blueprism®

Advanced Data Items

TRAINING GUIDE

Version 1.00

Blue Prism Limited, Cinnamon Park, 2 Crab Lane, Warrington, WA2 0XP, United Kingdom
Registered in England: Reg. No. 4260035. Tel: +44 370 879 3000. Web: www.blueprism.com

Contents

1 Training Overview

This module is intended to supplement the Blue Prism® Foundation Training course and is aimed at students who have completed the course and are beginning to put their learnings into practice.

The module will look at some additional Data Item features available in Blue Prism.

1.1 Prerequisites

- You have completed Blue Prism Foundation Training, including the Consolidation Exercise.
- You have downloaded the **Training Resources – Advanced Data Items** file from the course content and extracted the files.
- You have a login for your local Blue Prism training environment and a valid Blue Prism license.
- You have installed the following utility Business Objects into your training environment: *
    - **Utility – General**
    - **Utility – File Management**

* Utility Business Object installation steps are provided in the next section of this training document

Activity 1.1.1 Install the utility Business Objects

- From the Blue Prism menu, select File followed by Import.
- Click Browse and navigate to the folder containing the training resources downloaded in section 1.1.
- Open the folder named VBO.
- Select the file named **BPA Object - Utility - General.xml** and press open.
- Click through the install screens until the Business Object has been installed.
- Repeat the steps but this time select **BPA – Utility – File Management.xml**.

1.2 Learning Objectives

On completion of this module, you will

- understand more about working with Collections;
- understand more about Data Item exposure and initialization;
- understand the difference between standard Data Items, Environment Variables and Session Variables;
- understand the concept of Data Casting.

2 Undefined Collections

You will have used the Collection data type throughout Blue Prism Foundation training and Consolidation exercise, which until this stage would have been **defined,** meaning that the Collection column names, and data types have been configured in advance using the Collection Properties window.

When using a Collection to receive an Output Parameter from another Page or Business Object action, it is possible and perfectly acceptable to leave the Collection columns undefined, i.e., with no column names or data types assigned. You may have already used an undefined Collection as part of exercise 9.1.1 in Foundation training.

When a Collection is undefined, the **Fields** tab and the **Initial Values** tab within the Collections Properties window will appear blank; an example of this is below:

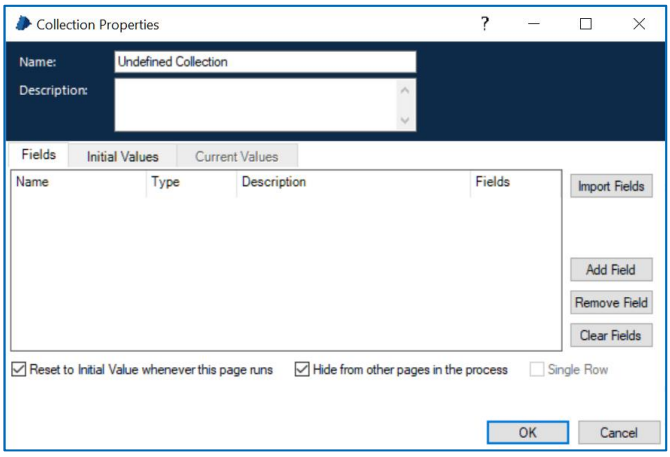

Figure 1: Collection Properties Window: 'Fields' view

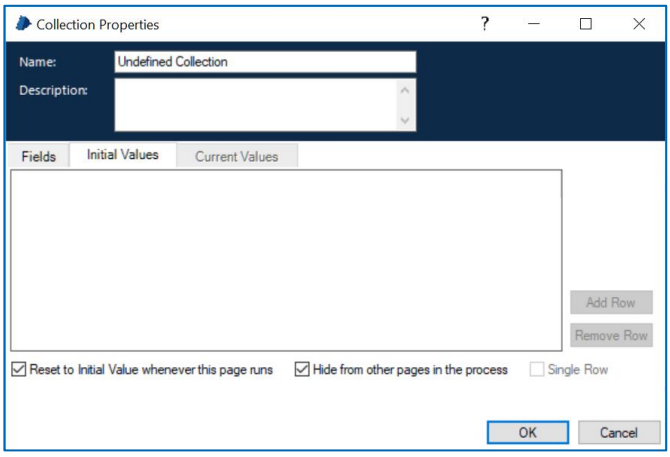

Figure 2: Collection Properties Window: 'Initial Values' view

Key Points

- A Collection can either be fully defined, or undefined; you cannot configure selected columns in a Collection, and not the remaining columns of the same Collection.
- Undefined Collections are very useful when reading varying data sources.
- Collections are the only kind of Data Item that can behave in this way. All other Data Items must be predefined with a data type.

