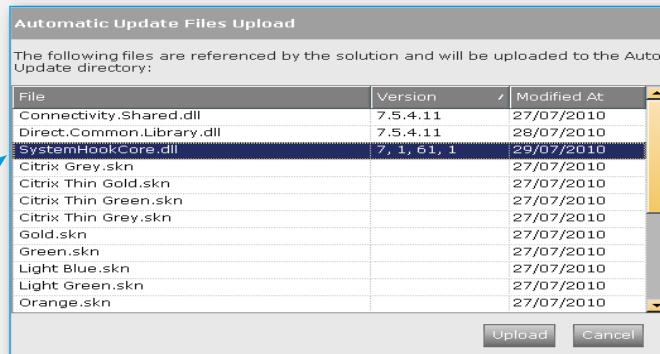
- Describe what the Auto Update process does
- Define and configure Auto Update
- Run Auto Update on the RT Client
- Verify if the Auto Update was performed successfully



What is Automatic Update?

- A generic process, which provides a means to update the RT Client with any file that is required (such as bug fixes, etc), without having to re-publish the solution or create an installation kit for that purpose
- It distributes files needed for client execution to teams of Agents, such as images and assemblies (DLLs and Callout skins)

When a user Publishes a Solution this dialog box automatically appears, which lists the DLLs and Callout skins (*.skn) used by the solution, that will be uploaded to the client



NICE®

- This method automatically uploads files used by a Solution when publishing a new version of that Solution.
- The Publish process actually scans every file that has a reference to the Project, compares them with the files in the Automatic Update folder, and if it finds a file that isn't updated, it presents a dialog box, asking the user to confirm the update.

Automatic Update Folders

- Enables you to specify files to upload to the RT Server for subsequent upload to the RT Client(s)
- Includes the following steps:



Create an Automatic Update folder

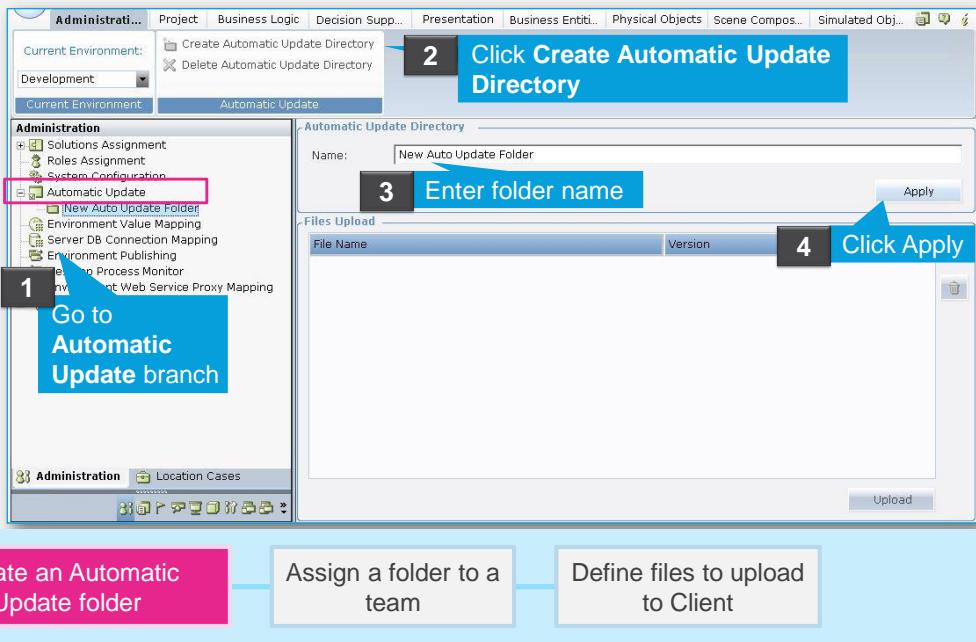
Assign a folder to a team

Define files to upload to Client

NICE®

- Automatic Update folders are saved in a folder on the SVN repository, together with all the Solution files
- For the automatic update mechanism, update files are stored in a dedicated folder for a team. By default, each team comes with a predefined folder called **Default**.
- Typically, each team uses a different folder, but some teams may use the same folder.

