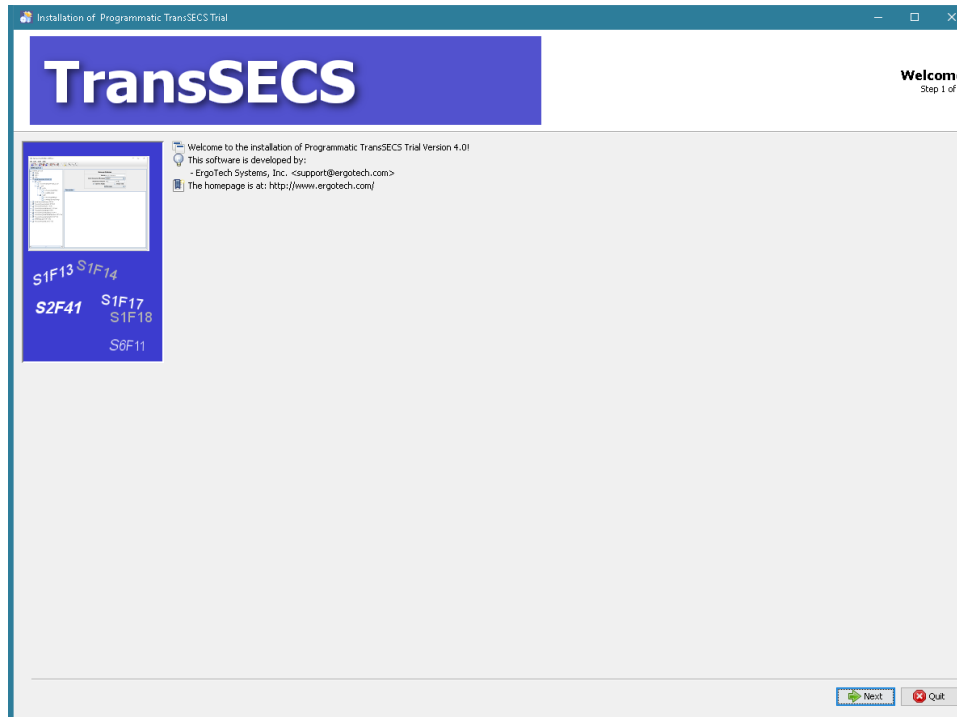




TransSECS for an MQTT Server

Using Servers TransSECS to create an MQTT SECS/GEM Host Server

Installing TransSECS



Double click on the installer and click **Next** when prompted. Once installed, start the TransSECS Builder application (MIStudioSuite/TransSECS/Builder/TransSECS.exe)



33.5
35.7
32.2
30.9
28.2

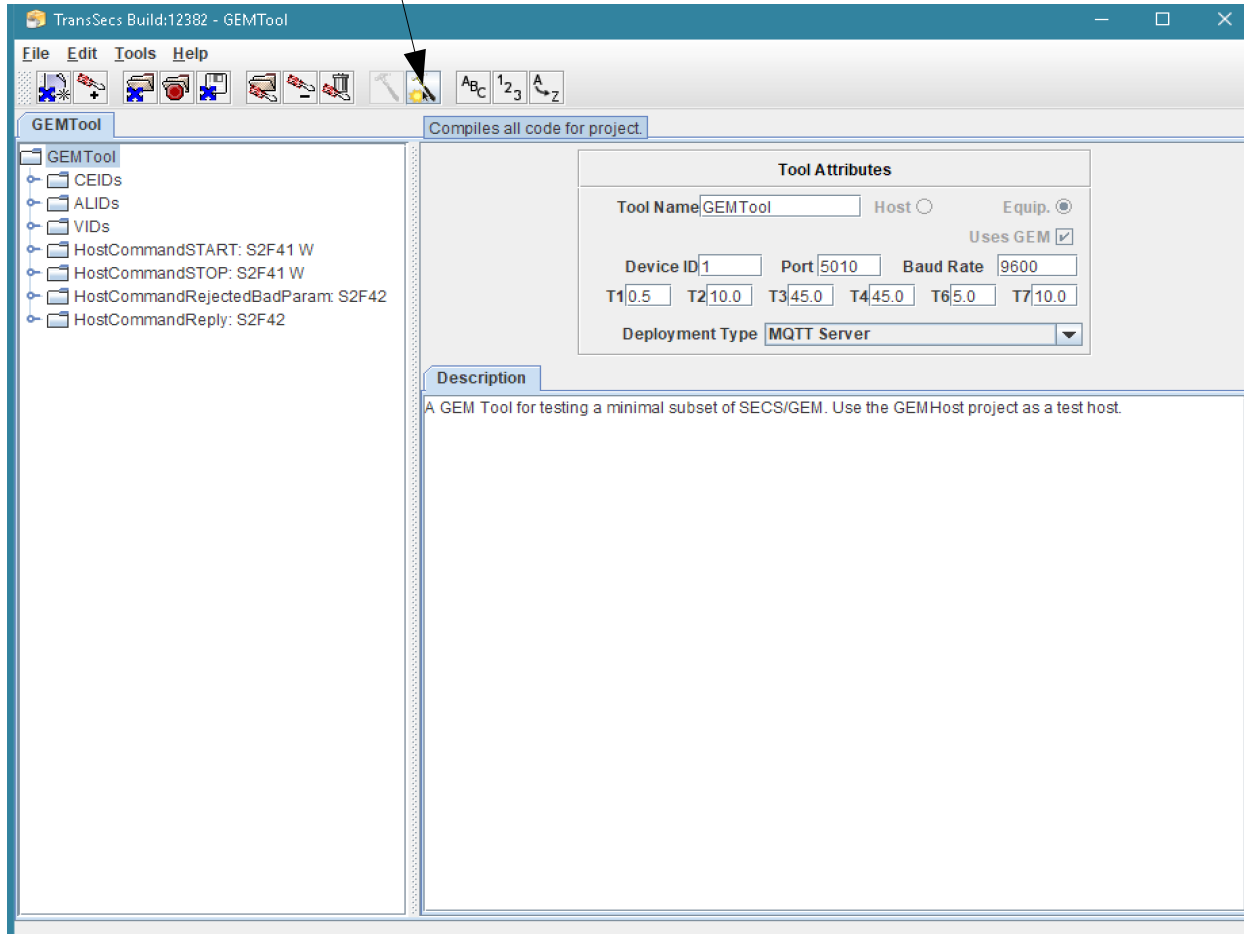
The example GEMTool will be loaded when you start the TransSECS Builder