Activity 2.1.1 Using an Undefined Collection

- Create a new Process named **Undefined Collection Exercise**.
- Add a new Data Item named Data File Path and set the Data Type to Text.
- Set the Initial Value of the new Data Item to the location of the file named Names.csv. This should be:
  - C:\BluePrism\Training\Resources\Exercise Files\Advanced Data Items\Names.csv
- Add a new Collection named **Data Collection.**
- Leave the Fields and Initial Values tabs empty.
- Add an Action Stage and then update the Properties window to match the examples below:

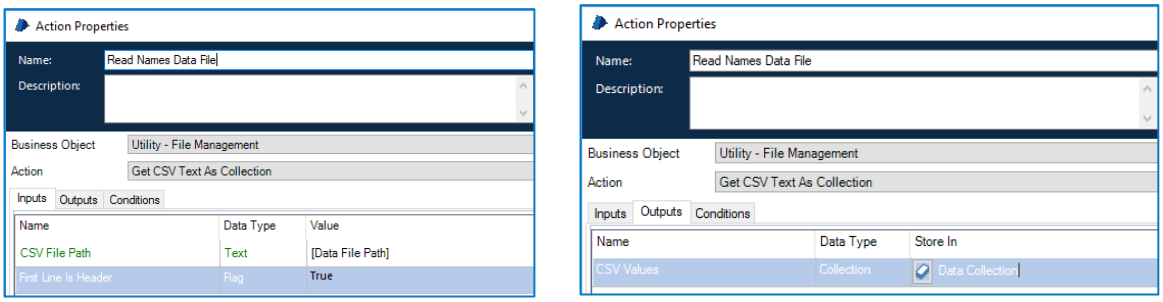

Figure 3: Action Properties Window – Inputs and Outputs configured

- Connect the stages up. Your Process diagram should look like the example below:

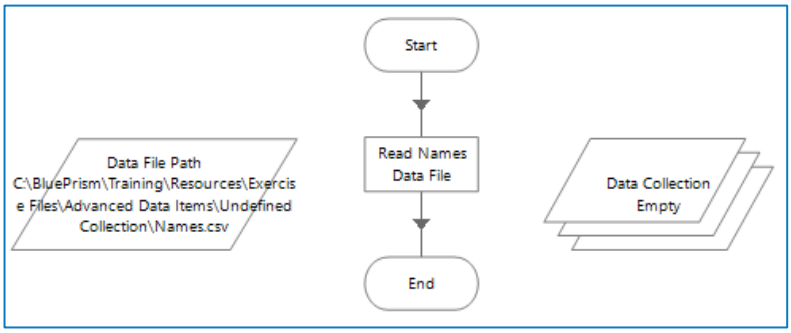

Figure 4: Example Process diagram

- Reset, save and run your Process.
- Open the Data Collection Properties window and observe the **Current Values** tab.

Notice how the Field names have been set at runtime using data from the Names.csv.

3 Data Item Initialization

We have already looked at how a data item can be given an initial value, and that this default value will also be used as the current value when the process starts. By default, a data item is only visible to the page it is added to, however, this can be changed by un-checking the **Visibility** check box on the data item Properties window, which will expose the data item globally across the process.

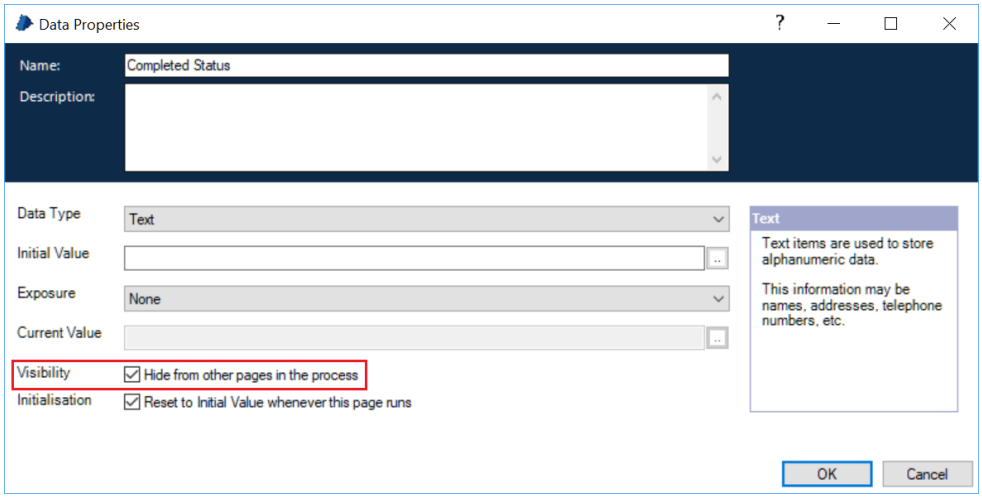

Figure 5: Data Item Properties Window: 'Visibility' property

Just below the Visibility checkbox is another checkbox named 'Initialisation'. This controls how the current value of a data item is affected when its page is run.