How to Create an Automatic Update Folder



- You can change the name assigned to a folder at a future point in time, if necessary. To do so, you must change the parameter here. You cannot change the folder name directly from the Administration tree



Notes from Demo

DO IT YOURSELF

Creating an Automatic Update Folder

- Create an Automatic Update Folder and give it a unique name

NICE®

Assigning an Automatic Update Folder to a Team

1 Go to Automatic Update branch

2 Decide whether to have the files downloaded to the Application Data folder or the Installation/Program Files folder, and then click Apply

3 Check the teams to assign a Solution

4 Select the folder you created in the previous slide

5 Click Apply

Create an Automatic Update folder → Assign a folder to a team → Define files to upload to Client

NICE®

- The folder selected applies to the team, its sub-teams and all users in them



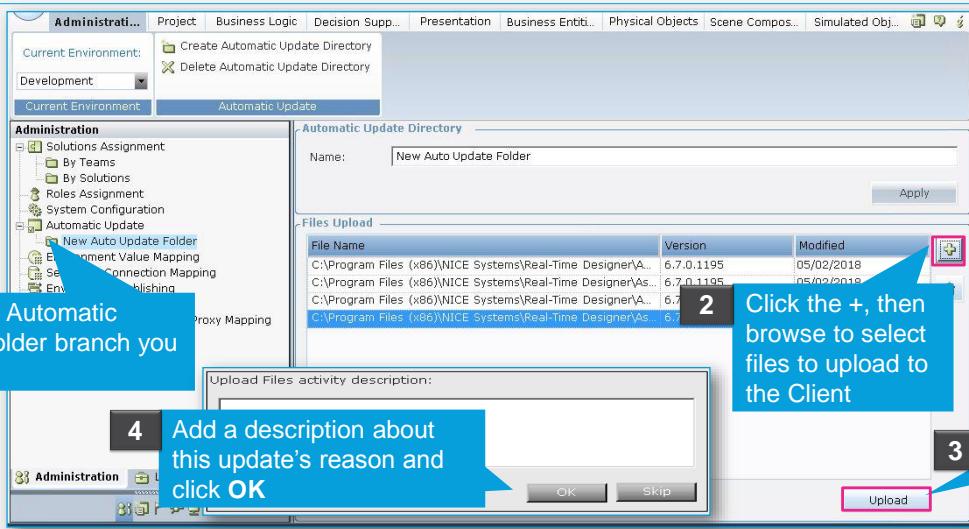
Notes from Demo

DO IT YOURSELF

- Assign your Automatic Update Folder to your team

NICE®

How to Add Files to an Automatic Update Folder



NICE®

- You may only upload files that are located under the Designer installation folder. The auto-update mechanism downloads the update files located under the RT Client installation folder
- The following information is displayed for each row in the Files Upload area:
 - **File:** Specifies the file name and path
 - **Version:** For .NET DLL files, specifies the version of the file. For other file types, this column is blank.
 - **Modified At:** Specifies when the file was last updated



Notes from Demo

DO IT YOURSELF

Adding Files to an Automatic Update Folder

- Create a file (BMP, TXT, DOC, etc.) that you intend to upload to the Automatic Update Folder
- Upload the file to the Automatic Update Folder

NICE®

Automatic Updater on the RT Client

- Updater is a standalone process that runs automatically on the client (when enabled), or manually from a command line interface
- The Updater performs the actual downloading of updated files from the SVN repository and then stores them in the correct location on the local Client machine

NICE®

- When the automatic update mechanism is enabled on the client, the Updater process automatically starts once the client is logged in
- For the Client to run manually, “**enabled**” parameter at the **Updater.exe.config** and the **RTClient.exe.config** files must be set to true

Running the Updater Manually on the RT Client

- Executing the Updater process standalone involves running it from a command line or via a scheduled task on the desktop
- Specify the values of the following mandatory parameters:

```
Updater -BU business_username -U username -P password -Team teamID
```

- Enter optional values:

```
-Url url -Retries #retries -Interval #seconds -Restart true -Progress true
```

Specifies the location of the files in the repository to be downloaded to the client

Specifies the maximum number of times to retry accessing the repository

Specifies the time between two retries, in seconds

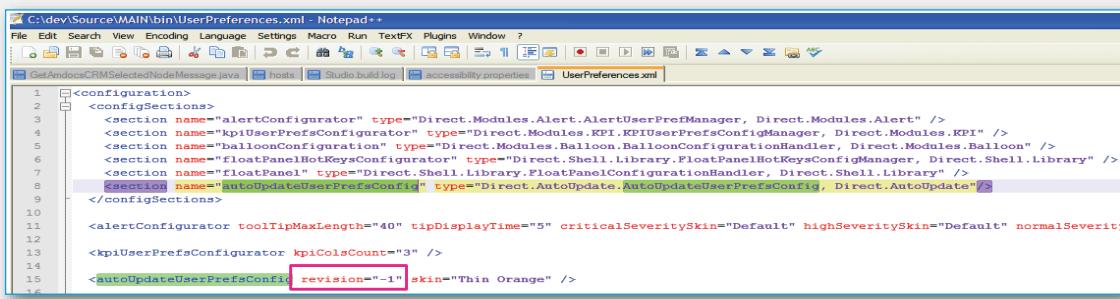
Indicates whether to restart the client after the automatic update process completes

NICE

- If only mandatory arguments are specified, then all other parameters are taken from the **Updater.exe.config** file.
- Default Values:
 - Retries: 3
 - Interval: 2
 - Restart: true
 - Progress: true
- If the automatic update succeeds, then after downloading the updated files to the client, the client restarts. During this restart, the user is not prompted to re-enter login credentials.