The tool interface defaults to running on port 5010 (HSMS)

and Device ID 1

The next steps in this tutorial will build this tool so that a simulator is built to test the host. These steps are not needed if you have a tool simulator already for testing.

Press the Hammer/Star button to build the project





33.5
35.7
32.2
30.9
28.8

When the "Compilation" popup closes the build is complete. This may take a minute or so.

The screenshot shows the GEMTool application window titled "TransSecs Build:12382 - GEMTool". The interface includes a menu bar (File, Edit, Tools, Help) and a toolbar. On the left is a project tree with folders like CEIDs, ALIDs, and VIDs, and files for HostCommandSTART, HostCommandSTOP, HostCommandRejectedBadParam, and HostCommandReply. The main area displays "Tool Attributes" for "GEMTool", including fields for Device ID (1), Port (5010), Baud Rate (9600), and Deployment Type (MQTT Server). A "Description" field contains the text: "A GEM Tool for testing a minimal subset of SECS/GEM. Use the GEMHost project as a test host." A "Compilation" dialog box is overlaid on the main window, showing an information icon and the text "Building All Files" with a "Cancel" button. A yellow callout box at the bottom right explains that when this build is complete, the tool simulator will be built in the "GEMToolSimulator" folder.

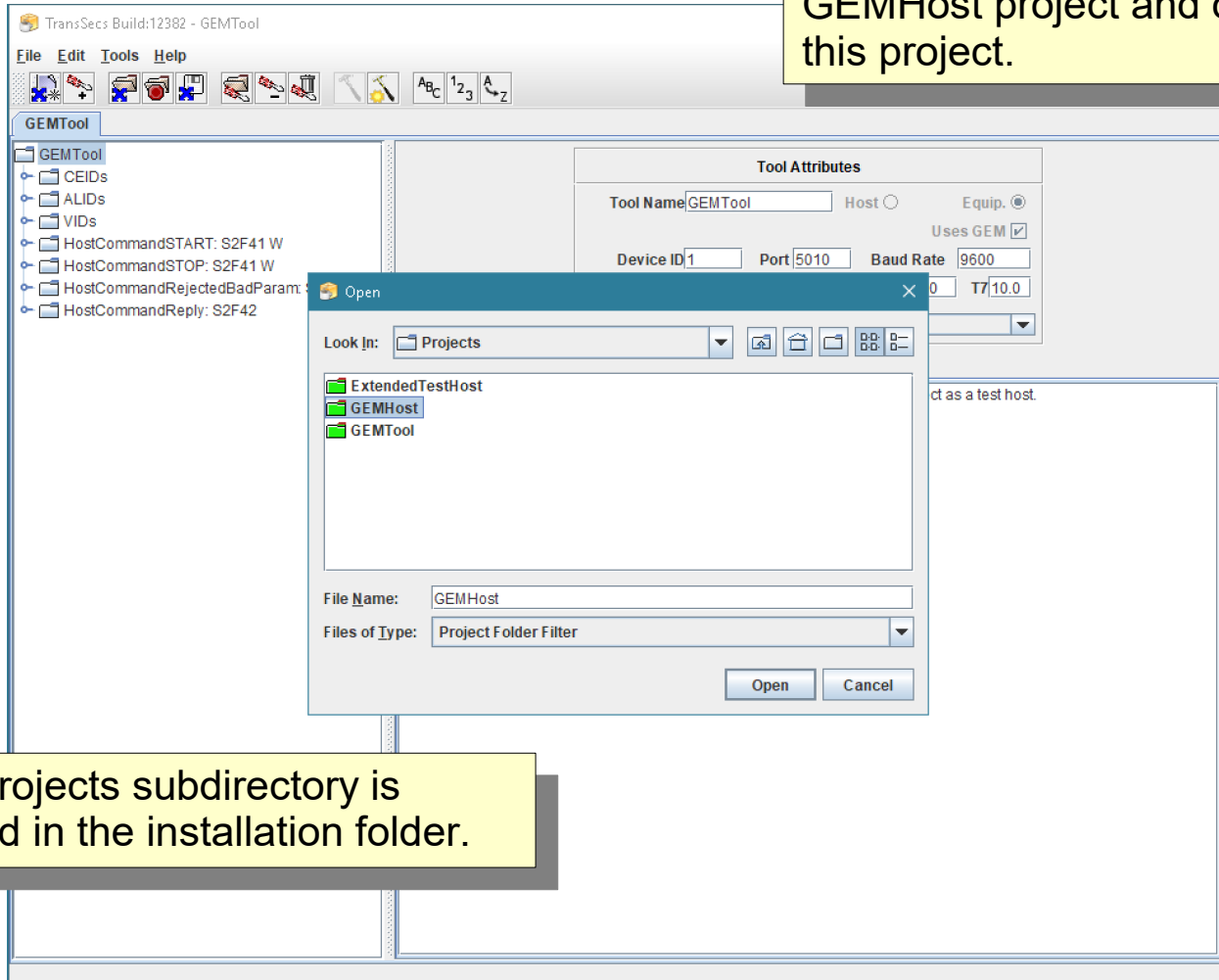
When this build is complete the tool simulator will be built in the GEMTool project directory in the "GEMToolSimulator" folder. This simulator will be used later in this tutorial.