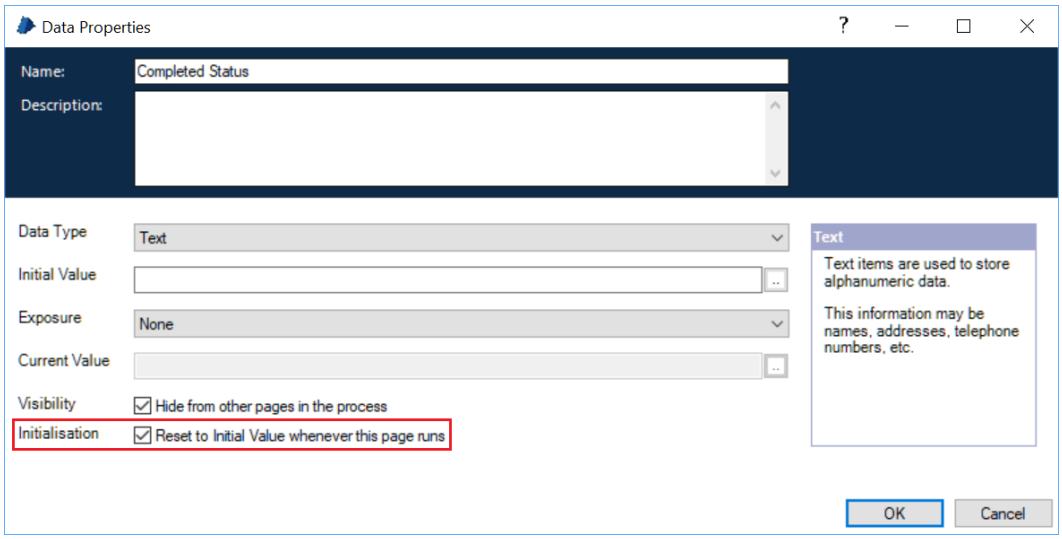

Figure 6: Data Item Properties Window: 'Initialisation' property

By default, the current value of the data item is reset to the initial value when the Start stage of the Page is reached. When the checkbox is un-checked, the current value is maintained and not reset. Both settings have their advantages, depending on the circumstances.

For example, global Data Items can be placed on a Page that simply links the Start stage to an End stage. Although the page will appear to do nothing, it can be used to ensure that the global Data Items are put back to their initial values simply by running the "Global Data Items" page.

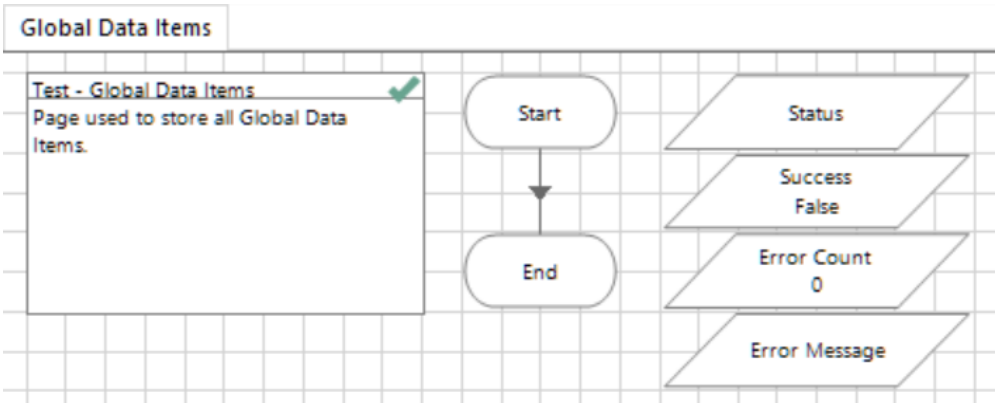

Figure 7: Resetting Global Data Items

4 Data Item Exposure

Data Items can have their values controlled from **outside** the diagram - this is done by **exposing** a Data Item. By default, Data Items are not exposed but we will look at how that can be changed.

4.1 Environment Variables

An Environment Variable is a value that is made available to all Processes and Business Objects, i.e., across the environment; it is exposed outside of the process diagram. An Environment Variable appears in the process like a normal data item; however, the data item exposure has been changed through the Properties window.

Environment Variables should be used to store information which is required by the Process to run but this information is subject to change. For example, the URL of an online application will differ in the development environment to that of the production environment.

The reason to use Environment Variables is so that minor configuration changes can be made without making changes to the solution design in Process and Object Studio.

Example Use Case for Environment Variables:

- The URL for a Browser based application.
- The file path used to collect data from a file.
- The file path to save completion reports to.
- Business application regular scheduled outage period.
- Process owner email addresses for alerts.
- Global timeout values for an application so they can easily be amended across all processes.
- The System or Environment Name to select when logging into an application (dev/test/prod).
- A Whitelist of email addresses which are authorized to send email updates the process.

4.2 Environment Variable Naming

It is important to name Environment Variables so that the process or system they are related to is obvious. This will make searching for and amending an Environment Variable easier.

The correct naming convention to use should be part of your own organisation best practices, but usually the name of an Environment Variable is prefixed with the name of the process or application the variable is related to.