How to Verify Successful Automatic Update

- When performing an automatic update, data is stored about the process in a database table. If information for the process cannot be audited, the automatic update fails
- The RT Solution uses the **revision** parameter in the **UserPreferences.xml** configuration file to determine if the client has the latest updated files



```
C:\dev\Source\MAIN\bin\UserPreferences.xml - Notepad++  
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?  
GetAmdocsCRMSelectedNodeMessage.java hosts Studio.build.log accessibility properties UserPreferences.xml  
1 <configuration>  
2   <configSections>  
3     <section name="alertConfigurator" type="Direct.Modules.Alert.AlertUserPrefManager, Direct.Modules.Alert" />  
4     <section name="kpiUserPrefsConfigurator" type="Direct.Modules.KPI.KPIUserPrefsConfigManager, Direct.Modules.KPI" />  
5     <section name="balloonConfiguration" type="Direct.Modules.Balloon.BalloonConfigurationHandler, Direct.Modules.Balloon" />  
6     <section name="floatPanelHotKeysConfigurator" type="Direct.Shell.Library.FloatPanelHotKeysConfigManager, Direct.Shell.Library" />  
7     <section name="floatPanel" type="Direct.Shell.Library.FloatPanelConfigurationHandler, Direct.Shell.Library" />  
8     <section name="autoUpdateUserPrefsConfig" type="Direct.AutoUpdate.AutoUpdateUserPrefsConfig, Direct.AutoUpdate" />  
9   </configSections>  
10  </configuration>  
11  
12  <alertConfigurator toolTipMaxLength="40" tipDisplayTime="5" criticalSeveritySkin="Default" highSeveritySkin="Default" normalSeveritySkin="Default" />  
13  <kpiUserPrefsConfigurator kpiColsCount="3" />  
14  <autoUpdateUserPrefsConfig revision="-1" skin="Thin Orange" />  
15  
16
```

NICE

- Initially, the revision parameter value in this file is set to -1. When an automatic update is performed successfully, this value is incremented. This value is a running number that is typically not reset. It indicates the last loaded revision on the client
- To determine whether the client has the latest version, this value is compared to a similar value on the server. If you want to force an automatic update on the client, you must open the UserPreferences.xml file on the client and edit it to set the revision number to -1.



Notes from Demo

DO IT YOURSELF

Verifying Successful Automatic Update

- Open the Application Data folder
- Open the UserPreferences.xml
- Run the RT Client and notice:
 - The file you uploaded appear in the Application Data folder
 - The revision parameter update

NICE®

How to Execute an Automatic Update Report

3 Filter the report by Auto Update Directory...

4 ... by Team... For each filter, enter a keyword(s)

Auto Update Directory	Agent Team	Computer Name	Revision	Update Time
Default	DEV	DAVIDXP1	48	Feb 23, 2012 12:46:04 PM
Default	DEV	DAVIDXP1	47	Feb 23, 2012 12:45:30 PM
Default	DEV	DAVIDXP1	46	Feb 23, 2012 12:42:29 PM
Default	DEV	DAVIDXP1	45	Feb 23, 2012 11:41:43 AM
Default	DEV	DAVIDXP1	45	Feb 23, 2012 11:40:17 AM
Default	DEV	DAVIDXP1	44	Feb 23, 2012 11:38:18 AM
Default	DEV	DAVIDXP1	43	Feb 23, 2012 11:36:51 AM
Default	DEV	DAVIDXP1	41	Feb 22, 2012 2:09:45 PM
Default	DEV	DAVIDXP1	40	Feb 22, 2012 1:58:11 PM
Default	DEV	DAVIDXP1	38	Feb 22, 2012 12:55:46 PM
Default	DEV	DAVIDXP1	35	Feb 22, 2012 12:49:16 PM
Default	DEV	DAVIDXP1	34	Feb 22, 2012 12:43:38 PM
Default	DEV	KOBSTERXP2	31	Feb 9, 2012 7:23:10 AM
Default	DEV	KOBSTERXP2	31	Feb 9, 2012 7:21:13 AM
Default	DEV	KOBSTERXP2	31	Feb 9, 2012 5:54:41 PM
Default	DEV	KOBSTERXP2	31	Jan 22, 2012 11:25:50 PM
Default	DEV	DAVIDXP1	31	Jan 19, 2012 10:17:01 AM
Default	DEV	DAVIDXP1	30	Jan 17, 2012 7:57:52 AM
Default	DEV	KOBSTERXP2	30	Jan 17, 2012 7:54:57 AM
Test1 Team	GroupA	KOBSTERXP2	28	Jan 16, 2012 2:49:05 PM