33.5
35.7
32.2
30.9
28.8

Load the GEMHost Project

Use the File "Open Project" menu to browse for the GEMHost project and open this project.



The Projects subdirectory is located in the installation folder.



33.5
35.7
32.2
30.9
28.2

GEMHost Project Loaded

The host defaults to connect to a tool on port 5010 (HSMS)

Tool Attributes

Tool Name: GEMHost Host Equip.

Tool IP Addr.: localhost Uses GEM

Device ID: 1 Port: 5010 Baud Rate: 19200

T1: 0.5 T2: 10.0 T3: 45.0 T4: 45.0 T6: 5.0 T7: 10.0

Deployment Type: OPCUA Server

Description

Sample Host for TransSECS. Use with the GEMTool project.

and Device ID 1

This GEMHost is designed to be used with the GEMTool for this demonstration.

GEMHost Setup

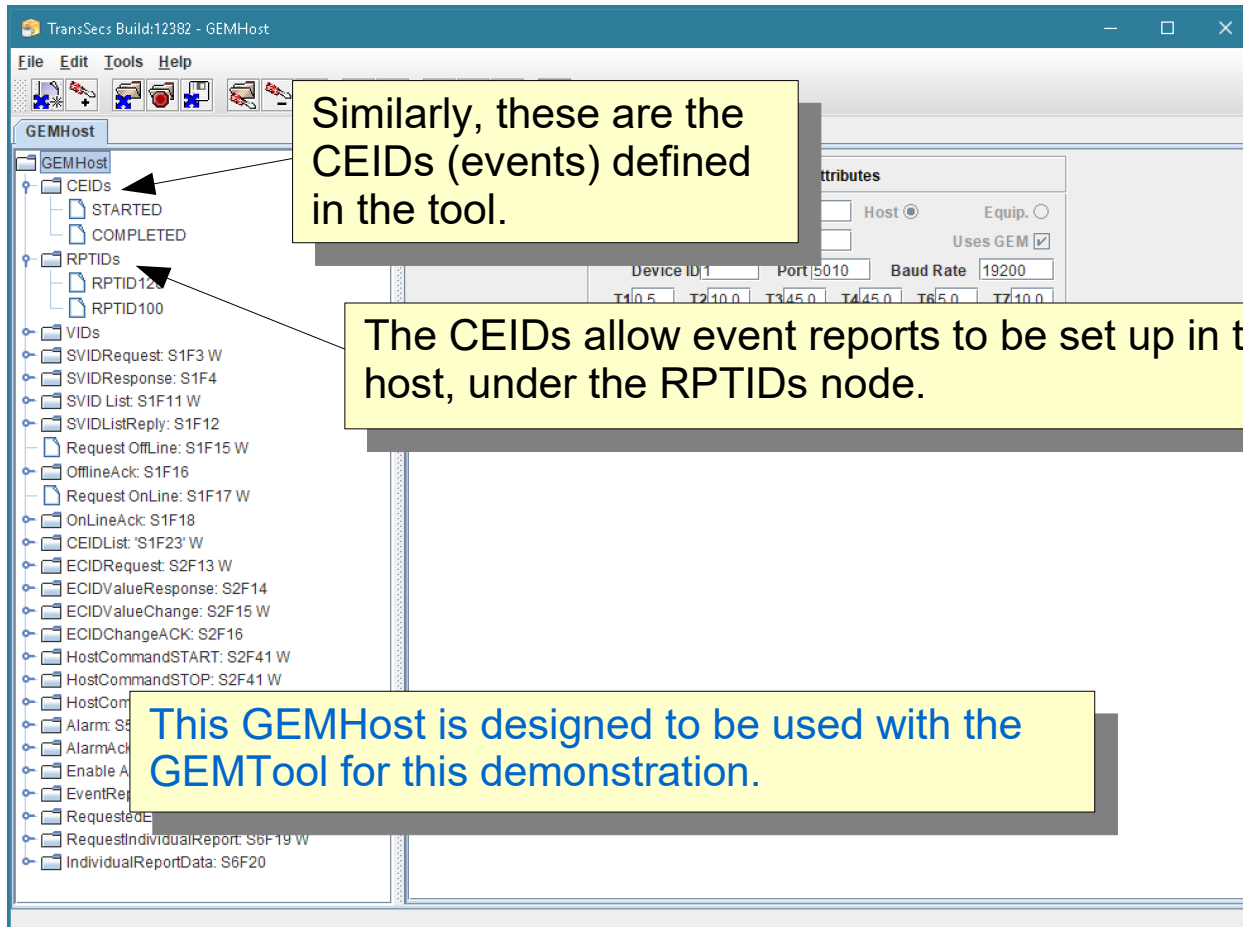
The screenshot shows the 'TransSecs Build:12382 - GEMHost' application window. On the left is a tree view under the 'GEMHost' folder, containing sub-folders for 'CEIDs', 'RPTIDs', and 'VIDs'. The 'VIDs' folder is expanded, showing a list of variables including SetPoint, WaferCount, GasFlow, ProcessTemperature, ControlState, LastControlState, LocalRemoteState, OfflineOnlineState, EnabledAlarms, SetAlarms, EnabledEvents, and several SVID (Setpoint Value Identifier) related variables like SVIDRequest, SVIDResponse, SVID List, SVIDListReply, Request OffLine, OfflineAck, Request OnLine, OnLineAck, CEIDList, ECIDReq, ECIDVal, ECIDVal, ECIDChang, HostCommandSTART, HostCommandSTOP, HostCommandReply, and Alarm. On the right is a configuration panel with fields for 'Device ID:1', 'Port:5010', and 'Baud Rate:19200'. Below these are seven input fields for VID values: T1:0.5, T2:10.0, T3:45.0, T4:45.0, T6:5.0, and T7:10.0. A 'Deployment Type' dropdown is set to 'OPCUA Server'. A 'Attributes' section has radio buttons for 'Host' (selected) and 'Equip.', and a checked 'Uses GEM' checkbox.