For better organisation, the name might also be prefixed with the organisation department or global region to which the variable belongs. If an environment variable is named 'Excel File Path' it will be meaningless in the wider context of an environment containing hundreds of completed processes across a large organisation.

Some examples of Environment Variable names:

- "Fraud Account Closures File Path" – this variable name includes the department name.
- "SAP System Name" – this variable name includes the application name.
- "CRM System URL" – this variable includes the application name.
- "NA_HR Holiday Requests File Path" – this variable name includes region, department, process name.

4.3 Creating an Environment Variable

Before an Environment Variable can be included in your Process of Object diagram, it must be setup in the System area of Blue Prism.

Activity 4.3.1 Creating an Environment Variable

- Navigate to the **System** area of Blue prism, and then locate the **Processes** section.
- Select the **Environment Variables** sub-item.

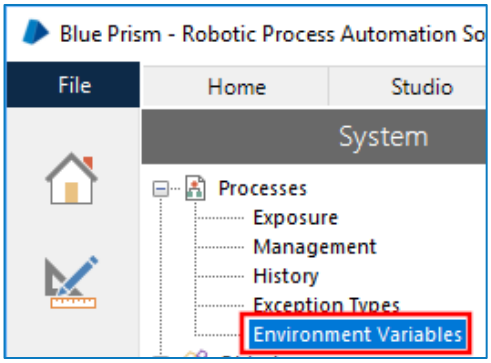

Figure 8: System – Environment Variables

- Any existing Environment Variables will be listed and can be edited from here.
- Click on the Add Variable link. A new row will be appended to the bottom of the list of existing items, which you can now edit.

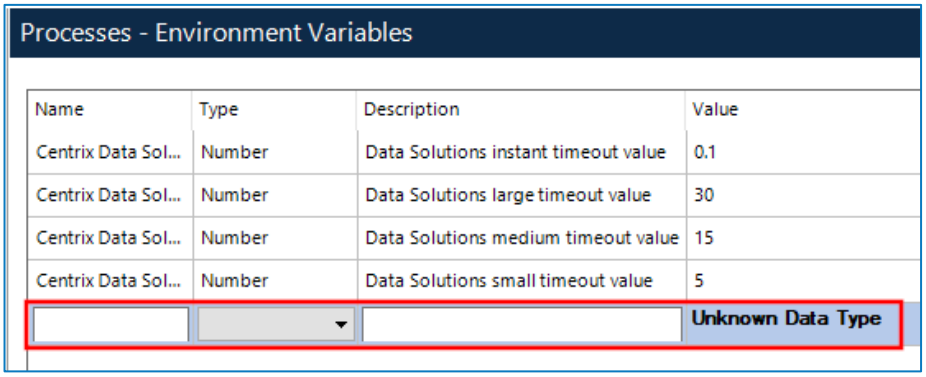

Figure 9: System – New Environment Variable

- Set the **Name** column to **Stopping Time**.
- Select **Time** from the Type column and update the **Description** column.
- Set the **Value** column to any time value of your choice.
- Click Apply to save your changes.

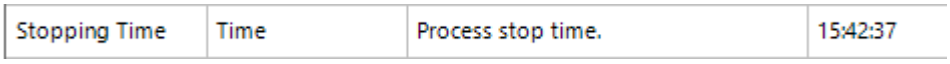

Figure 10: System – New Environment Variable Saved

We will use this environment variable in the next exercise.

4.4 Using an Environment Variable

Once an Environment Variable has been created it is available for use in Process and Object Studio. If you already had either of the Studio areas open when creating the new variable, you will need to hit refresh.

Activity 4.4.1 Using an Environment Variable in a Process

For this exercise we will use the **Sleep** Action which is found in the **Utility – General** Business Object we imported earlier. In you have not imported this Business Object, take a look at the Prerequisites section of this training guide.

- Create a new Process named **Stopping Time Exercise**.
- Add a new Data Item to your diagram and open the Properties window for the new item.
- Set the name field to **Stopping Time**.
- From the **Exposure** list, select **Environment – Read the corresponding Environment Variable from System Manager**. The updated Properties window is illustrated below:

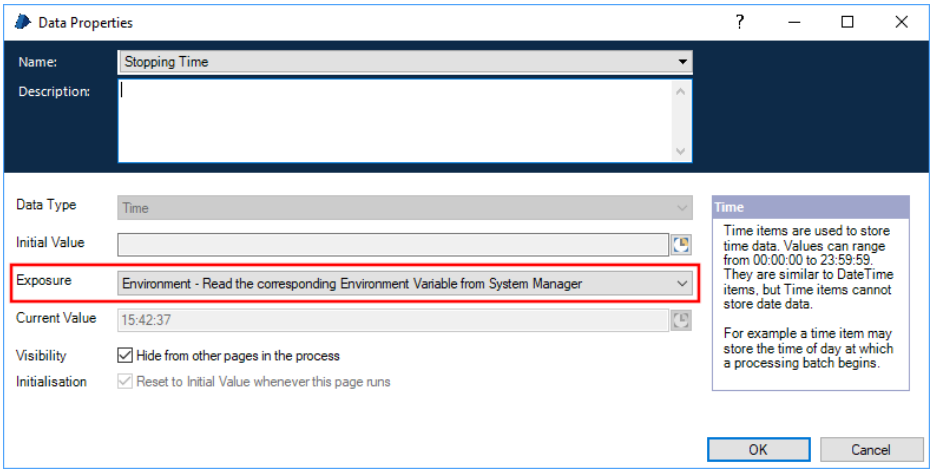

Figure 11: Process Studio – Stopping Time Data Item with Exposure set to Environment

