Critical manufacturing
an ASM PT company

SECS/GEM

11.3

April 2026

DOCUMENT ACCESS

Public

SECS/GEM Integration Tutorial

Estimated time to read: 21 minutes

This tutorial covers the integration of a secs/gem interface to update the resource state (using SEMI E10 model) in MES. Updating the state in MES will allow OEE calculation.

> **Note**
>
> During this tutorial, `Automation Manager` will run in console mode in order to highlight the most important events when they occur.

SECS/GEM Communication Standard

Overview

SECS/GEM (SEMI Equipment Communication Standard / Generic Equipment Model) is a standardized communication protocol used extensively in the semiconductor industry to enable interoperability between semiconductor manufacturing equipment and factory host systems. Developed and maintained by SEMI (Semiconductor Equipment and Materials International), SECS/GEM allows for real-time equipment monitoring, control, data collection, and process automation.

SECS/GEM is built upon several key SEMI standards:

- **SEMI E37**: High-Speed SECS Message Services (HSMS) - the primary communication protocol.
- **SEMI E37.1**: HSMS-SS (Single Session) - a simplified version of HSMS designed for point-to-point communication.
- **SEMI E4**: SECS-I (SEMI Equipment Communications Standard - I) - an older protocol using serial communication.
- **SEMI E5**: SECS-II - defines message structures and data formats.
- **SEMI E30**: GEM - outlines the Generic Equipment Model, defining equipment behavior, capabilities, and state models.

Transport Protocols

SECS/GEM supports two primary communication methods:

HSMS (High-Speed SECS Message Services)

- **Standard:** Defined in SEMI E37 and E37.1.
- **Transport:** TCP/IP over Ethernet networks.
- **Mode:** Full-duplex communication using socket connections.
- **Performance:** High-speed, low-latency, and scalable, suitable for modern semiconductor fabs.
- **Connection:** Uses a client-server model. Typically, the host acts as the client, and the equipment acts as the server.
- **Session Control:**

- Establishes a persistent connection between host and equipment.
- Supports message acknowledgment and ensures data integrity.

SECS-I (SEMI Equipment Communications Standard - I)

- **Standard:** Defined in SEMI E4.
- **Transport:** RS-232C serial communication (or RS-422/485 for extended range).
- **Mode:** Point-to-point, half-duplex communication.
- **Performance:** Lower speed and shorter range compared to HSMS, but still used for legacy equipment.
- **Message Framing:** Uses start and end-of-block characters, checksums, and message headers to ensure reliable data transmission.

Message Structure and Data Formats

SECS/GEM messages are based on the SECS-II standard (SEMI E5), which defines both the message structure and supported data types. Each message is identified by:

- **Stream (S):** Represents a category of related messages.
- **Function (F):** Identifies a specific message within a stream.

For example, `S1F1` is a "Are You There?" request, and `S1F2` is the corresponding "On-Line" reply.

Supported Data Types

- **Integer:** 1, 2, 4, or 8 bytes (signed or unsigned).
- **Floating Point:** 4 or 8 bytes (IEEE 754 format).
- **ASCII:** Variable-length character strings.
- **Boolean:** 1 byte (true or false).
- **Lists:** Hierarchical, nested lists that can contain other data elements or lists.

Message Acknowledgment

SECS/GEM uses a request-reply mechanism to ensure message delivery. Each primary message (request) is followed by a secondary message (reply) confirming receipt or providing requested data.

GEM (Generic Equipment Model) Functionality

The GEM standard (SEMI E30) defines the behavior and capabilities that equipment must provide, ensuring consistency and interoperability across different vendors. Key features include:

Host-Initiated Control

- **Remote Commands:** Start, stop, pause, resume, and abort equipment operations.
- **Recipe Management:** Upload, download, and select recipes for production processes.
- **Correlation with Streams:**
  - **Stream 2 (S2):** Equipment Control messages, such as S2F41 (Remote Command Request) and S2F42 (Remote Command Acknowledge).
  - **Stream 7 (S7):** Process Program Management, including S7F1 (Process Program Send) and S7F5 (Process Program Request).

Event Notification

- **Event Reports:** Automatic notification of predefined events, such as process milestones, alarms, and state changes.

- **Custom Events:** User-defined events for specific equipment behavior.
- **Correlation with Streams:**
  - **Stream 6 (S6):** Event Reports and Data Collection messages, such as S6F11 (Event Report) and S6F12 (Event Report Acknowledge).

Data Collection

- **Trace Data Collection:** Periodic sampling of process parameters, sensor readings, and equipment status.
- **Collection Events:** Triggered data collection based on specific events or conditions.
- **Variable Access:** Real-time access to equipment variables, including both static configuration data and dynamic process data.
- **Correlation with Streams:**
  - **Stream 6 (S6):** Data Collection messages, including S6F1 (Data Collection Initialization Request) and S6F2 (Data Collection Acknowledge).
  - **Stream 9 (S9):** Data Acknowledgment messages, such as S9F1 (Data Message Not Allowed) and S9F9 (Unrecognized Device ID).

Alarm Management

- **Alarm Detection:** Real-time monitoring of equipment status to detect abnormal conditions.
- **Alarm Reporting:** Immediate notification to the host when alarms occur.
- **Alarm Acknowledgment:** Host acknowledgment to confirm receipt of alarm notifications.
- **Correlation with Streams:**
  - **Stream 5 (S5):** Alarm messages, including S5F1 (Alarm Report) and S5F2 (Alarm Acknowledge).

Equipment State Models

- **Control State Model:** Defines the equipment's control mode (e.g., Local, Remote, or Offline).
- **Processing State Model:** Represents the current operational state of the equipment (e.g., Idle, Processing, or Error).
- **Correlation with Streams:**
  - **Stream 1 (S1):** Equipment Status messages, such as S1F3 (Equipment Status Request) and S1F4 (Equipment Status Data).

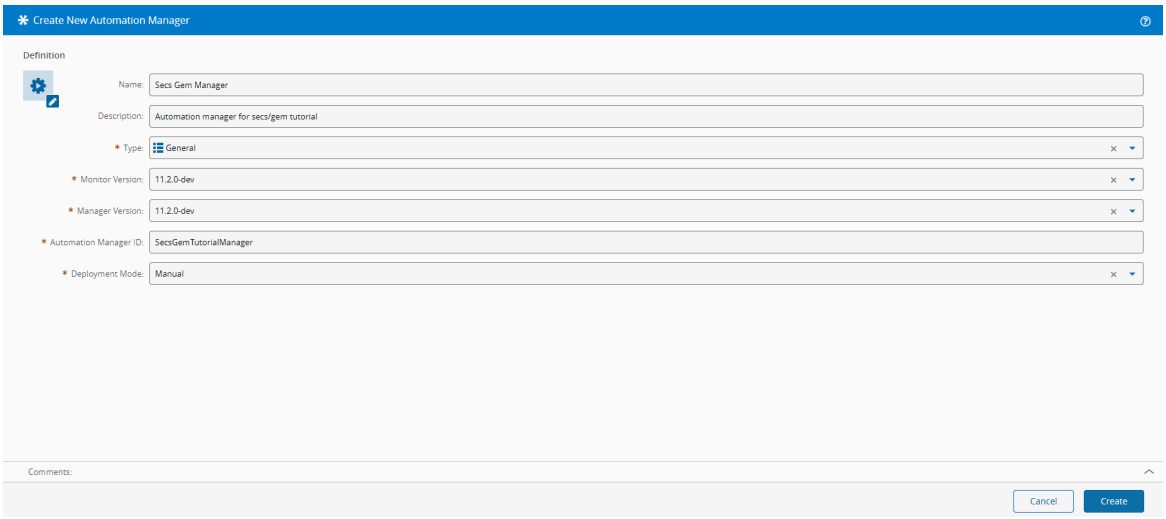
For `Connect IoT` an event is an `S6F11`.

Automation Manager

The `Automation Manager` entity represents a server process that will run and control a number of instances. This can be seen as an Operating System that will execute several processes. Each of the processes will be responsible to either connect to an interface of an equipment (network address, shared folder, etc) using a protocol driver, or execute the logic intended to model the behavior of the automation (what to do when the equipment raises an alarm, send a stop command, etc).

Go to `Automation > Automation Manager` and select the `New` button.

Enter the name, description, type and provide a unique name (in this example `SecsGemTutorialManager`) as the Automation Manager ID.



Select the `Create` button.

After creating an `Automation Manager`, download the automation manager. In order to do that:

Select the `Download` button.

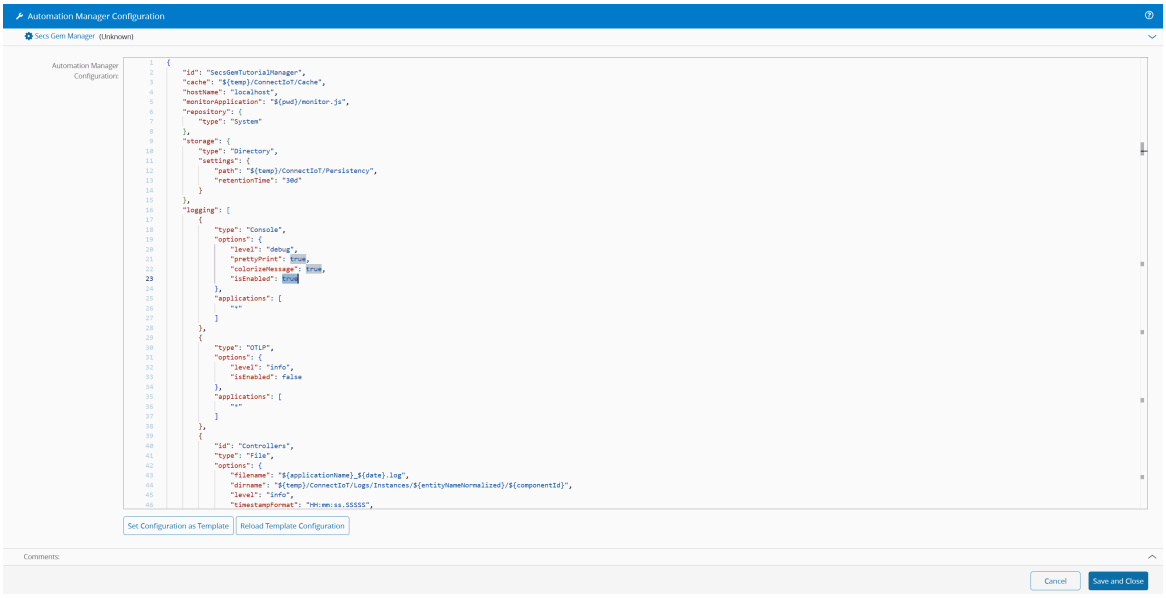
Select the `Authenticate With Current User` radio button.

> **Note**
>
> The user must be an integration user.

Select the `Download` button. This will download a `.zip` file.

Select the `Configuration` in the automation manager page.

We will be using the console logging transport, change the `isEnabled` to true.



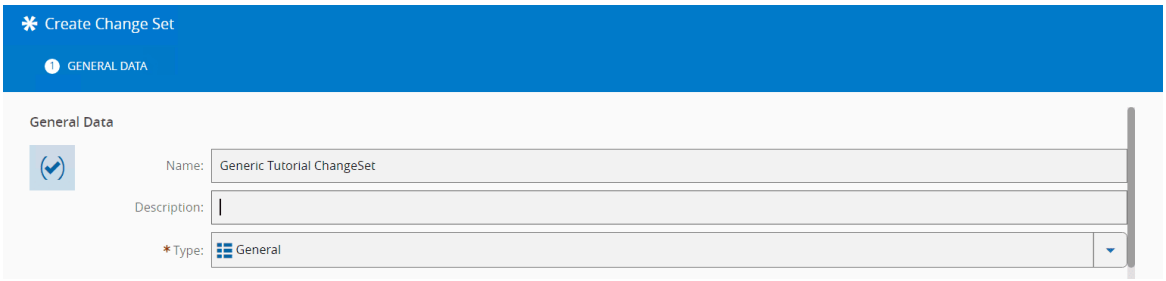
> **Note**
>
> By default the console transport is disabled and the verbosity for all transports is in the `info` level.

Automation Protocol

The Automation Protocol entity represents the technical specification of the protocol communication interface, implemented by the corresponding communication driver package. It holds the protocol parameters, the protocol data types and extended data meaningful to the protocol implementation. The default values for the protocol parameters are imported directly from the communication driver package, but they can be overridden as well.

Go to `Business Data > Automation Protocol` and select the `New` button.

Select the `Create` button to create a new ChangeSet or select an existing one. If creating a new one, enter the name, type and select the `Create` button.