These VIDs correspond to the VIDs in the GEMTool.

When the host receives a SECS message from the tool with any of these VIDs, the value of the data in the server will update.

This GEMHost is designed to be used with the GEMTool for this demonstration.

GEMHost Setup



The screenshot shows the GEMHost software interface. On the left is a tree view with the following nodes: GEMHost, CEIDs (STARTED, COMPLETED), RPTIDs (RPTID12, RPTID100), VIDs, SVIDRequest: S1F3 W, SVIDResponse: S1F4, SVID List: S1F11 W, SVIDListReply: S1F12, Request OffLine: S1F15 W, OfflineAck: S1F16, Request OnLine: S1F17 W, OnLineAck: S1F18, CEIDList: 'S1F23' W, ECIDRequest: S2F13 W, ECIDValueResponse: S2F14, ECIDValueChange: S2F15 W, ECIDChangeACK: S2F16, HostCommandSTART: S2F41 W, HostCommandSTOP: S2F41 W, HostCom, Alarm: S, AlarmAck, Enable A, EventRe, Requeste, RequestIndividualReport: S6F19 W, IndividualReportData: S6F20. On the right is a configuration panel with fields for Device ID, Port (5010), Baud Rate (19200), and a table of T1-T7 values. A 'Attributes' section has radio buttons for Host and Equip, and a checked 'Uses GEM' checkbox.

Similarly, these are the CEIDs (events) defined in the tool.

The CEIDs allow event reports to be set up in the host, under the RPTIDs node.

This GEMHost is designed to be used with the GEMTool for this demonstration.



33.5
35.7
32.2
30.9
28.0

GEMHost Setup

TransSecs Build:12382 - GEMHost

File Edit Tools Help

GEMHost

- GEMHost
 - CEIDs
 - RPTIDs
 - VIDs
 - SVIDRequest: S1F3 W
 - SVIDResponse: S1F4
 - SVID List: S1F11 W
 - SVIDListReply: S1F12
 - Request OffLine: S1F15 W
 - OfflineAck: S1F16
 - Request OnLine: S1F17 W
 - OnLineAck: S1F18
 - CEIDList: 'S1F23' W
 - ECIDRequest: S2F13 W
 - ECIDValueResponse: S2F14
 - ECIDValueChange: S2F15 W
 - ECIDChangeACK: S2F16
 - HostCommandSTART: S2F41 W
 - HostCommandSTOP: S2F41 W**
 - <L [2]>
 - <A Command[=STOP]>
 - <L [0]>
 - HostCommandReply: S2F42
 - Alarm: S
 - AlarmAck
 - Enable A
 - EventRe
 - Request
 - Requestron
 - IndividualReportData: S0F20