You will notice that the Data Item Name field has been replaced with a list. If you are unsure of the exact name of your Environment Variable, you can use this list to select the relevant item.
You must update the Exposure field to Environment before the Name list is made available.

We want to use a **Sleep** Action if the **Stopping Time** is not reached. This means the Process can pause and wait for a defined period of time.

- To use a **Sleep** Action, add an Action stage to your Process diagram.
- Open the Action stage Properties window and select **Utility – General VBO** from the Business Object list.
- Now select **Sleep** from the Action list.
- You will see there is a number **Input** named **Sleep Time (Seconds)**. Set this to 10 seconds.

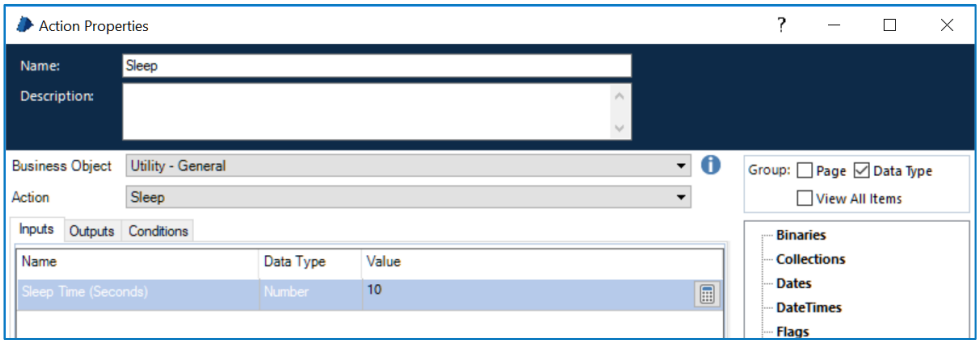

Figure 12: Setting up the Sleep action

Based on the configuration of our new Action stage, the Process will pause for 10 seconds before proceeding to the next stage of the Process.

We need the Process to check if the Stopping Time has been met, therefore we need to construct a loop in the process. The loop needs to check if the Stopping Time has passed; if not the Process will need to sleep for a further 10 seconds.

- Add a Decision stage before the Sleep Action to check if the current local time is passed the Stopping Time value. You could use the Expression LocalTime() >= [Stopping Time].

- Connect all the stages. Your process should look similar the example below:

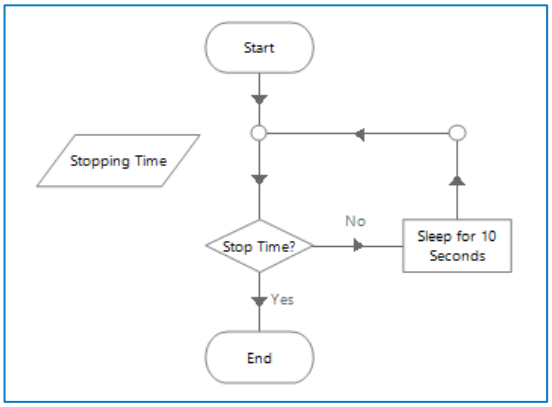

Figure 13: Example Stopping Time process

- Publish and save your Process, and then close Process Studio.
- Amend the value of the Stopping Time Environment Variable through the System area of Blue Prism, so the value is several minutes in the future.
- Run the Process and check it stops at the appointed time, which is time value stored in your Stopping Time Environment Variable.

Activity 4.4.2 Environment Variable – Value Update

Environment Variables are centralized shared variables which can be accessed by all Process and Business Objects in the same environment.

In this exercise, we will attempt to update the value of a Data Item exposed as an Environment Variable from within the Process diagram.

- Open your Stopping Time Exercise Process.
- Add a Calculation stage and attempt to change the value of Stopping Time Environment Variable through an Expression.
- Press the Validation button and observe the error message.