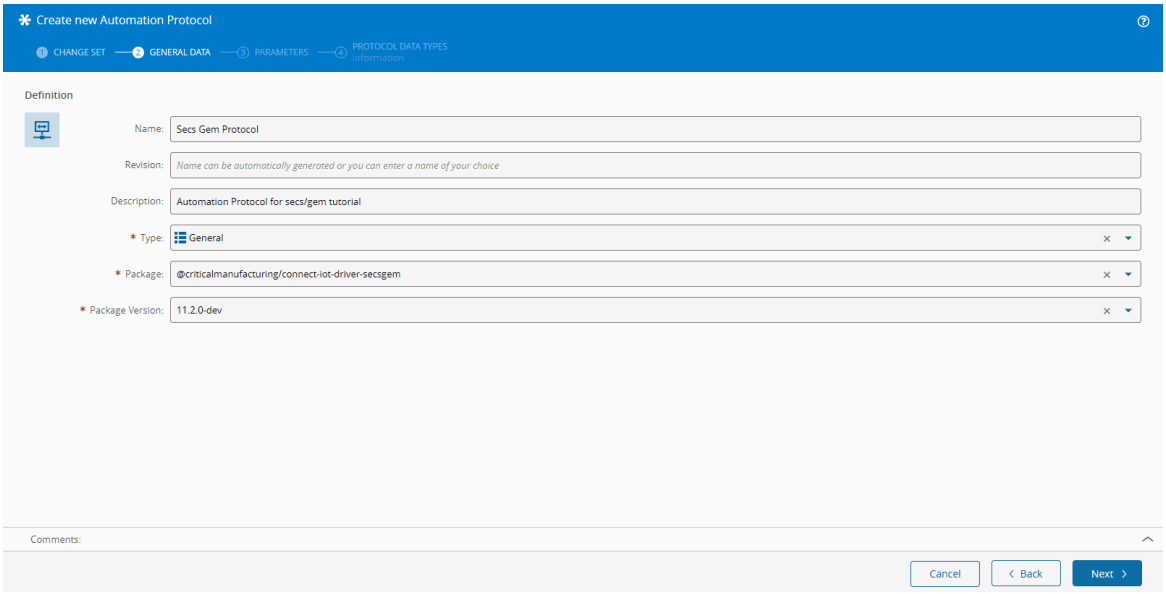


Select the `Next` button.

In the general data, provide:

- A name that represents the automation protocol;
- A description;
- The type (for classification purposes);
- `@criticalmanufacturing/connect-iot-driver-secsgem` as the package name;
- The package version.

> **Note**
>
> The package name and version selected will later indicate to the Automation Manager that is running, what process and version to download, prepare and execute.



Select the `Next` button.

Set the default values for the SECS/GEM protocol parameters.

> **Note**
>
> These values can be individually overridden by the Automation Controller workflows. In this step we will only define what will be the default values for all automations that use this protocol.

Select the `Next` button.

Select the `Create` button.

The Automation Protocol state becomes `Created`.

> **Note**
>
> The Automation Protocol entity is versioned.

Automation Driver Definition

The Automation Driver Definition entity represents the characteristics of a particular equipment type interface according to the vendor specification. A list of properties, events, the relation between them, commands and command parameters, must be chosen according to the requirements of the integration to implement.

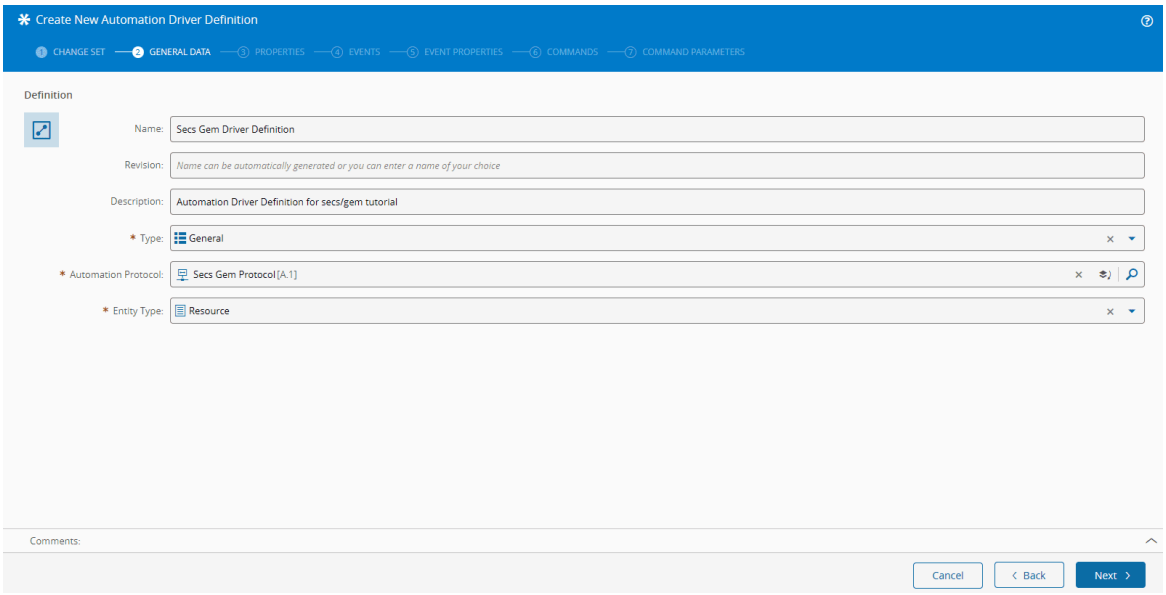
This tutorial, will require only one property (SVID in SECS/GEM) and one event (CEID in SECS/GEM) to detect the equipment changing its process status.

Go to `Automation > Automation Driver Definition` and select the `New` button.

Select or create an existing ChangeSet. Select the `Next` button.

In the general data, provide:

- A name that represents the automation driver
- A description
- The type (for classification purposes)
- The name of the previously created Automation Protocol - `Secs Gem Protocol.1`
- The `Resource` type as the entity type for the new Automation Driver Definition.



Select the `Next` button.

Let's assume that according to the specification, the VID representing the equipment status is 2011, of data type U1 (Format) and it is a Status variable (SV).

VID	VID Type	Format	Name	Description
2011	SV	U1	Process status	Indicates the current process status of the equipment

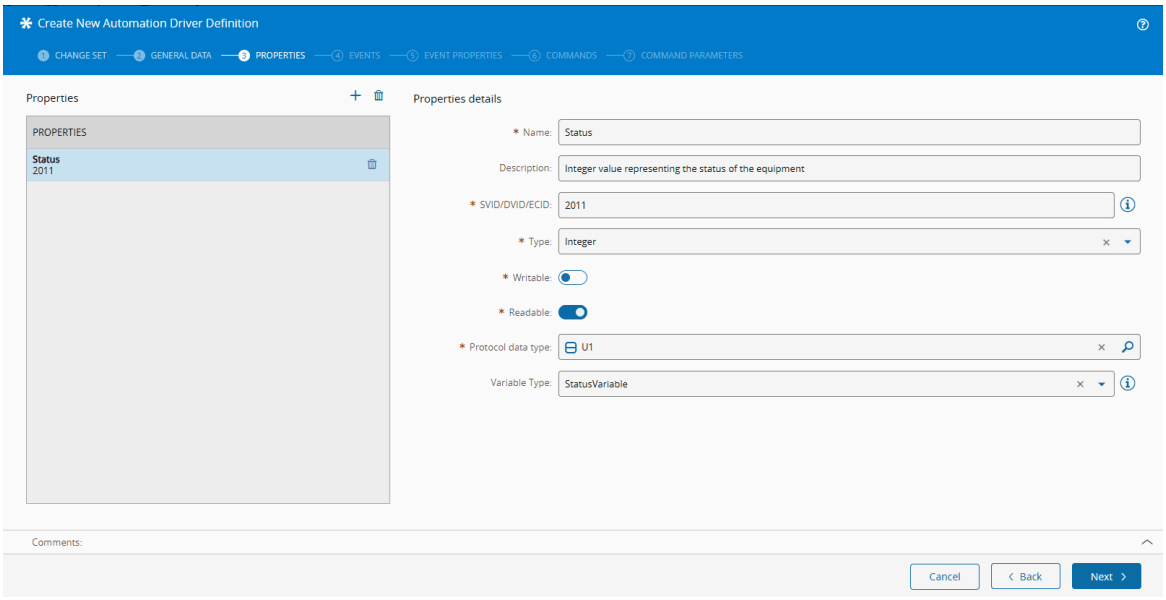
Table: VID Property

> **Note**
>
> Each equipment will have it's different interface and mapping of variables and reports. It's important to validate the interface with the specification and with the real machine behavior.

Add a new entry to the list of Properties by selecting the `+` button.

In the property details, provide:

- A name that represents the variable name;
- A description;
- The device id which is the SVID in SECS/GEM;
- The type of data of the variable;
- The writable and readable flags (in this case, we cannot change the value);
- The data type of the variable in SECS/GEM format;
- Finally, the SECS/GEM variable type (Status variable, data variable or constant) ;



Select the `Next` button.

Let's assume that according to the specification, the CEID representing the equipment changing status event is 1000.

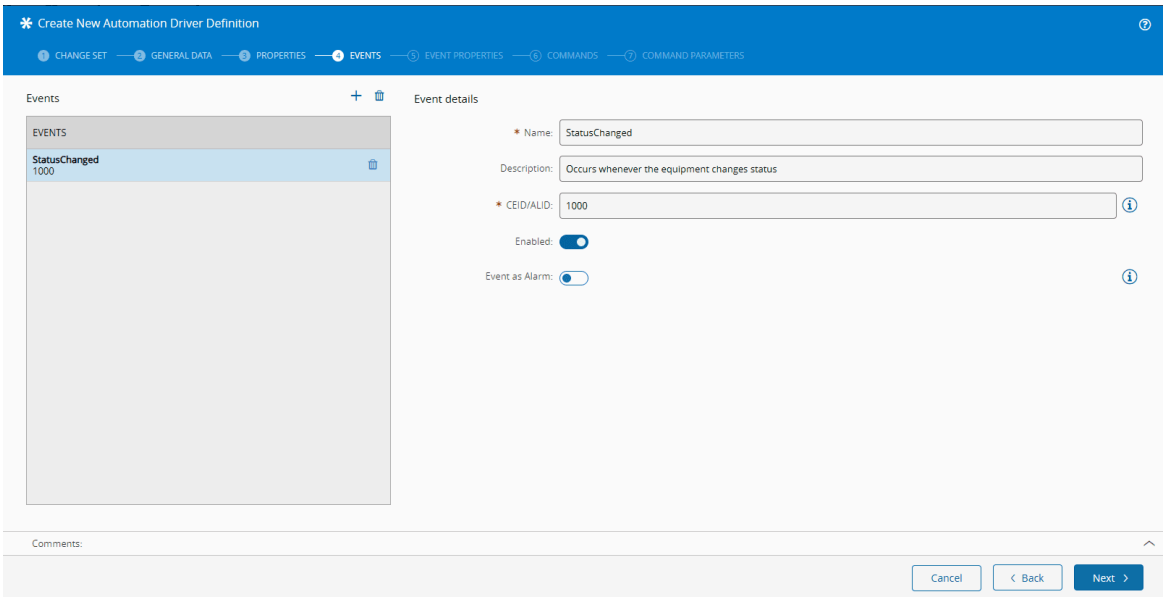
CEID	Description/Trigger	RPTID
1000	Occurs when the equipment process status changes	2

Table: CEID Property

Add a new entry to the list of Events by selecting the `+` button.

In the event details, provide:

- A name that represents the event name;
- A description;
- The device id which is the CEID in SECS/GEM;
- The enabled flag (in this case, turn it on. If off, the event will be ignored);
- The event as alarm flag (identify this event as an alarm in SECS/GEM. In this case, turn it off);



Select the `Next` button.

In the following step, we combine the information between events and properties. The list of configured events is available on the left side. For each event, add the associated properties on the right side according to the vendor specification. If a specific report id is also provided by the specification (RPTID in SECS/GEM) , you should set it as well.

Note

If the report id is left blank, it will be auto generated.