Message Attributes

Name: HostCommandSTOP

Auto Response Message: <none>

Stream & Function: \$2 F41

Expects Reply Snoop Only

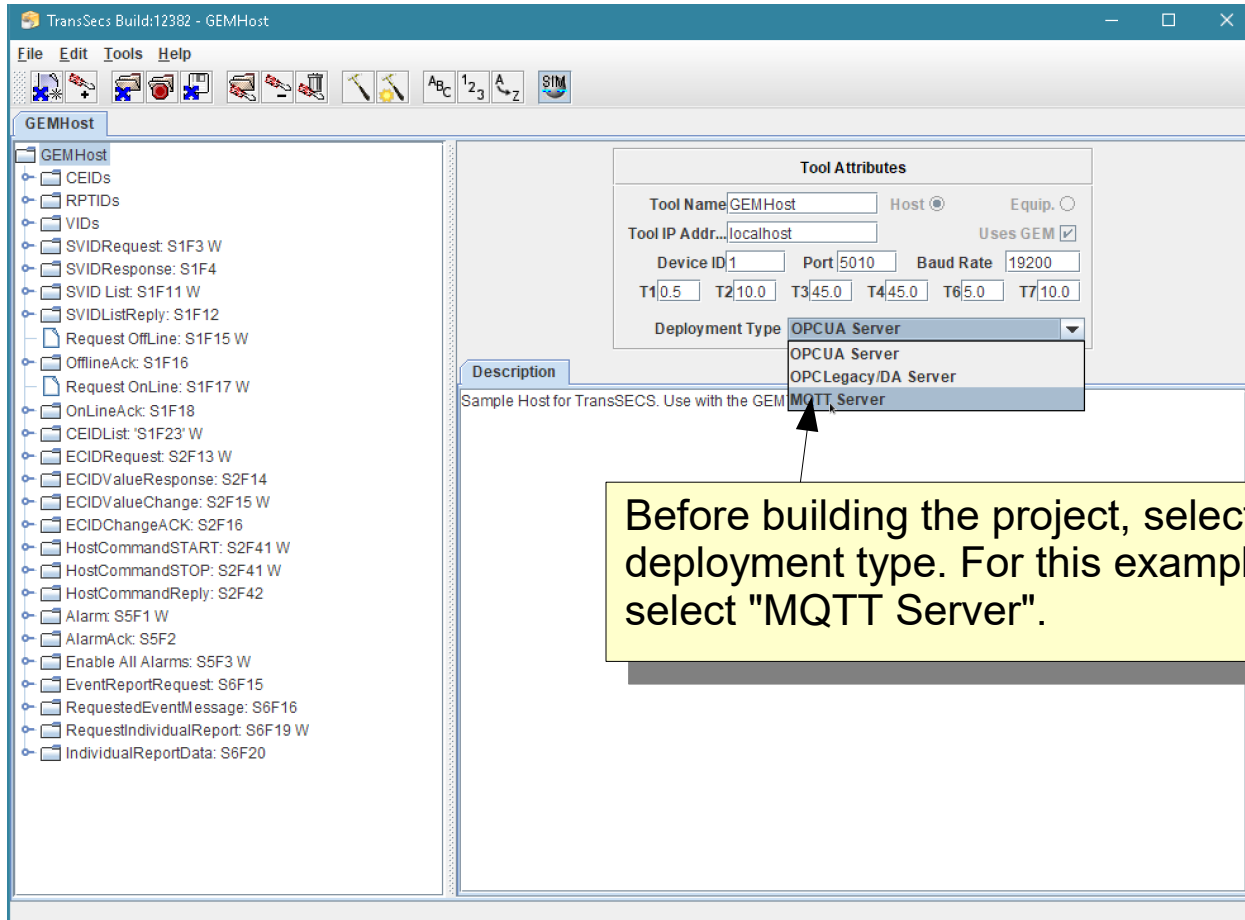
A few simple Host Commands have been defined for testing with the GEMTool which can be sent from the server.

This GEMHost is designed to be used with the GEMTool for this demonstration.



33.5
35.7
32.2
30.9
28.8

Build the GEMHost Project





33.5
35.7
32.2
30.9
28.2

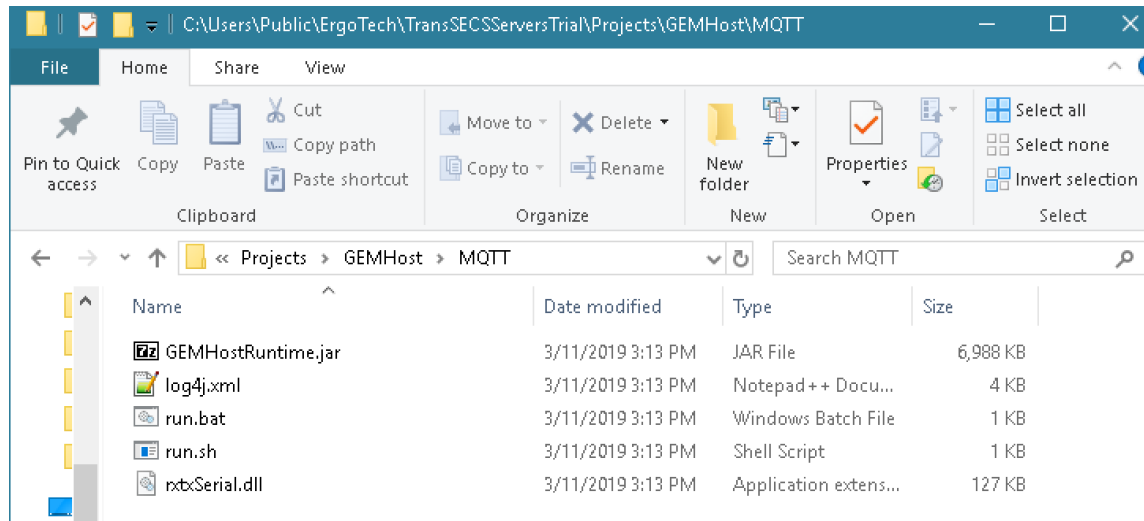
Build the GEMHost Project

To start the MQTT Server build, press the hammer/star button on the menu bar.

The screenshot displays the GEMHost software interface. The main window title is "TransSecs Build:12382 - GEMHost". The menu bar includes "File", "Edit", "Tools", and "Help". The toolbar contains various icons, including a hammer/star icon. The left pane shows a project tree with folders like "CEIDs", "RPTIDs", "VIDs", and various SVID and ECID related files. The right pane shows the "Tool Attributes" dialog box with fields for "Tool Name" (GEMHost), "Host" (selected), "Equip." (unselected), "Tool IP Addr..." (localhost), "Uses GEM" (checked), "Device ID" (1), "Port" (5010), "Baud Rate" (19200), and "Deployment Type" (MQTT Server). A "Description" tab is also visible. A "Compilation" popup window is overlaid on the main window, displaying an information icon, the text "Building All Files", and a "Cancel" button.

The build is underway when the Compilation popup appears. The build will be complete in a minute or two.