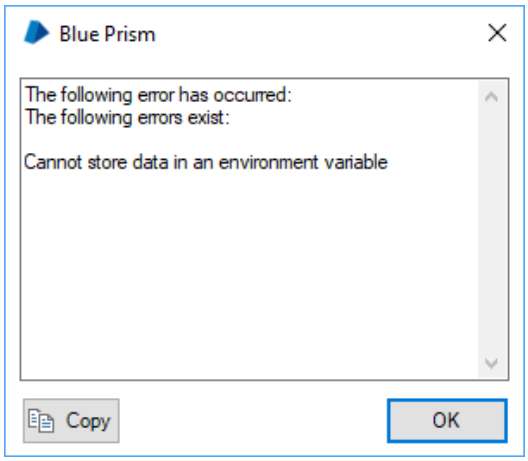

Figure 14: Error message – attempting to change the value of a Data Item exposed as an Environment Variable

Data Items exposed as Environment Variables are read-only, and therefore their values cannot be updated from within a Process or Object diagram. Environment Variable values can only be changed from within the System area of Blue Prism. Note that any change made won't be reflected in any Process which is running whilst the change is made. The change will be picked up when the Process is next run.

Key Points

- Environment Variables are available to all Processes and Business Objects in the same environment.
- Data Items exposed as Environment Variables are **read-only**.
- Environment Variables value changes will only be reflected in Processes which have been started in Control Room after the change was applied.

4.5 Session Variables

Like Environment Variables, Session Variables are exposed outside of the process diagram, and as the name suggests are applicable to running Process sessions. Session Variables appear in the process like a normal Data Item; however, the Data Item Exposure has been changed through the Properties window.

Data Items exposed as Session Variables can be updated from within the Process diagram, and also from Control Room when a Process is running. A Session Variable should be considered as an option if you have a solution where communication is required from the Controller team to the running Process session. If there is a configuration that may need changing during the working day whist the solution is in use, the Session Variable allows it to be done without the need to stop, change or restart the running session or having multiple versions of the same Process.

The table below provides example use case for Session Variables:

Stopping After Time	The Initial Value could be set the time the Process should stop running. It might be set to a time before a system used by the Process is no longer available, or a time when another Process needs to be started instead. This Session Variable can be changed by a Blue Prism Controller whilst the Process is running, if there is an operational reason that a running session should finish at a different time to the default.
Stop After Items	This Data Items allows the developer to configure the process to only work a pre-defined number of cases. The Data Item could have and Initial Value set to 9999999999, which effectively means that the process will work all items. This Session Variable can be changed by a Blue Prism Controller at run time if there is an operational reason to only work a set number of cases before completing.
Stop ASAP	This session variable can be changed from False to True by a Blue Prism Controller to stop the process. If set to True the process will stop cleanly, if a case is currently being worked it will be completed before stopping. **NOTE: The Stop ASAP Session Variable has been superseded in Version 5 of Blue Prism with the Request Stop Control Room functionality and IsStopRequested() function.**

Activity 4.5.1 Creating a Session Variable

For this exercise, we will work again on the Stopping Time Exercise process we created earlier.

- Open the **Stopping Time Exercise** Process.
- Open the Properties window of the **Stopping Time** Data Item and change the Exposure to **Session – Expose the data item to Control Room**.
- Set the Initial Value of the Data Item to 23:00:00.

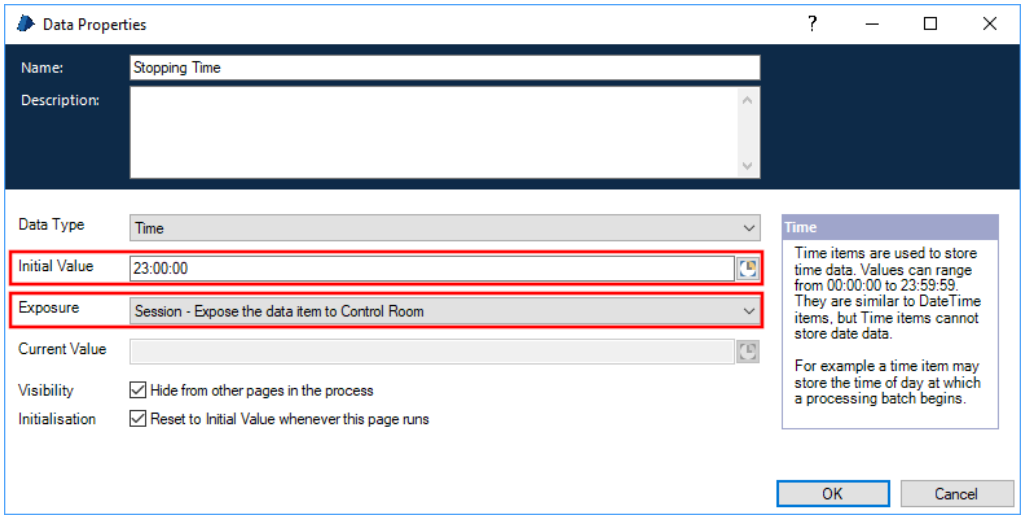

Figure 15: Process Studio – Stopping Time Data Item with Exposure set to Session and Initial Value set to 23:00:00

- Save the changes to the Process and close the Process.

We will use this Session Variable in the next exercise.

Activity 4.5.2 View and Update Session Variable Values

- Start two sessions of the **Stopping Time Exercise** Process in Control Room. You can run both these sessions on a single Resource PC.