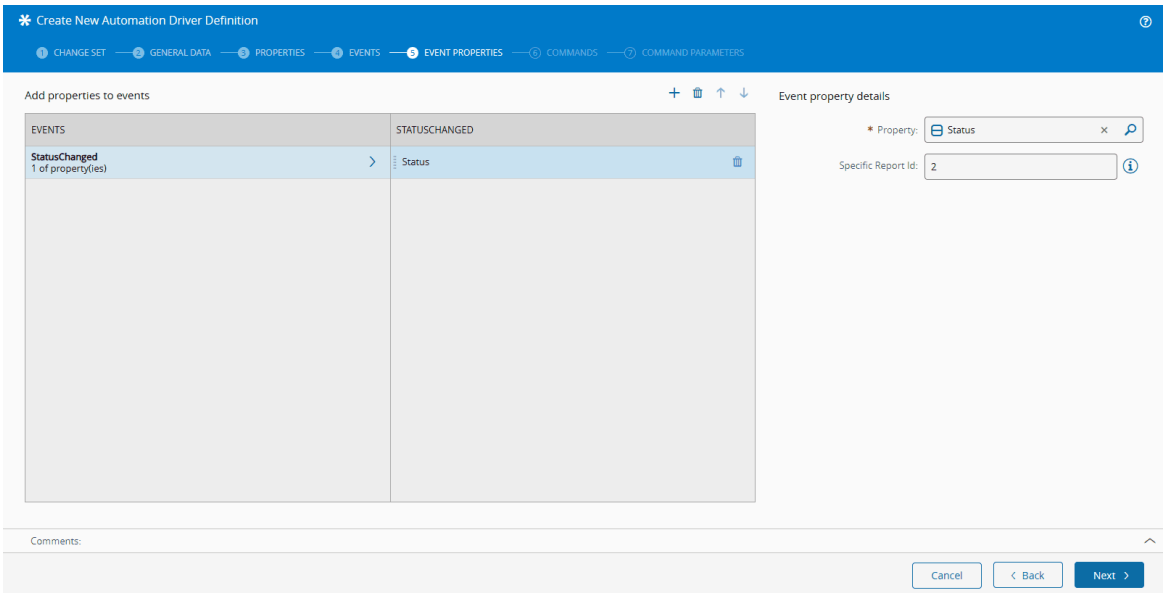
Let's assume that according to the specification, the RPTID associated to the event StatusChanged is 2.

CEID	Description/Trigger	RPTID
1000	Occurs when the equipment process status changes	2

Table: Status changed event

Note

For each event, make sure to add the properties and the reports in the exact order mentioned by the vendor specification. Otherwise it will compromise the mapping between the data sent by the equipment and the expected data configured by the driver definition, which in turn might lead to an unexpected behavior by the automation controller.



Select the `Next` button.

This tutorial does not focus on the equipment commands, so skip the Commands panel by selecting the `Next` button.

Skip the Command Parameters panel as well by selecting the `Create` button.

The Automation Driver Definition state becomes `Created`.

> **Note**
>
> The Automation Driver Definition entity is versioned.

Automation Controller

The Automation Controller will orchestrate the behavior between the equipment and the MES system. In this tutorial it will detect the change of the equipment status property and map it to a `SEMI E10` Resource State, according to the following table:

Equipment Status	MES Resource SEMI E10 State
1	Nonscheduled
2	Unscheduled Down
3	Scheduled Down
4	Engineering
5	Standby
6	Productive

Table: Mapping SEMI E10 Resource State to a property

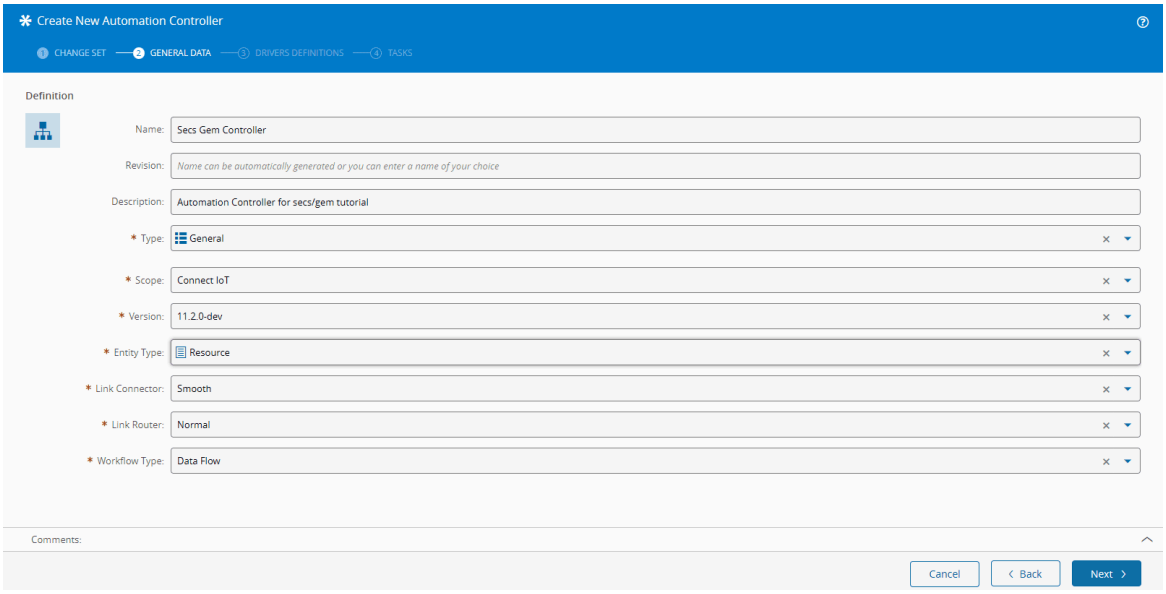
Go to `Automation > Automation Controller` and select the `New` button.

Select or create an existing ChangeSet. Select the `Next` button.

Provide a name, description, type, controller package version and set **Resource** as the entity type for the new Automation Driver Controller.

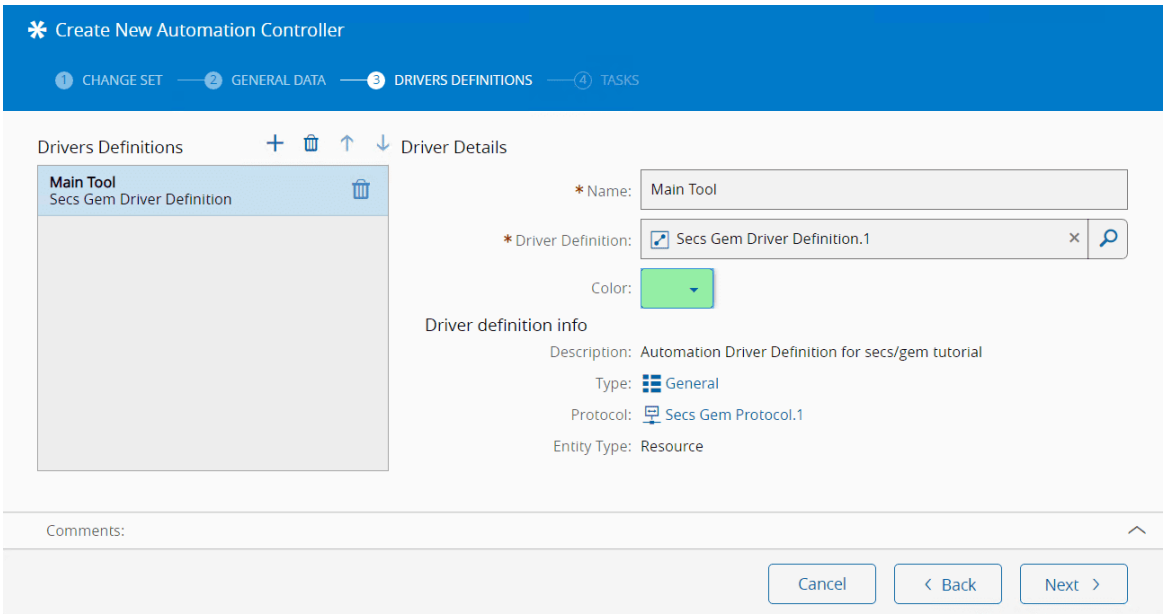
For this tutorial we will use `Data Flow`, Connect IoT also supports `Control Flow` as a different engine of ingestion of events and commands. You can also customize how your workflow behaves by configuring the link router and connector.

Select the `Next` button.



In the drivers definitions panel, add a new driver definition by selecting the `+` button. Provide a friendly name for it, select the **previously created Automation Driver Definition** and choose a color to paint the tasks that will be associated to this driver definition.

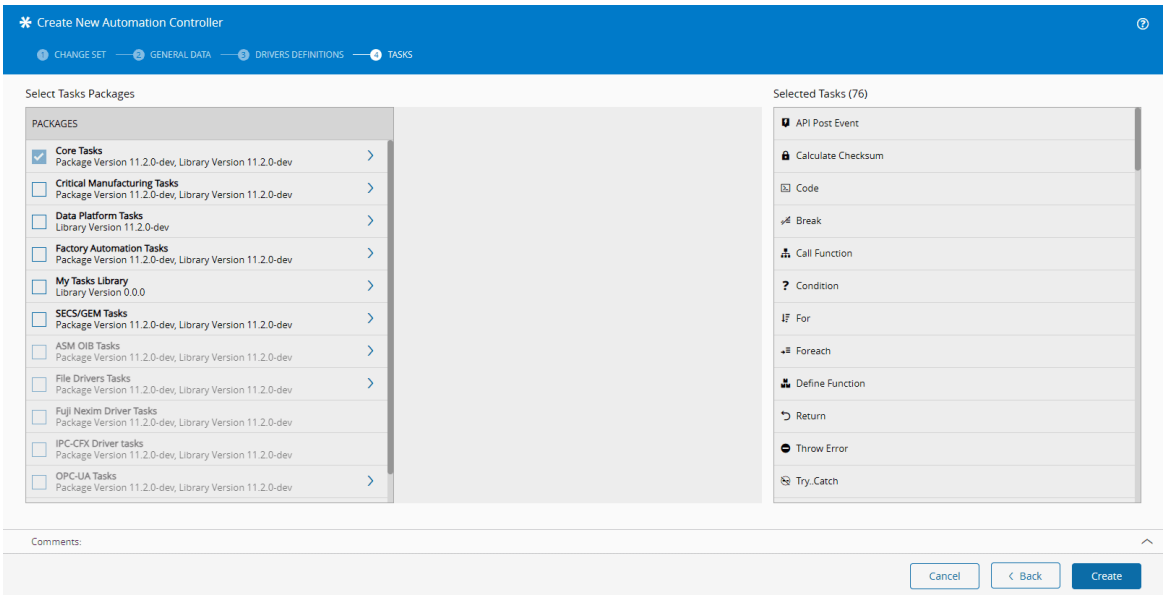
Select the `Next` button.



In the tasks panel, select the task packages that will be relevant for the integration itself and available in the workflow designer. For this tutorial, the Core Package - `connect-iot-controller-engine-core-tasks` - is enough.

Note

This package contains driver related tasks such as equipment setup, event triggering, property handling, command handling, etc. Also contains system related tasks such as entity instance, execute a service or action, adjust entity state, etc. And finally logic/flow related tasks, such as persistency handling, switch statement, timers, logging messages, etc.



Note

The tasks loaded by the Automation Controller are set in metadata through a `ControllerEngineFilter` structure, making the packages and tasks visible and/or selected and mandatory according to the loaded information. These filters can indicate mandatory and or dependency status and there are several rules that dictate the proper functioning:

- If a package matches the mandatory filters, it will be selected in the package list and cannot be unselected. All others will be unselected;
- If a tasks package is listed with a dependency and doesn't match the filter, it should appear as disabled in the list of packages available for selection. Specific tasks follow the same rules, meaning that the system can have a configured package with some tasks that only work along with specific drivers and scopes;
- The GUI sorts the available package in the list by the following order:
  - Mandatory (and selected)
  - Available for selection
  - Disabled

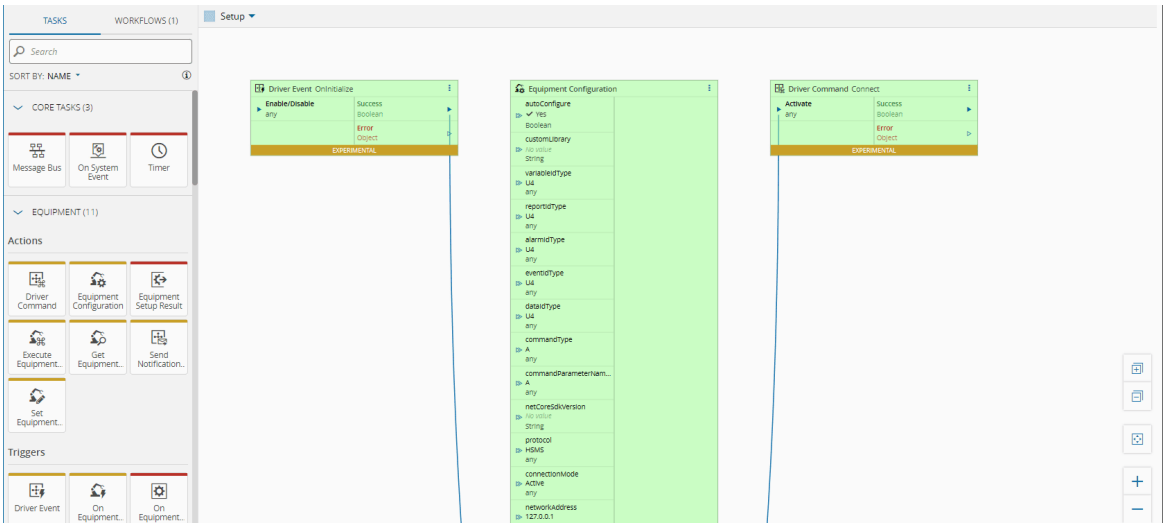
Select the `Create` button.