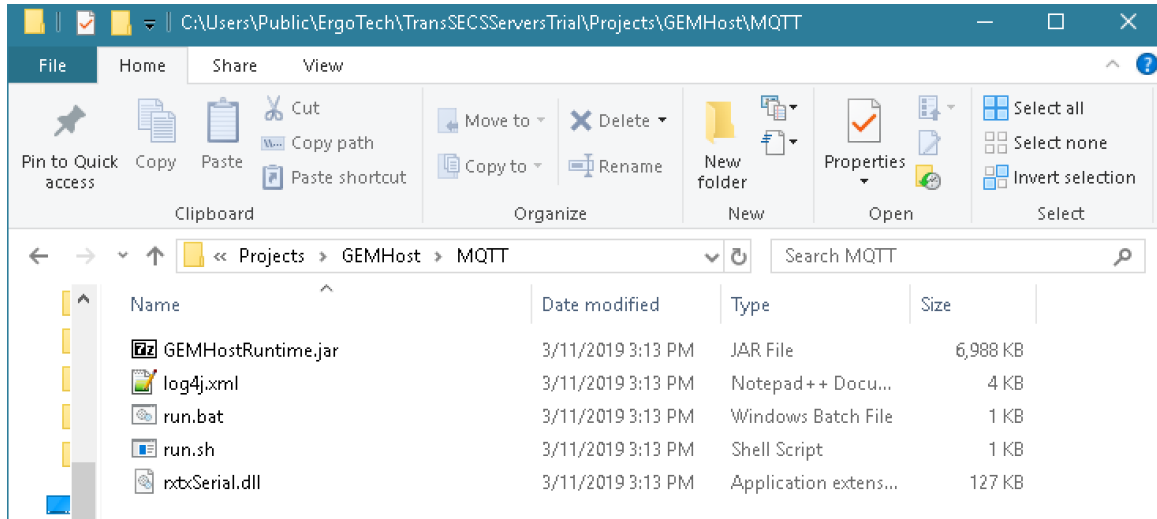
After the code is generated the server code for the tool will be in the Projects/GEMHost/MQTT directory.



Everything you need to run on Windows is in this directory. For Linux systems you will need install rtxSerial on the system and make appropriate changes to the run.sh file.

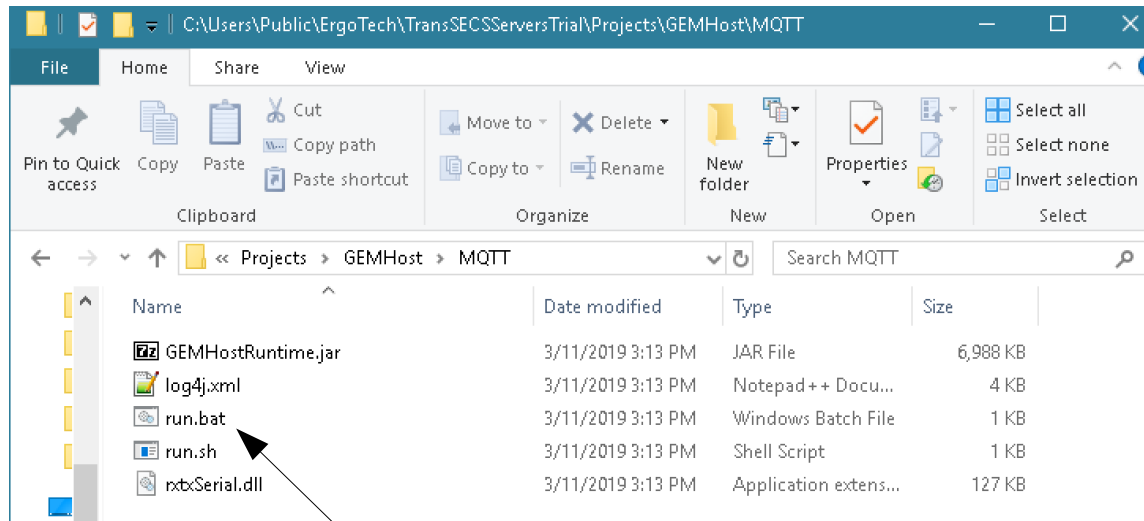
You may need to edit the path to the jre in run.bat if you move the deployment location.

After the code is generated the server code for the tool will be in the **Projects/GEMHost/MQTT** directory.



Before starting the new MQTT SECS/GEM Host, please exit TransSECS Builder to ensure that only one Host application is running.

After the code is generated the server code for the tool will be in the `Projects/GEMHost/MQTT` directory.



Use the run.bat file to open a command shell and start the MQTT Server.



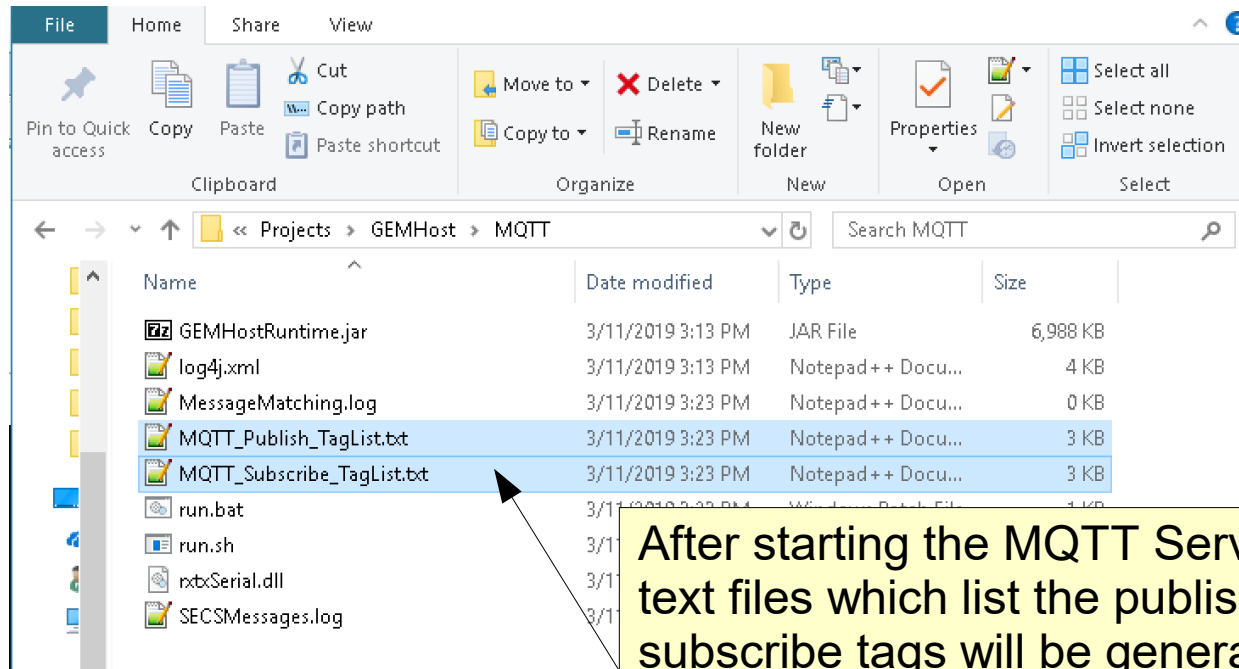
Run the SECS/GEM Interface as a MQTT Server