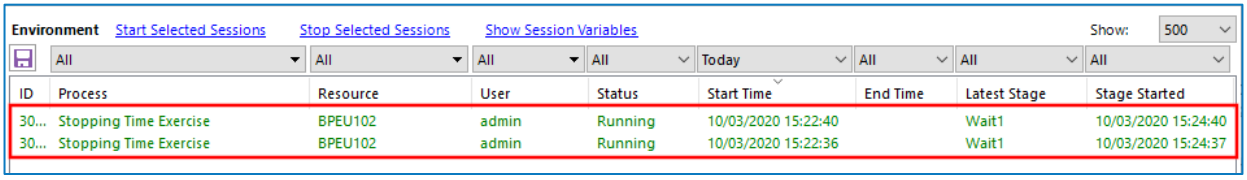

Figure 16: Control Room: 2 sessions of Stopping Time Exercises running on a single Resource PC

- Select 1 of the Running Process sessions in Control Room to observe any Session Variables and their associated values which are used in the Process. Click **Show Session Variables** to reveal the Session Variables.

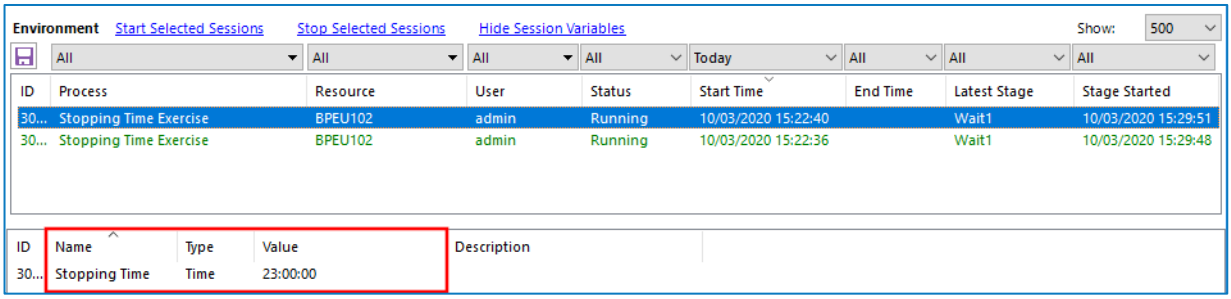

Figure 17: Control Room: Session Variable current value revealed

- Modify the Stopping Time Session Variable for the selected session to be in 10 seconds time. To do this right mouse click on the Session Variable and select **Edit Value**.

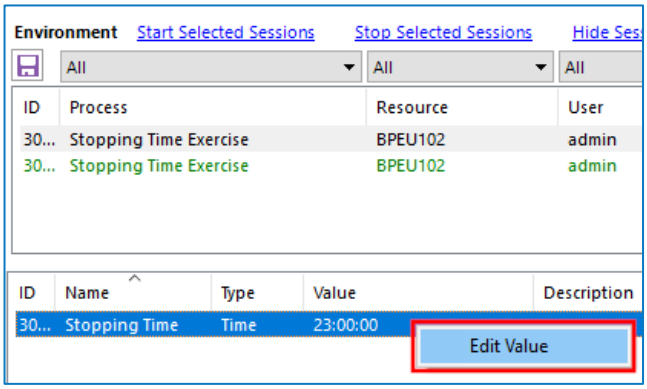

Figure 18: Control Room: Edit Session Variable value.

- A new Sessions Variable window will open. Update the Stopping Time value and click **Modify**.

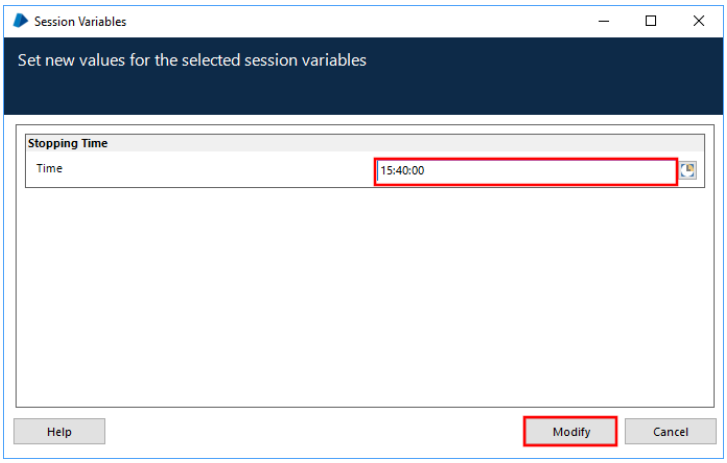

Figure 19: Control Room: Session Variable Edit Window

Notice how the Process session we selected and updated the Stopping Time Session Variable on, has been influenced by the change, and the Process session has completed. Also, the second Process session has been unaffected by the change. This is because Session Variables are associated with running Process sessions.