The Automation Controller state becomes `Created`.

> **Note**
>
> The Automation Controller entity is versioned.

Automation Controller Workflow

The workflow designer is where the user can define and orchestrate the integration business logic. The workflow template already comes ready to connect with a `Setup` page.



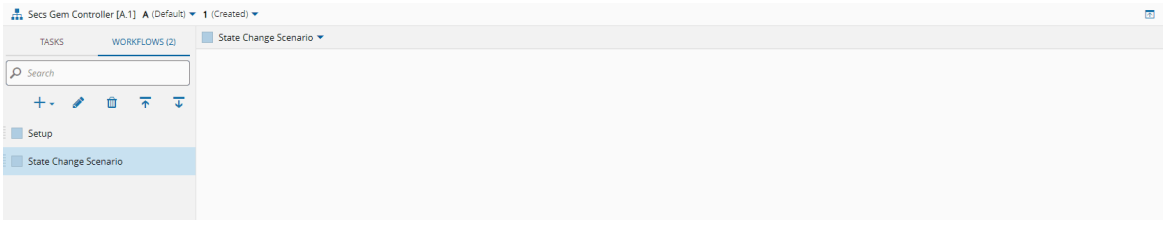
The `Setup` workflow page has three tasks within it:

- Driver Event - OnInitialize
- Equipment Configuration
- Driver Command - Connect

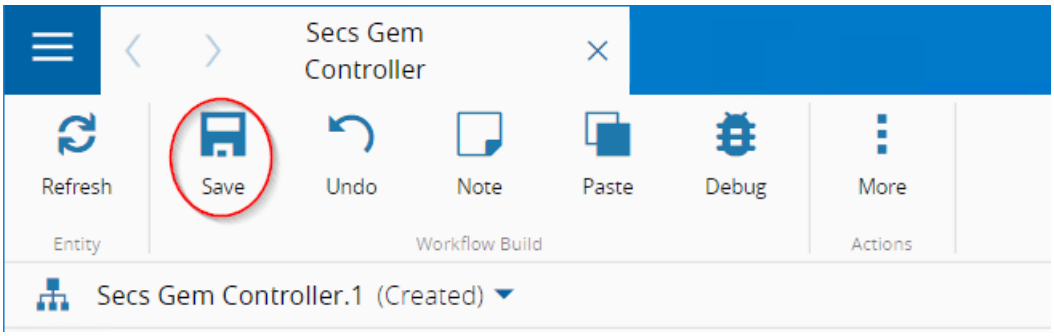
The Driver Event OnInitialize will signal the start of the driver lifecycle, then the equipment configuration will expect all the definitions related with the driver and when it's ready it will send a Driver Command to Connect.

State Change Scenario

Create a new Page and give it a meaningful name. For example, `State Change Scenario`.

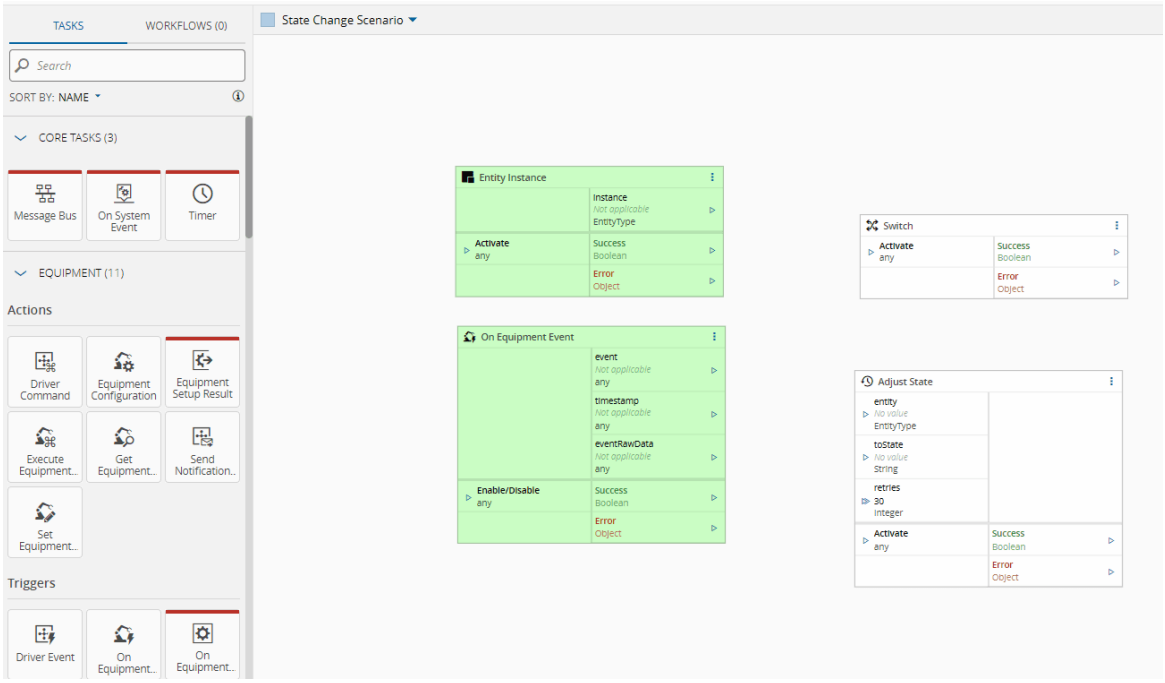


You should save the changes in order to not lose them. Just select the `Save` button at the top ribbon. You may save the changes whenever needed from now on, after performing any change within the workflow.

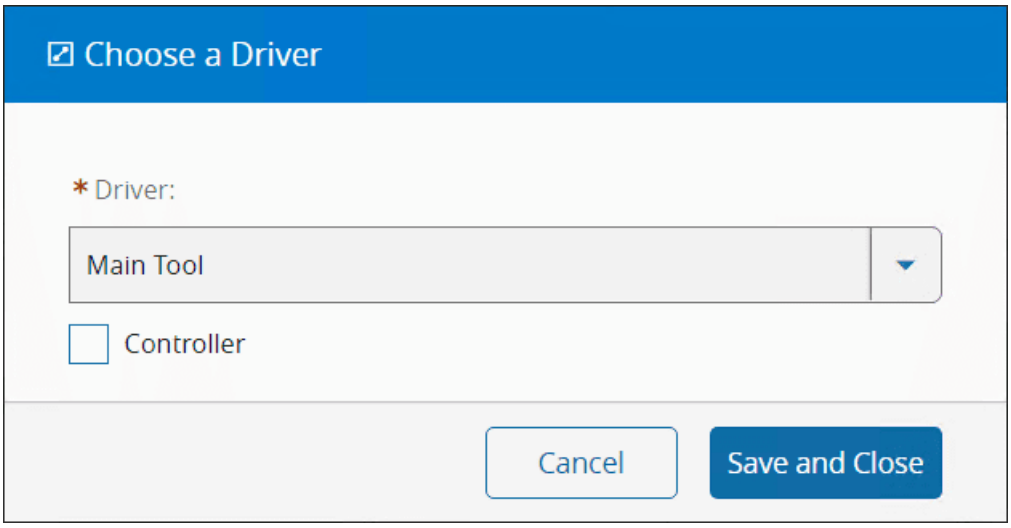


Drag and drop the following tasks to the State Change Scenario page designer:

- Entity Instance (will represent the MES Resource entity in runtime)
- On Equipment Event (Will be triggered when the state changes in the equipment/simulator)
- Switch (Will decide the final Semi E-10 state name)
- Adjust State (Will change the state of the resource in MES)



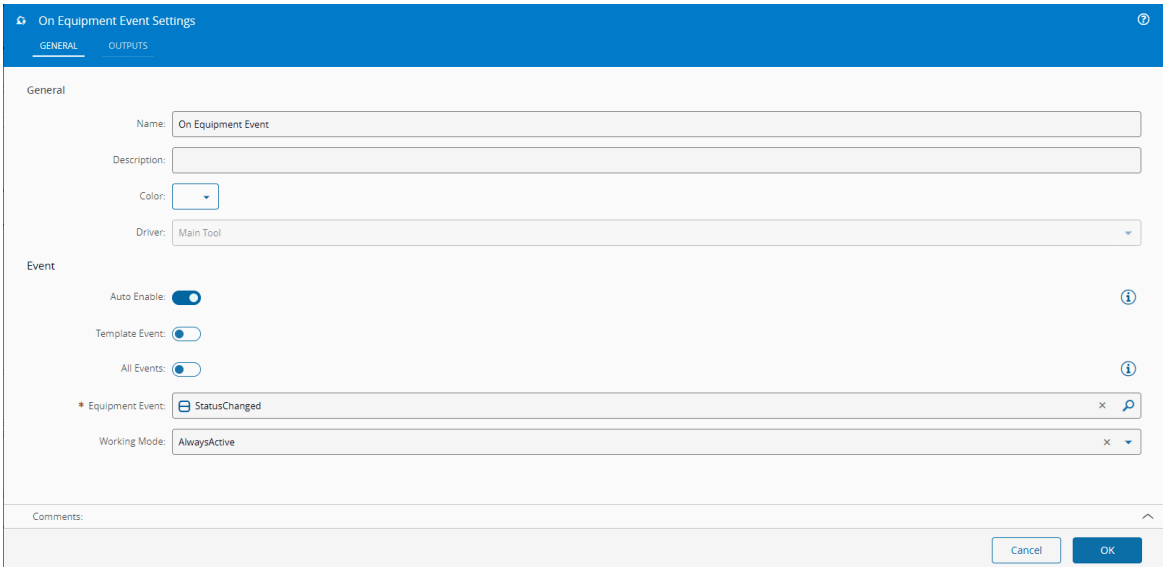
When adding the Entity Instance task, it will be asked for you to choose between the entity associated to one of the drivers or the entity associated to the controller. Choose the `Main Tool` driver and select the button `Save and Close`.



Note

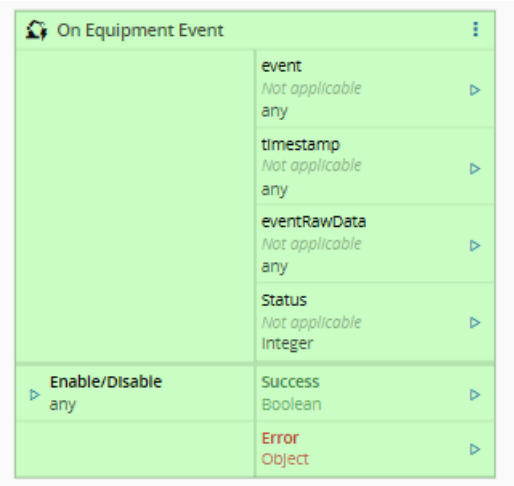
There is only three outputs for the *On Equipment Event* task. These are the default ones. As soon as a specific event is configured, the associated properties, if any, will be displayed as outputs as well.

Go to the `Settings` of the `On Equipment Event` task and select the event `StatusChanged`. Select the `OK` button.



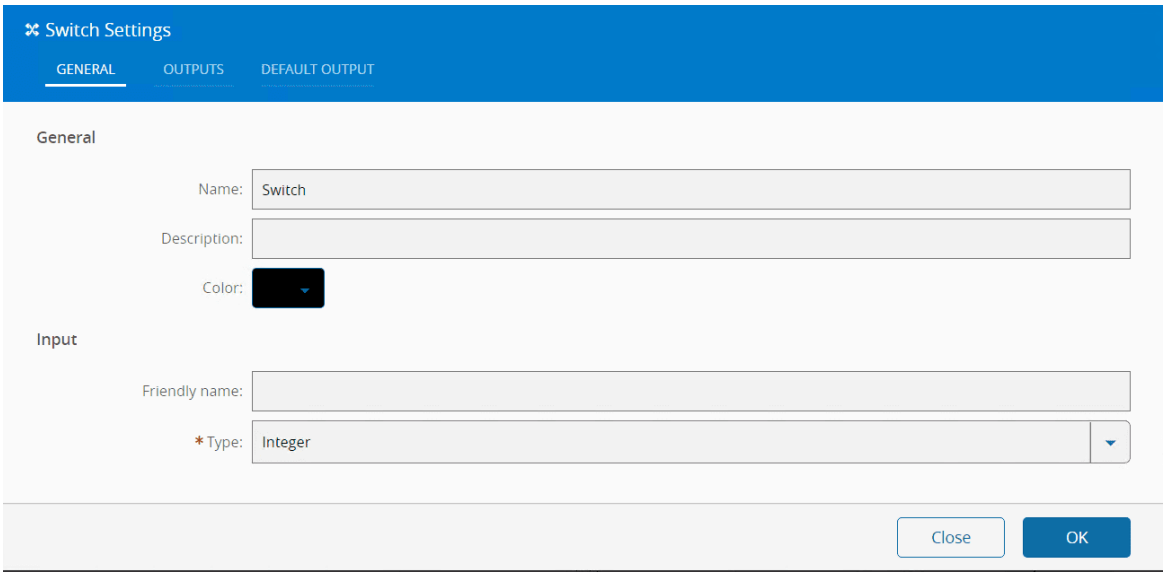
Note

Notice that there is now one new output available - `Status`, as defined previously at the Automation Driver Definition for the `StatusChanged` event.