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Public\ErgoTech\TransSECServersTrial\Projects\GEMHost\MQTT>java deploy.GEMHost.EquipmentController
at ip address localhostStarted GEMHost connecting to localhost on port 5010 with device id 1
Publish Item id: "gemhost/ecidrequest/ecid"
Subscribe Item id: "gemhost/ecidrequest/ecid"
Publish Item id: "gemhost/ecidrequest/responsestatus"
Subscribe Item id: "gemhost/ecidrequest/responsestatus"
Subscribe Item id: "gemhost/ecidrequest/sendmessage"
Publish Item id: "gemhost/svidlistreply/svidlist"
Publish Item id: "gemhost/svidlistreply/responsestatus"
Subscribe Item id: "gemhost/svidlistreply/responsestatus"
Subscribe Item id: "gemhost/configuration/activeport"
Subscribe Item id: "gemhost/configuration/equipmenthostname"
Subscribe Item id: "gemhost/configuration/deviceid"
Subscribe Item id: "gemhost/configuration/activet1"
Subscribe Item id: "gemhost/configuration/activet2"
Subscribe Item id: "gemhost/configuration/activet3"
Subscribe Item id: "gemhost/configuration/activet4"
Subscribe Item id: "gemhost/configuration/activet5"
Subscribe Item id: "gemhost/configuration/activet6"
Subscribe Item id: "gemhost/configuration/activet7"
Subscribe Item id: "gemhost/configuration/activet8"
Subscribe Item id: "gemhost/configuration/baudrate"
Publish Item id: "gemhost/hostcommandstart/ppselectparams"
Subscribe Item id: "gemhost/hostcommandstart/ppselectparams"
Publish Item id: "gemhost/hostcommandstart/responsestatus"
Subscribe Item id: "gemhost/hostcommandstart/responsestatus"
Publish Item id: "gemhost/hostcommandstart/command"
Subscribe Item id: "gemhost/hostcommandstart/command"
Subscribe Item id: "gemhost/hostcommandstart/sendmessage"
Publish Item id: "gemhost/requestedeventmessage/dataid"
Publish Item id: "gemhost/requestedeventmessage/ceid"
```

When you run the generated run.bat, the host will be running as an MQTT server and attempting to connect to a tool on localhost Port 5010 and Device ID 1.

Run the server with the run.bat file.

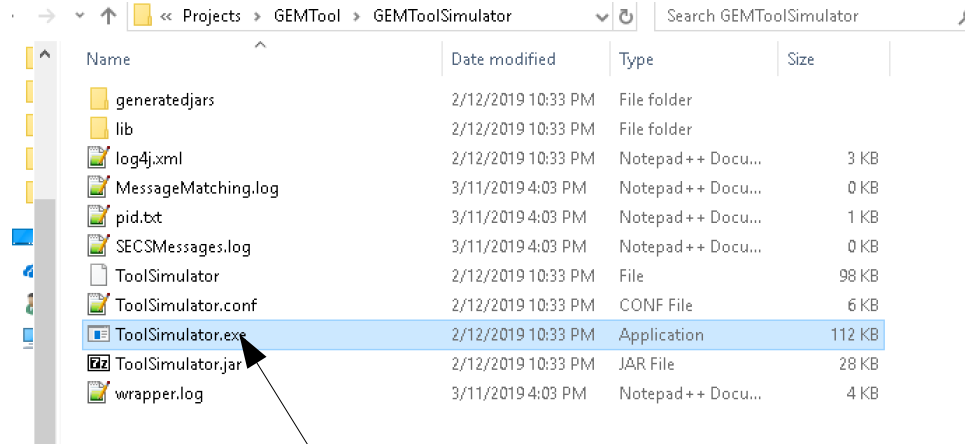


After starting the MQTT Server, two text files which list the publish and subscribe tags will be generated for reference.



33.5
35.7
32.2
30.9
28.0

Starting the Tool Simulator



The simulator for the GEMTool is in the GEMToolSimulator. If this directory does not exist, load the GEMTool into the TransSECS Builder and build the project.



Starting the Tool Simulator

The screenshot shows the GEMTool Simulator window. At the top, there are two buttons: 'Remote Online' and 'Communicating', both with 'Online' and 'Remote' buttons respectively. Below this is a 'Primary' section with a dropdown menu showing 'HostCommandSTART' and a 'Reply' section with a dropdown menu showing 'HostCommandRejectedBadParam'. Underneath are input fields for 'Command' (containing 'START'), 'CPName' (containing 'PPID'), 'CPValue', and 'RecipeName'. The bottom section displays MQTT communication logs:

```
SENT:
S2F36 Accepted.