- **Automatic Update Report** displays information about recent automatic update downloads across the users of the current environment.
- To execute an Automatic Update Report:
 1. Click on **Server Dashboard Reports**
 2. Select **Automatic Update**
 3. You can filter the report by Auto-Update Directory, Team, Revisions or Computer Name
 4. For each filter, enter a keyword(s) and search for it, then select the relevant results and insert them as part of the filter
 5. Clicking **Finish** will run the report, and the results will appear on your screen.



Notes from Demo

DO IT YOURSELF

Automatic Update Report

- Execute an Automatic Update Report, filtering only on your Team

NICE®

Auto Update Limitation

- The following core files cannot be pushed through the auto-update mechanism:

- Direct.AutoUpdate.dll
- Direct.Shared.Controls.dll
- Direct.Configuration.dll
- Direct.ExecutionResources.dll
- Direct.WebServiceProxy.dll
- Direct.Shared.dll
- Direct.Interface.dll
- Direct.Shared.Library.dll
- Updater.exe
- Updater.exe.config
- Userpreferences.xml

NICE®

- These files are used by the update mechanism itself.
- Since the files being used by the process cannot be updated themselves by the process that is using them, the only alternative to updating these files specifically is to create an MSI, batch file, or any other customized mechanism that performs the same task.

Summary

In this lesson we covered:

- What the Auto Update process does
- Defining and configuring Auto Update
- Running Auto Update on the RT Client
- Verifying if the Auto Update was performed successfully

NICE®

- Automatic Update provides a means to update the RT Software without having to create an installation kit for that purpose.
- It distributes files needed for client execution to teams of Agents, such as images and assemblies (DLLs and Callout skins).
- Manual update includes the following steps:
 - Create an Automatic Update folder
 - Assign a folder to a team
 - Define files to upload to Client
- Updater process performs the actual downloading of updated files from the SVN repository and then stores them in the correct location on the local Client machine.
- Successful Automatic Update is verified through:
 - Auditing in DB
 - Revision parameter in UserPreferences.xml configuration file



Thank You





Contact us: training@nice.com
Visit us: [Technical training on ExtraNICE](#)

PROPRIETARY AND CONFIDENTIAL INFORMATION

Information herein is proprietary information and trade secrets of NICE Systems Ltd (NICE) and/or its affiliated companies (Affiliates). This document and the information herein is the exclusive property of NICE and its Affiliates and shall not be disclosed, in whole or in part, to any third party or utilized for any purpose other than the express purpose for which it has been provided.

IMPORTANT NOTICE

Subject always to any existing terms and conditions agreed between you and NICE or any Affiliate with respect to the products which are the subject matter of this document, neither NICE nor any of its Affiliates shall bear any responsibility or liability to a client or to any person or entity with respect to liability, loss or damage caused or alleged to be caused directly or indirectly by any product supplied or any reliance placed on the content of this document. This includes, but is not limited to, any interruption of service, loss of business or anticipatory profits or consequential damage resulting from the use or operation of any products supplied or the content of this

document. Information in this document is subject to change without notice and does not represent a commitment on the part of NICE or any Affiliate.

All information included in this document, such as text, graphics, photos, logos and images, is the exclusive property of NICE or in Affiliate and is protected by United States and international copyright laws. Permission is granted to use, view and photocopy (or print) materials from this document only in connection with the products to which this document relates and subject to the terms of license applicable to such products. Any other use, copying, distribution, retransmission or modification of the information in this document without the express prior written permission of NICE or an Affiliate is strictly prohibited. In the event of any permitted copying, redistribution or publication of copyrighted material, no changes in, or deletion of, author attribution, trademark legend or copyright notice shall be made. Products supplied may be protected by one or more of the US patents listed at www.nice.com/Patents. For the full list of trademarks of NICE and its Affiliates, visit www.nice.com/Nice-Trademarks. All other marks used are the property of their respective proprietors.

All contents of this document are: Copyright © 2019 NICE Systems Ltd. All rights reserved.

NICE®