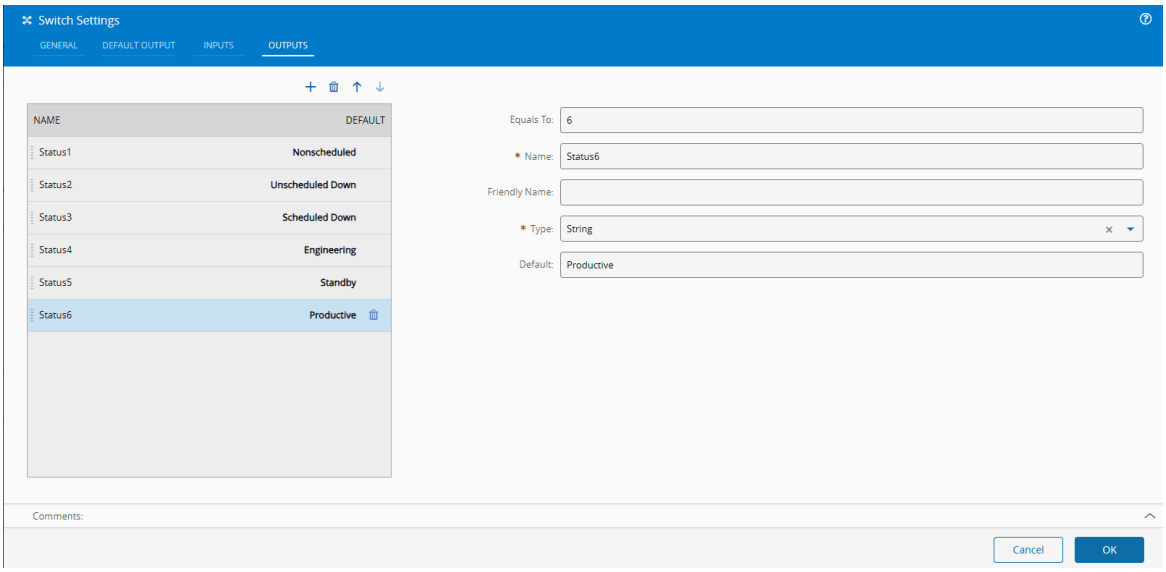
Go to the `Settings` of the `Switch` task and set the Input Type to `Integer`.

Select the `OK` button.



In the Outputs panel, add 6 output entries. For each one, configure the Name, Equals to, Type and Value, as shown below. All output types are `String` and the `Equals to` goes from 1 to 6.

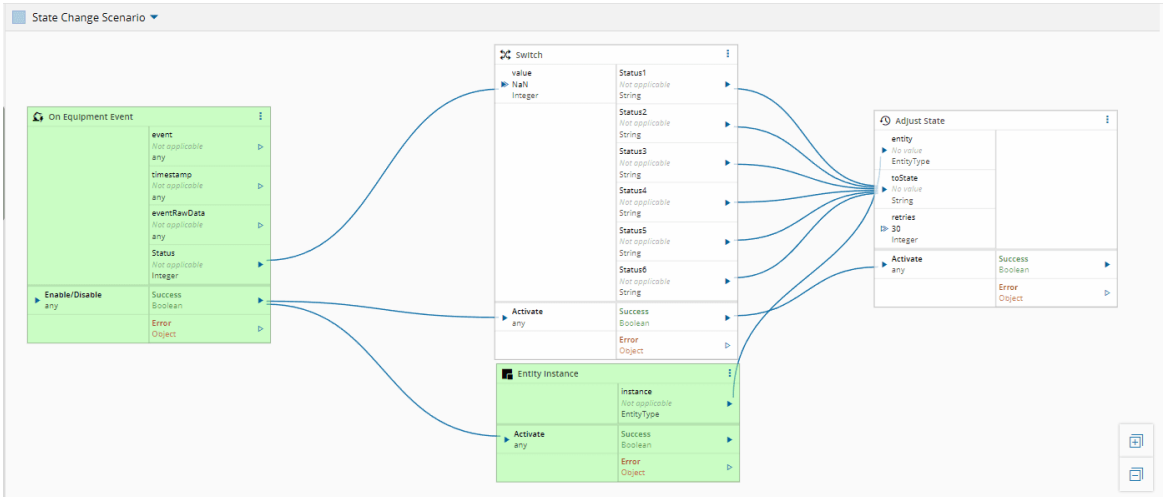
Select the `OK` button.



Now we can link the tasks together, in order to produce the desired result of detecting the change of the equipment status and map it to a SEMI E10 Resource State in the MES for the resource entity that will be associated to this controller:

- Link the `Status` output of the `On Equipment Event` task instance to the `value` input of the `Switch` task instance;
- Link the `Instance` output of the `Entity Instance` task instance to the `entity` input of the `Adjust State` task instance;
- Link each one of the outputs of the `Switch` task instance to both `toState` and `Activate` inputs of the `Adjust State` task instance.

It should look like this.



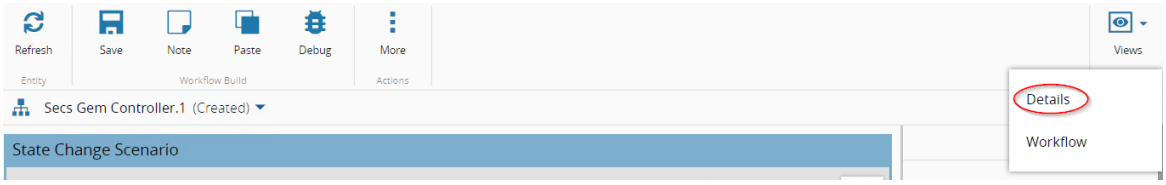
So, basically what will happen behind the scenes, is that whenever the driver process detects the trigger of an event defined in the Automation Driver Definition, it will forward it to the controller process. The controller process by its turn will check if the event configured in the `On Equipment Event` matches the event sent by the driver. If so, the `Switch` task instance gets activated, and compares the `Status` value with the switch case. When a match is found it activates the `Adjust State` task instance. The `Adjust State` task instance will update the SEMI E10 state for the given Resource Entity Instance in MES.

> **Note**
>
> The Resource Entity Instance will be assigned in the next step when creating the Automation Controller Instance.

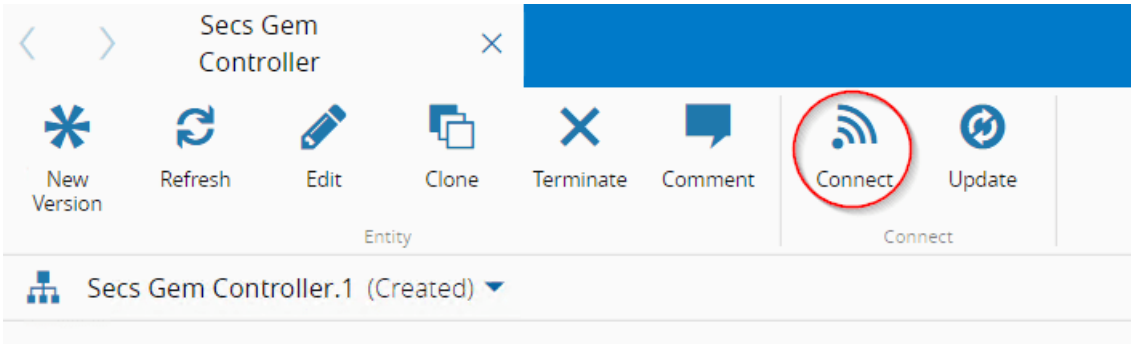
Automation Controller Instance

Now that we defined the intended behavior through the Automation Controller, we need to associate it to an entity and a manager.

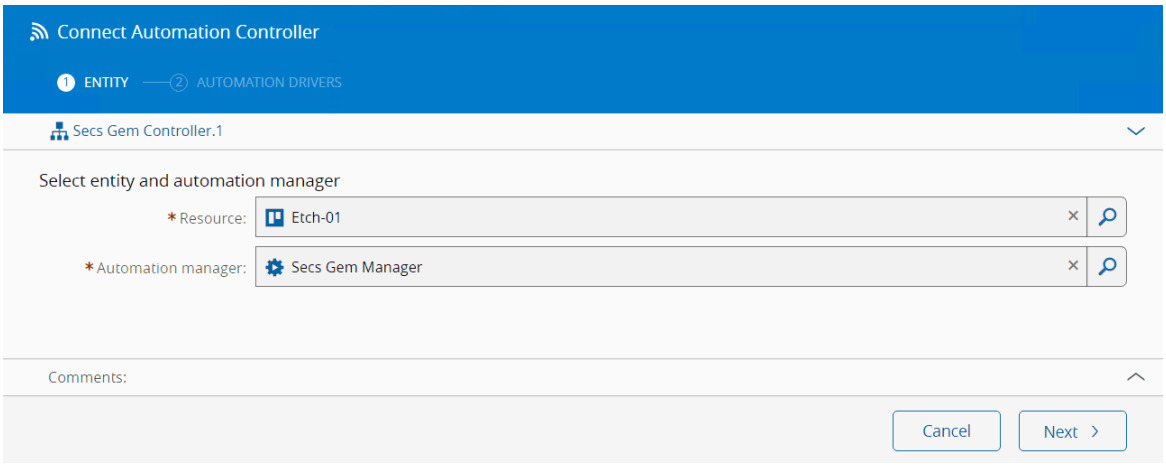
Go back to the workflow details. Go to `Views > Details`.



Select the `Connect` button at the top ribbon.

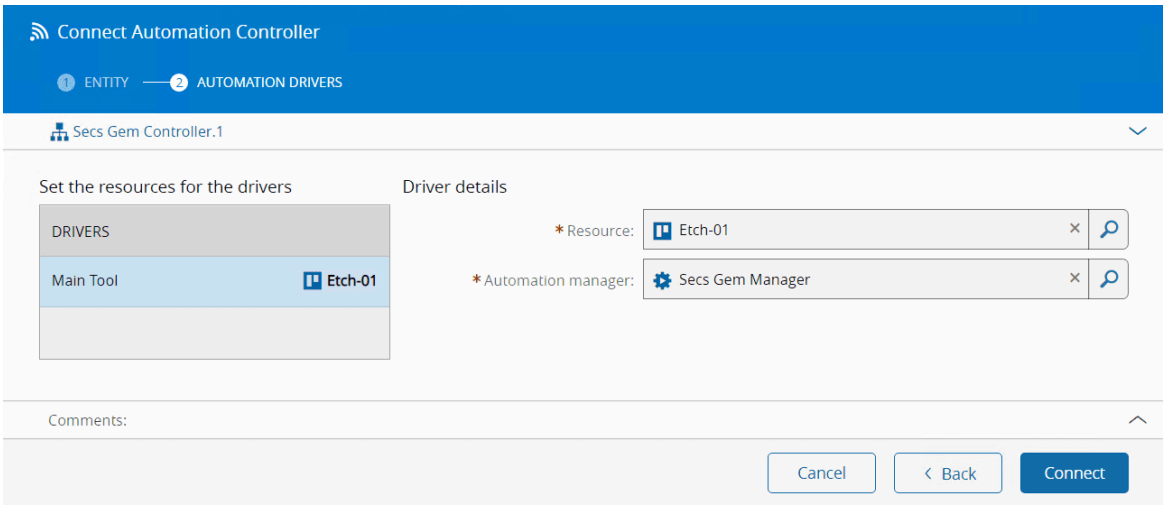


You must provide the resource and the automation manager this instance will be associated to. Choose the resource and select the automation manager previously created in this tutorial.



Select the `Next` button.

In the Automation Drivers panel select the `Main Tool` driver and choose the same resource chosen before for the controller (Etch-01 in this example).