RECEIVED:
S2F37 W <L[2]
<BOOLEAN 1>
<L[2]
<U4 7501>
<U4 7502>
>
> .

SENT:
S2F38 Accepted.

RECEIVED:
S1F1 W .
```

When the tool simulator starts up, the GEMHost running as an MQTT Server will connect and set up and enable reports, events, and alarms.

MQTT Client Examples

Use your MQTT client to publish to tags in the host server. For example to send an S1F3 for svid 33008 (the VID Clock value) to the tool first publish "33008" to the tag gemhost/svidrequest/svid, then publish a boolean "true" (such as "1") value to the "sendmessage" for this message using tag gemhost/svidrequest/sendmessage.

```
C:\utils\mosquitto>mosquitto_pub -h localhost -t "gemhost/svidrequest/svid" -m "33008"  
C:\utils\mosquitto>mosquitto_pub -h localhost -t "gemhost/svidrequest/sendmessage" -m "1"  
C:\utils\mosquitto>
```



All messages, including host commands, are sent using the "sendmessage" for the message from the list of published tags.

MQTT Client Examples

The S1F3 message sent from the MQTT Server was received by the tool and a reply was sent with the current value of SVID 33008 (the GEM CLOCK value).

The screenshot shows a software interface for MQTT communication. At the top, it says "Communicating" and "Remote". Below that, there are two tabs: "Primary" and "Reply". The "Primary" tab shows "HostCommand START" and the "Reply" tab shows "HostCommandRejectedBadParam". Below these are input fields for "Command" (START), "CPName" (PPID), "CPValue", and "RecipeName".

The main area is a log window showing the following messages:

```
S1F1 W .  
SENT:  
S1F2 <L[2]  
<A 'Model '>  
<A 'Rev1.0'>  
> .  
RECEIVED:  
S1F3 W <L[1]  
<U4 33008>  
> .  
SENT:  
S1F4 <L[1]  
<A '2019031116421340'>  
> .
```

Two yellow callout boxes with arrows point to the log entries:

- "S1F3 received from host" points to the "RECEIVED: S1F3 W <L[1] <U4 33008>" entry.
- "S1F4 sent to host" points to the "SENT: S1F4 <L[1] <A '2019031116421340'>" entry.

At the bottom of the interface is a "Send Message" button.

MQTT Client Examples

Use your MQTT client to subscribe to tags in the host server. For example after sending the S1F3 message the list of VID values in the S1F4 can be subscribed to using the tag "gemhost/svidresponse/svidlist"

```
C:\utils\mosquitto>mosquitto_sub -h localhost -t "gemhost/svidresponse/svidlist"  
< "values": [ "2019031116421340" ] , type:"20" >
```

The value returned for this subscribe is in JSON format (see the following notes for JSON list formats)



Notes on Host MQTT Servers : Reports

For an MQTT host, reports and events are published as JSON format.

for example, a subscription to the report:

'gemhost/variables/rptid/rptid9'

may yield a response such as this:

```
Client mosqsub|8236-hidden received PUBLISH (d0, q0, r0, m0,
'gemhost/variables/rptid/rptid9', ... (176 bytes)) { "RPTID9":
[ {"GasFlow":"20.419", "type":44 }, {"ProcessTemperature":"56.654",
"type":44 }, {"WaferCount":"1135", "type":54 } ], rptid:100,
timestamp:"2019-02-12 14:47:55.321"}
```

Notes on Host MQTT Servers : Reports

Which is this format:

```
{"reportName":[  
{"GasFlow":"1.2", "type":34},  
{"AString":"Test", "type":20},  
{"BinarValues":[5,6,7,8], "type":10},  
{"WaferCount":"15", "type":54}  
] rptid=109 timestamp:"2019-02-12 14:29:13.306" }
```

The values are provided as valuenam:value type:secsformattype



Notes on Host MQTT Servers: Events

Events are published as JSON; for example, the format for the request ('gemhost/variables/ceid/started'):

```
Client mosqsub|9161-hidden received PUBLISH (d0, q0, r0, m0,
'gemhost/variables/ceid/started', ... (206 bytes)) { "STARTED":
[ { "RPTID9": [ { "GasFlow": "17.406", "type": 44 },
{ "ProcessTemperature": "49.293", "type": 44 }, { "WaferCount": "1203",
"type": 54 } ], rptid: 100 } ], ceid: 7501, timestamp: "2019-02-12
14:53:35.330" }
```

Notes on Host MQTT Servers: Events

Reports are provided as an array so that there can be multiple:

```
{ "STARTED": [  
  { "RPTID9": [ {"GasFlow":"17.406", "type":44 },  
{"ProcessTemperature":"49.293", "type":44 }, {"WaferCount":"1203",  
"type":54 }], rptid:101} ,  
  { "RPTID10": [ {"ControlState":"5", "type":54 } ], rptid:101}  
], ceid:7501, timestamp:"2019-02-12 14:53:35.330"}
```

Where "STARTED" is the CEID name. The list is the list of reports associated with the CEID.



That's it.

The GEMHost interface is not complete. More code must back the MQTT client and more messages may be added to the GEMHost, but this simple example should get you going. You will need to handle host command replies in the client application. You can also set up simple automatic recipe handling, or add recipe messages to the GEMHost and handle these replies in your client application.