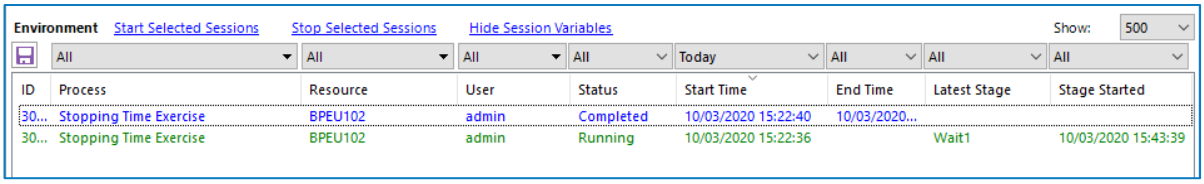

Figure 20: Control Room: Process session outcome influenced by a Session Variable update

Key Points

- Session Variables are specific to an instance of the Process. If two instances of the same Process are running at the same time, they will both have the same named Session Variables assigned to the session, however the Session Variable value may have different values as it is associated to that specific session.
- Session Variables need no setup in the System area of Blue Prism.
- Data Items exposed as Session Variables are writable.
- Session Variables can be viewed and modified from Control Room.

5 Data Type Casting

Data type Casting is when a value is stored as one data type and is automatically transformed or converted to another data type.

An example of data type Casting is converting a number value which is stored as text, to a number data type. This might be done to enable the comparison of two numbers, when one number is saved as text and the other as a number.

Casting in Blue Prism is performed using a Calculation stage.

Activity 5.1.1 Casting using a Calculation Stage

- Create a new Process named **Casting Exercise**.
- Add a Calculation stage to the Main Page and set the Expression to 123.
- Press the Evaluate button to see the message **Expression Result = 123 (number)**.
- Now change the Expression to "123" (i.e., with quotes).
- Evaluating the expression will now display the message **Expression Result = 123 (text)**. Why is this?
- The first expression is a number, but the second expression is not, it is text containing numerical characters.
- Now change the Calculation stage expression to "True" (i.e., with quotes) and press evaluate.
- And then change the expression to True (i.e., without quotes) and evaluate again.

What is happening? The answer is that the expressions are resulting in different data types.

We have mentioned that a Data Item must have a data type, and that data type must be set up in advance and cannot change when the Process runs. However, it is possible to **"force"** a Data Item to accept a value of a different data type. When faced with a different data type, a Data Item will attempt to interpret the value as its own data type.

Activity 5.1.2 A Casting Calculation

- Got back to your Calculation Exercise Process and add a Data Item with the Data Type of Number.
- Change the Expression in the Calculation stage "123" (with quotes). We know the result will be text but assign the new number Data Item to the 'Store In Result' field anyway.
- Connect the Start stage, the Calculation stage and End stage.
- Step through the Process diagram and observe how the number Data Item will be assigned the number value of 123 (number).

The text result has been automatically converted into a number so that it can be stored in the number Data Item. **This is casting.**

- Change the Calculation stage Expression to "abc" (with quotes), and step through the Process again.

This time, the Expression result could not be interpreted as a number and an Exception is thrown.

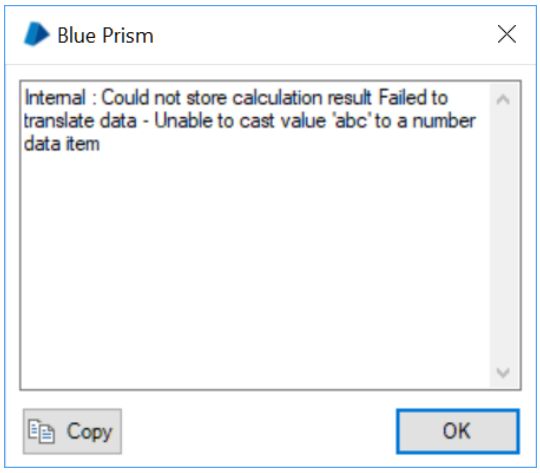

Figure 21: Data Type Casting Exception Message

This is because only numeric characters can be stored in a data item with a data type of numeric.

It is important to realize that although casting can be very useful, it should be used with care. Casting is not infallible; sometimes the translation cannot be made, causing an exception.

The most common practical use of casting is in the handling of application screen data. Normally these data are read and written as text but it is likely that some of it will need to be cast to another data type at some point.

Casting can be done automatically by changing the element data type in Application Modeler. This will change the data type of the inputs and outputs of read and Write stages that use that element. However, this setting should only be changed with careful consideration, as it depends on the casting operation always succeeding.

For example, suppose an element named Account Balance was given the number data type. A read stage using this element would then have a number output parameter. This would be fine as long as the application always showed a number in that field. If the application ever displayed a non-numerical value (such as "1.23DR", "£???" or "ERR"), then the read stage would create an Exception because it failed to cast.

These kinds of problems are common, and it is recommended that data are read from and written to application as text. This simplifies moving data to and from the application and allows any necessary casts and checks to be explicitly shown in the diagram.