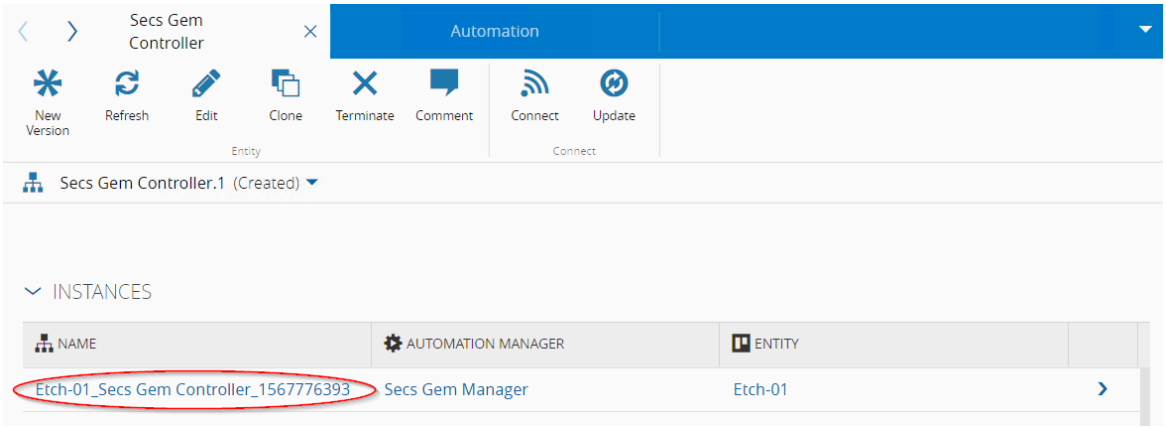


Note

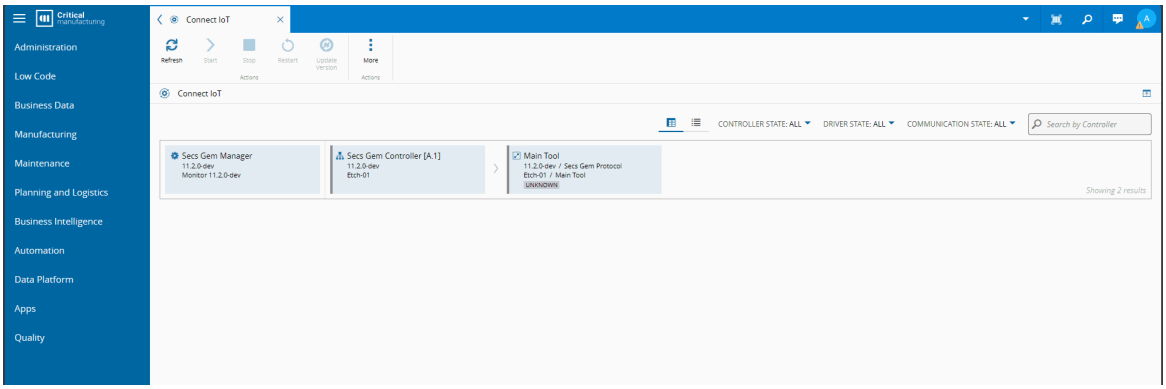
The entity that gets selected will provide context to either the controller or driver definitions. The controller and each driver definition might have distinct entities associated to it. For example, let's consider a line with 3 equipment. The controller entity might be associated to the resource that represents the line, and each individual driver definition the corresponding equipment resource.

Select the `Connect` button.

Now, if you scroll down you can find a summary of the instances for this controller. In this case, you will see the one that was just created.



You can also check for the controller instance and the associated driver instance if you go to the `Automation > Connect IoT` menu item. Here are listed all available instances in the system, grouped by the automation manager.



Note

This UI will show the last known state of the automation and will then query all managers for their state. Whenever you see a spinner on a card it means the current state has not yet been resolved.

Now unzip the manager you have downloaded and start it. In order to start it you can got to the folder scripts and run the `StartConsole.bat`.

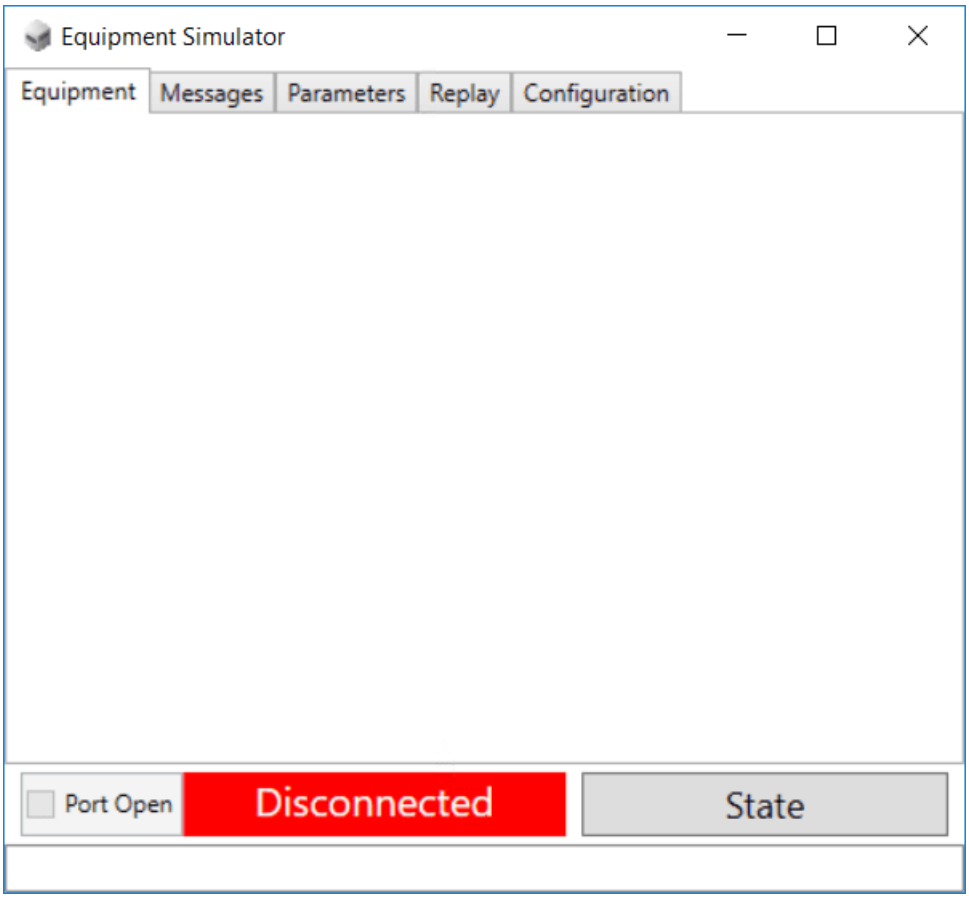
Now check the Automation Manager console. As the Automation Controller information was updated the process now gets again the list of instances to start and monitor. The Automation Monitor then asks the Automation Manager to start the processes for the Automation Controller instance and the Automation Driver instance.

```
2019-09-06 14:26:33.461 info: Controller information for this AutomationManager was updated! Recalculating processes inf
ormation
2019-09-06 14:26:33.462 info: Getting list of instances (controller and driver) to monitor ('SecsGemTutorialManager')
2019-09-06 14:26:33.839 info: Preparing package '@criticalmanufacturing/connect-iot-driver-secsgem', version '7.0.0-6525
3'
2019-09-06 14:26:33.841 info: Resolving package '@criticalmanufacturing/connect-iot-driver-secsgem' with version '7.0.0-
65253' location.
2019-09-06 14:26:33.847 info: Preparing package '@criticalmanufacturing/connect-iot-controller', version '7.0.0-65253'
2019-09-06 14:26:33.855 info: Monitor is requesting list of running processes
2019-09-06 14:26:33.847 info: Resolving package '@criticalmanufacturing/connect-iot-controller' with version '7.0.0-6525
3' location.
2019-09-06 14:26:33.857 info: Processes to start:
2019-09-06 14:26:33.863 info: Monitor is requesting to start a process...
2019-09-06 14:26:33.857 info: AutomationControllerInstance/1909061355490000001  @criticalmanufacturing/connect-iot-contr
oller@7.0.0-65253
2019-09-06 14:26:33.863 debug:  Process Data: type='connect.iot.process.start', content={object: id='AutomationDriverIns
tance/1909061355490000001', path='C:\IoT\cache\connect-iot-driver-secsgem@7.0.0-65253\src\index.js', arguments='--id=Aut
omationDriverInstance/1909061355490000001,--componentId=Main Tool,--serverHost=localhost,--monitorPort=64470,--monitorHo
st=localhost,--monitorToken=monitorSecurityToken,--monitorSsl=false,--config=c:\IoT\Configs\SecsGemTutorial.json,--dev=f
alse,--entityName=Etch-01'},
2019-09-06 14:26:33.857 info:     AutomationDriverInstance/1909061355490000001  @criticalmanufacturing/connect-iot-drive
r-secsgem@7.0.0-65253 (Main Tool)
2019-09-06 14:26:33.876 info: Process started with PID 20796.
2019-09-06 14:26:33.859 info: Starting process id='AutomationDriverInstance/1909061355490000001', name='@criticalmanufac
turing/connect-iot-driver-secsgem', version='7.0.0-65253'
2019-09-06 14:26:33.861 info: Starting 'Main Tool' main script 'C:\IoT\cache\connect-iot-driver-secsgem@7.0.0-65253\src\
index.js'
2019-09-06 14:26:33.879 info: Monitor is requesting to start a process...
2019-09-06 14:26:33.861 info: Requesting fork Id='AutomationDriverInstance/1909061355490000001', Path='C:\IoT\cache\conn
ect-iot-driver-secsgem@7.0.0-65253\src\index.js'
```

Equipment Simulator Tests

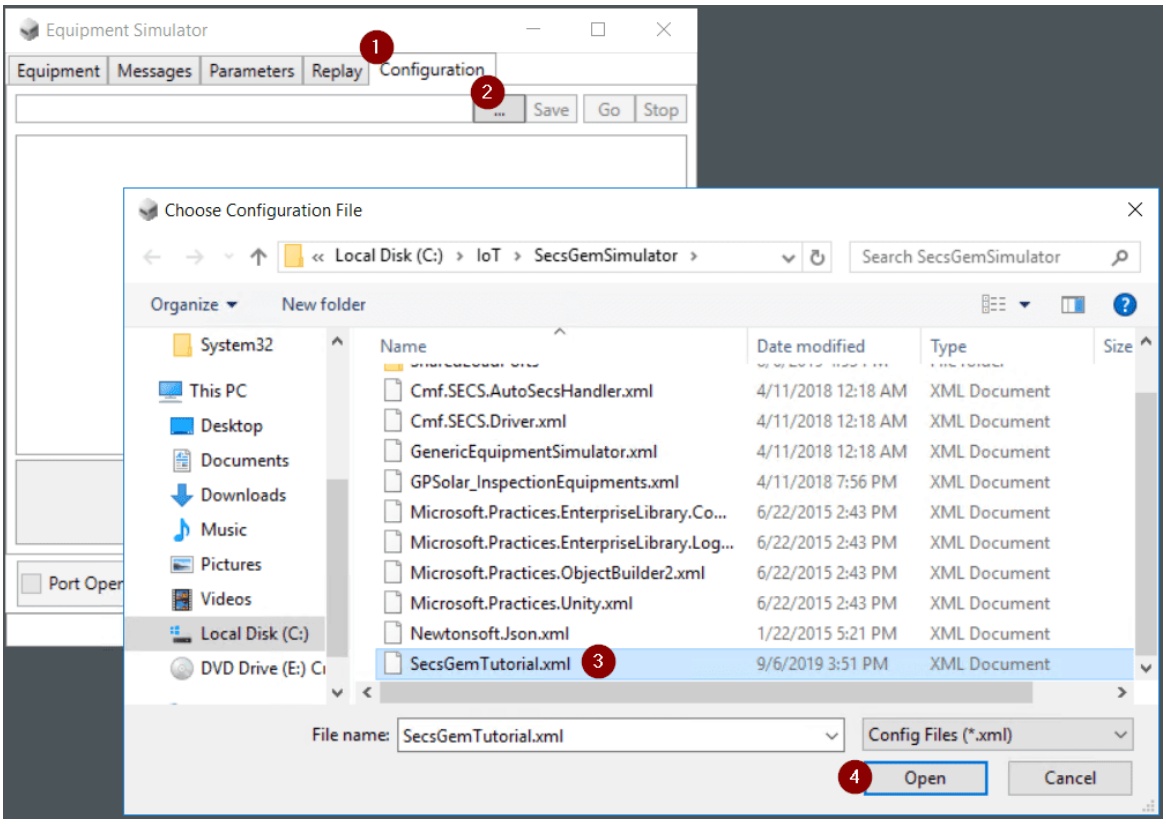
In order to test this integration tutorial, let's open the Secs/Gem Equipment Simulator Tool (you can download it from this location.

Run the GenericEquipmentSimulator.exe file. The following window will appear.

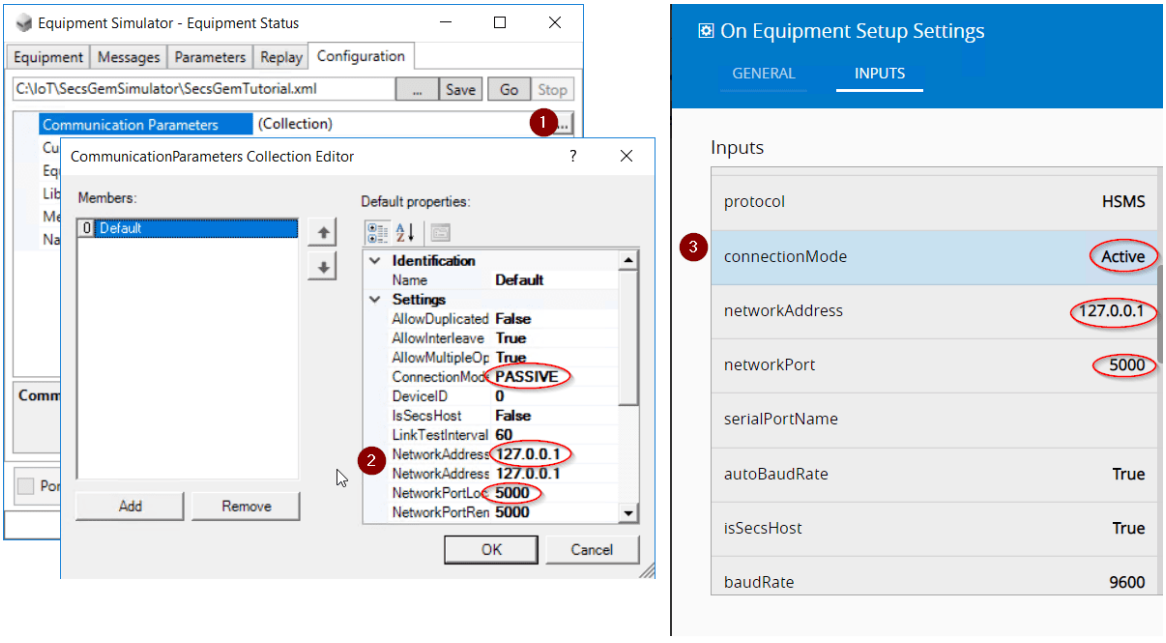


Notice that the simulator communication state is set to `Disconnected`.

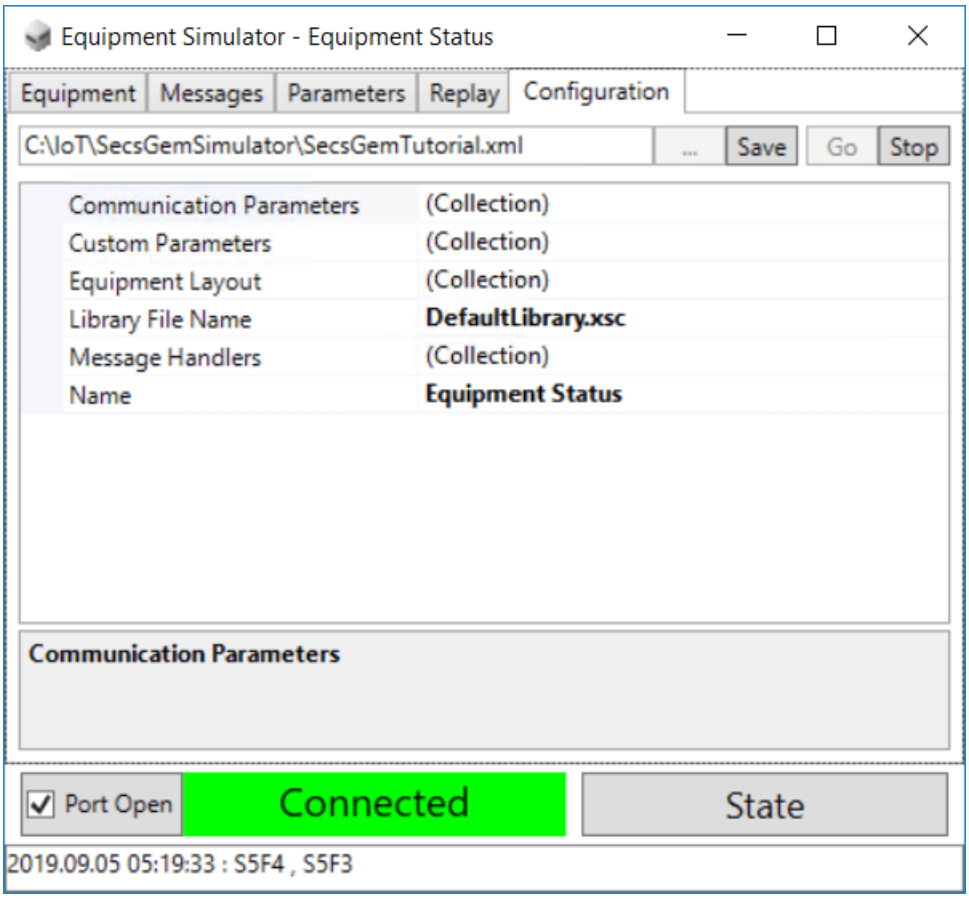
Go to the `Configuration` tab (1). Select the `...` button (2) to open the simulation configuration file. Select the `SecsGemTutorial.xml` file and select the `Open` button.



You must check the `Communication Parameters` (1) and validate if the communication settings (2) are compatible with the default settings (3) of the `On Equipment Setup` task instance configured in the workflow.

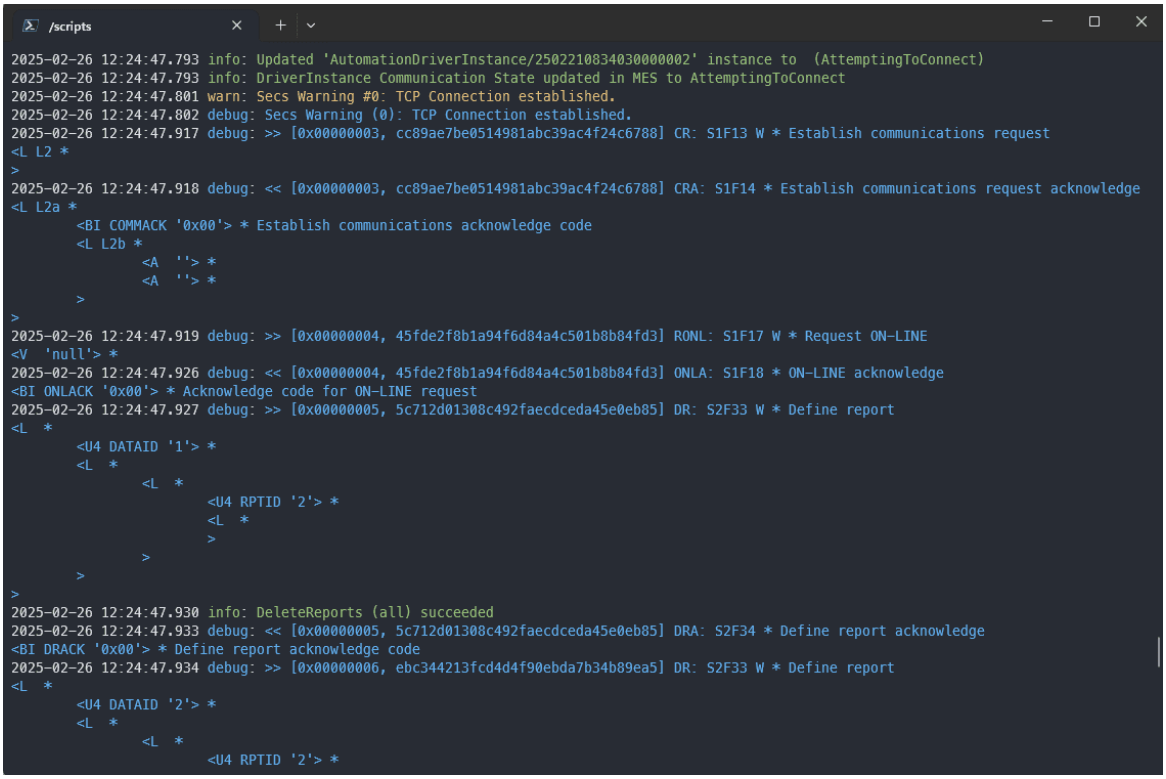


Once the parameters are validated, you can turn the simulator on by selecting the `Go` button.

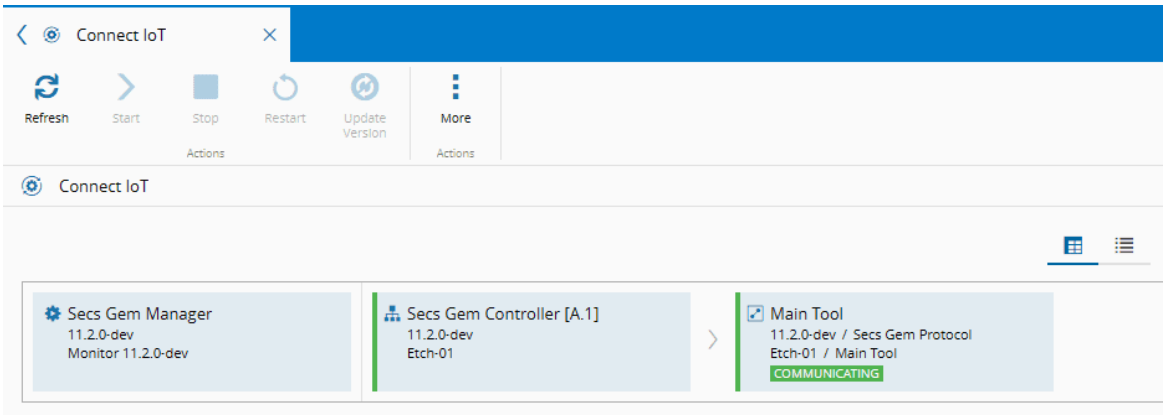


Notice that the simulator communication state changes from `Disconnected` to `Connected`.

It is also interesting to see all the lifecycle messages for secs/gem happening. We can see that there was an establish communication adn a go online, followed by a set of commands to define the reports.



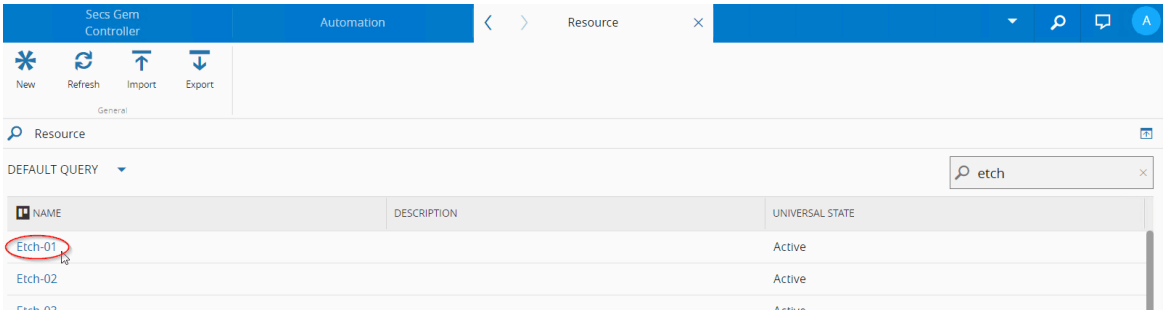
Also notice that the communication state in the `Automation` tab in the MES system changes to `Communicating` for the driver instance.



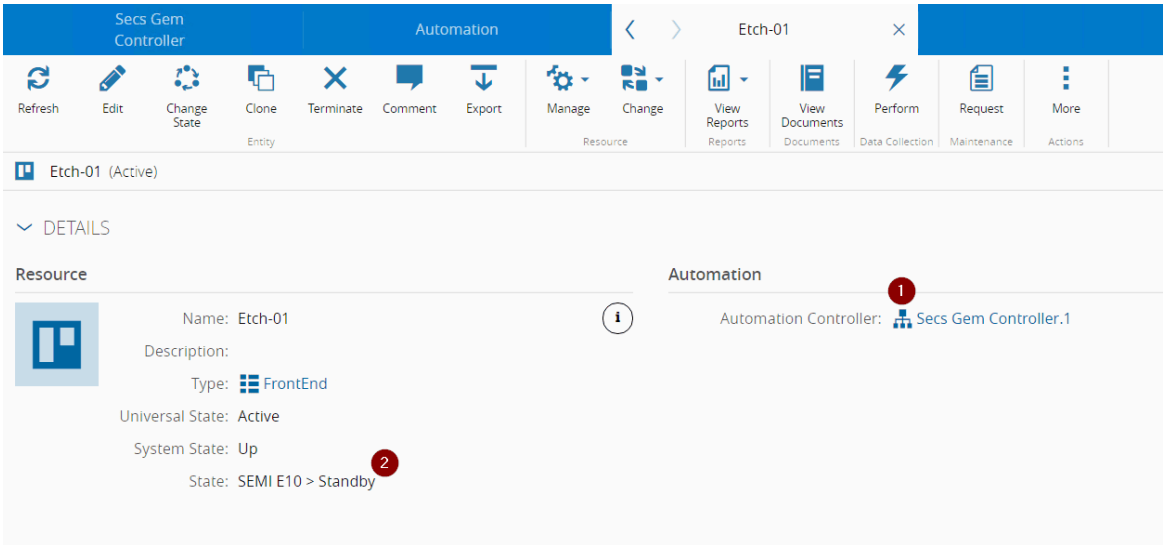
Note

If you turn the simulator off by selecting the 'Stop' button, the communication states will change back to `Disconnected`. You can test by yourself by alternating between the `Go` and `Stop` button, but then leave it communicating for being able to follow the next steps of the tutorial.

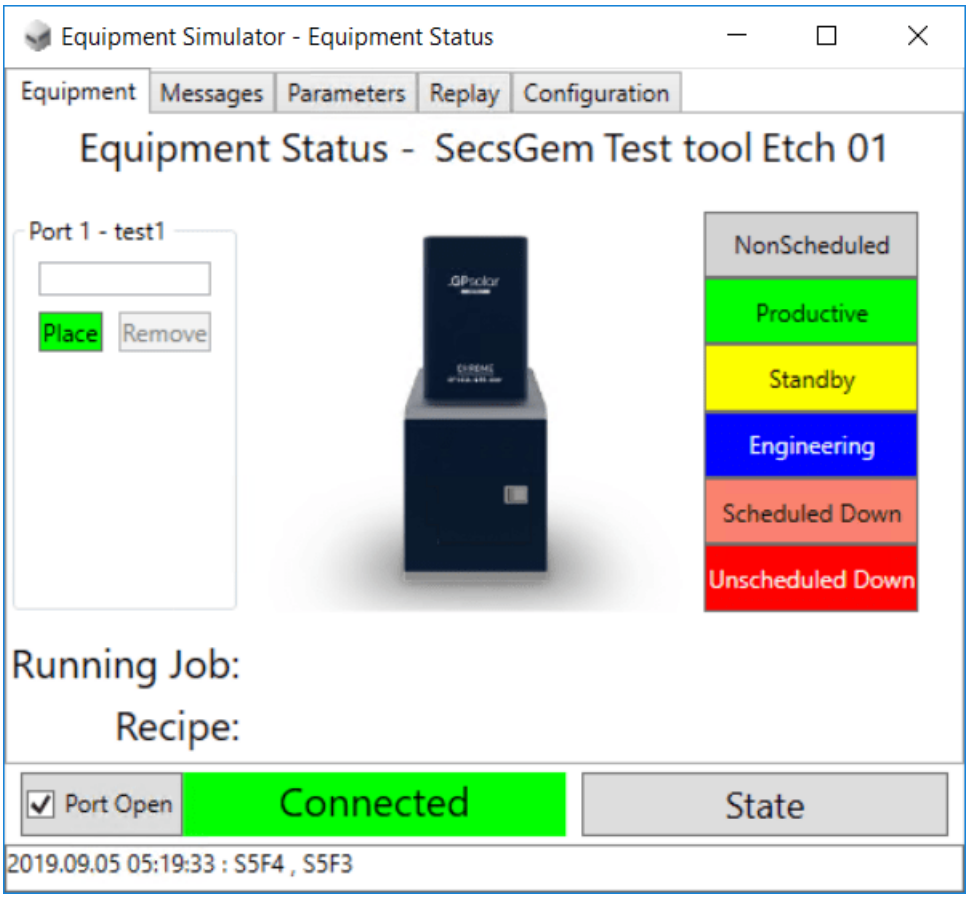
Go to `Business Data > Resource` and search for the resource. Select over the resource `Etch-01` to open its details.



You can see that the resource is associated to the `Secs Gem Controller.1` Automation Controller (1) and that the current SEMI E10 state is set to `Standby` (2).



Let's switch to the `Equipment` tab in the Secs/Gem Equipment Simulator Tool.



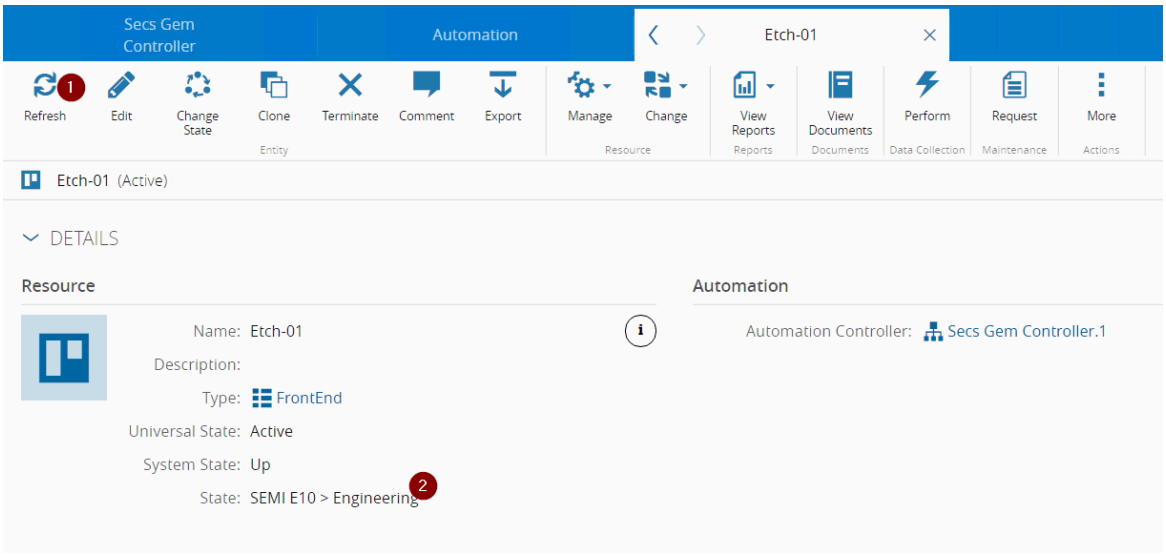
There you can find six colored buttons, each one emitting the status changed event (CEID 1000) with the status variable (SVID 2011) value ranging between 1 and 6 according to the following mapping.

Simulator Status Button	SVID Value
Nonscheduled	1
Productive	6
Standby	5
Engineering	4
Scheduled Down	3
Unscheduled Down	2

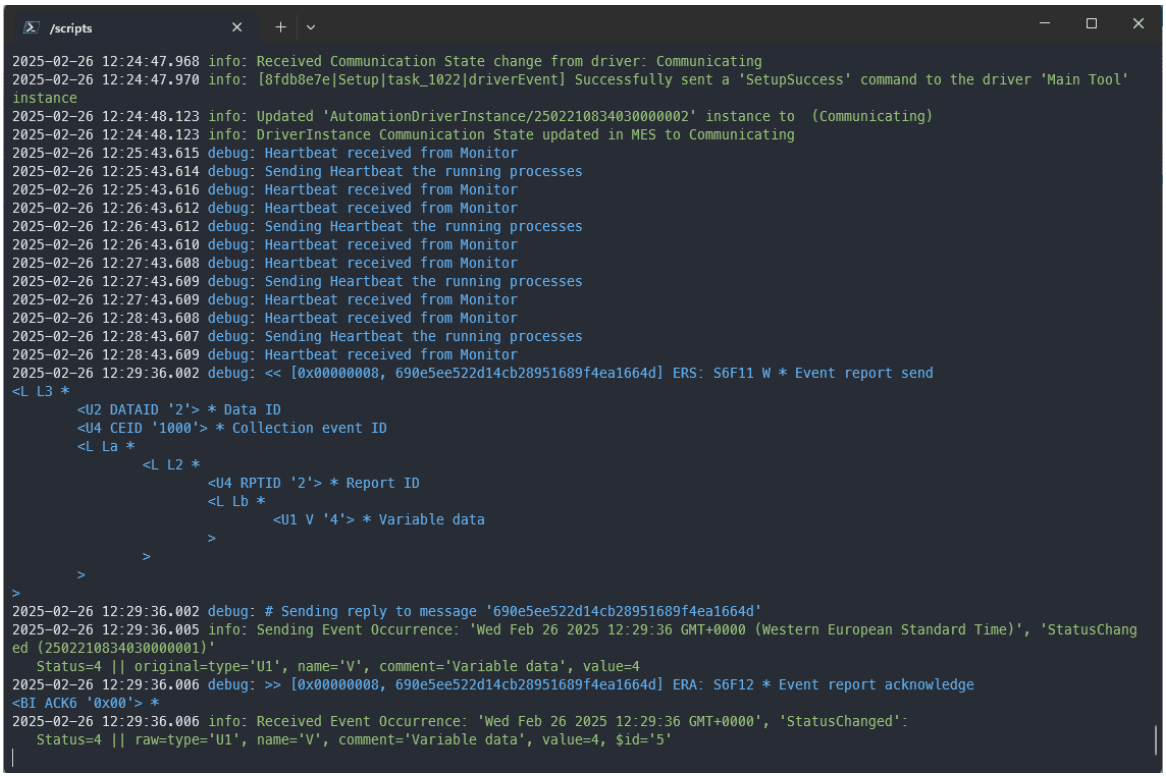
Table: Simulator Status Button

Let's select for example in the blue `Engineering` button and see what changes in the resource details.

Select the `Refresh` button (1) and notice that the SEMI E10 state changed to `Engineering` as well.



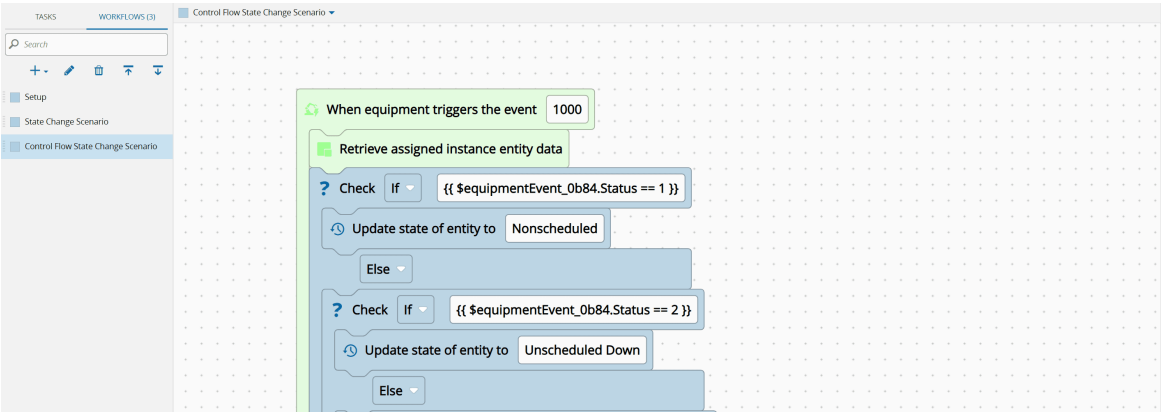
Also in the Automation Manager console, we can see what is going on. The value 4 is sent along with the event 1000, and the switch case handles it, outputting the `Engineering` value.



You can try, by selecting the other buttons of the simulator, and then refreshing the resource details and checking the update of the resource state happening in the MES system.

Equipment Simulator Tests Control Flow

A small example for control flow would be using a similar pattern with the use of the equipment event, adjust state and condition task.



Summary

This tutorial is a simple showcase of a secs/gem integration. The secs/gem driver for Connect IoT supports a plentiful feature set but with a simple example it is easy to understand the fundamentals of how we can integrate a secs/gem interface.

Legal Information

Disclaimer

The information contained in this document represents the current view of Critical Manufacturing on the issues discussed as of the date of publication. Because Critical Manufacturing must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Critical Manufacturing, and Critical Manufacturing cannot guarantee the accuracy of any information presented after the date of publication. This document is for informational purposes only.

Critical Manufacturing makes no warranties, express, implied or statutory, as to the information herein contained.

Confidentiality Notice

All materials and information included herein are being provided by Critical Manufacturing to its Customer solely for Customer internal use for its business purposes. Critical Manufacturing retains all rights, titles, interests in and copyrights to the materials and information herein. The materials and information contained herein constitute confidential information of Critical Manufacturing and the Customer must not disclose or transfer by any means any of these materials or information, whether total or partial, to any third party without the prior explicit consent by Critical Manufacturing.

Copyright Information

All title and copyrights in and to the Software (including but not limited to any source code, binaries, designs, specifications, models, documents, layouts, images, photographs, animations, video, audio, music, text incorporated into the Software), the accompanying printed materials, and any copies of the Software, and any trademarks or service marks of Critical Manufacturing are owned by Critical Manufacturing unless explicitly stated otherwise. All title and intellectual property rights in and to the content that may be accessed through use of the Software is the property of the respective content owner and is protected by applicable copyright or other intellectual property laws and treaties.

Trademark Information

Critical Manufacturing is a registered trademark of Critical Manufacturing.

All other trademarks are property of their respective owners.