MES Template How-to

Add basic MES functions (plant model, downtime and production events, OEE) to a FrameworX solution using the built-in MES template.

Starter template, version 10.1.6. Read *Known limitations* before using it in production. The template does not create the MES database tables, and the OEE Performance factor is not yet computed from production data.

What the template contains

The MES template is a starting point, not a finished MES. It gives you an ISA-95 plant model, event recording for downtime and production runs, an OEE calculation and three starter screens. Event data is stored in a PostgreSQL database; the plant model lives in the Unified Namespace.

Module	Objects	Purpose
UNS: UserTypes	`MES_Plant`, `MES_Area`, `MES_Line`, `MES_Equipment`, `MES_EquipmentType`, `MES_Variable`, `MES_Shift`, `MES_Reason`, `MES_Event_Downtime`, `MES_Event_Production`, plus the enumerations `MES_EquipmentMode`, `MES_EquipmentStatus`, `MES_ReasonCategory`, `MES_VariableDataType`	The plant model and its catalogs.
UNS: Tags	`MES/EquipmentTypes` (Filler, Capper, Packer, Palletizer, Generic), `MES/Reasons/Standard` (36 downtime reasons in Planned, Unplanned, Quality and External), `MES/Shifts/Shift_Default`, `MES/Runtime/CurrentShiftCode`, `MES/Runtime/KpiEquipmentList`	Seed catalogs and runtime control tags.
Scripts: Classes (Server)	`MES_Util`, `MES_Equipment`, `MES_Event`, `MES_Event_Production`, `MES_Shift`, `MES_OEE`, `MES_OEE_Availability`, `MES_OEE_Performance`, `MES_OEE_Quality`, `MES_Curation`, `MesRuntimeTools`	The MES library: events, OEE, shifts, corrections, AI tools.
Scripts: Tasks	`MES_SeedDefaults` (server startup), `MES_ShiftTransition` (every 60 s), `MES_OeeScheduler` (every 5 min)	Seed catalog values, close and reopen production runs at shift change, compute hourly OEE.
Datasets	DB `MES_PG` (PostgreSQL, Npgsql); queries `Q_MES_Downtimes`, `Q_MES_KPI_Latest`	Connection to the MES database and the queries behind the screens.
Displays	`MES_PlantModel`, `MES_OEE_Dashboard`, `MES_Downtime_List`	Plant navigation, latest-hour OEE, downtime review.
Security: Users	`MES_Service_Alert`, `MES_Service_Calculation`, `MES_Service_Email`, `MES_Service_EventDetection`	Service accounts to name as the actor in script calls.

Requirements

- FrameworX 10.1.6 or later. The template is available for Multi-Platform (.NET 10) and Windows (.NET Framework 4.8) solutions. On Windows, the AI tools class does not compile (see *Known limitations*).
- A PostgreSQL server reachable from the FrameworX server, with a database for MES data and the MES tables created (see *Known limitations*).
- The Npgsql provider, used by the `MES_PG` connection.

Option A: start a new solution from the template

1. In Solution Center, click **New...**.
2. Choose the product series and the platform (**Multi-Platform** or **Windows**).
3. Enter the solution name and select the **MES** template.
4. Click **Create New Solution**.

The solution opens with everything listed above. Continue at *Connect the database*.

Option B: add MES to an existing solution

Use this when you already have a solution and want to add the MES objects to it. You move them with a FrameworX plugin (.dbplugin). See Solution Import Plugin Reference.

1. Create a temporary solution from the MES template (Option A).
2. In that solution, go to **Solution Export** and mark the MES objects:
    - UNS: the `MES_*` UserTypes and all tags under `MES/`. The enumerations and asset folders are added automatically as dependencies.

- Scripts: every class whose name starts with `MES_`, plus `MesRuntimeTools`; the tasks `MES_SeedDefaults`, `MES_ShiftTransition`, `MES_OeeScheduler`.
- Datasets: the DB `MES_PG` and the queries `Q_MES_Downtimes`, `Q_MES_KPI_Latest`.
- Displays: `MES_PlantModel`, `MES_OEE_Dashboard`, `MES_Downtime_List`.

3. Export the marked objects as a plugin. The plugin name accepts letters, digits and underscore only (for example `MES_Starter`).
4. Back up your project solution, open it and go to **Solution Import Plugin**. Select the plugin and import with **Merge to the Solution**. If an object with the same name already exists, the import asks whether to replace it.
5. Run **Build** with **Rebuild All** checked. Expect 0 errors; warnings about a duplicate `using System` are harmless.
6. Create the four `MES_Service_*` users in **Security Users** if you want them: users are not carried by plugins.

The MES screens are not added to any menu. Add them to your navigation (header buttons or menu) or open them with `@Client.OpenDisplay("MES_PlantModel")`.

Connect the database

1. Go to **Datasets DBs** and open `MES_PG`. The connection string ships with placeholders:

   ```
   Host=YOUR_PG_HOST;Port=5432;Database=mes;Username=YOUR_USER;Password=${SecuritySecrets.MES_PG_Password}
   ```

   Replace the host, database and user. Keep the password out of the string: create a Security Secret named `MES_PG_Password`.
2. The scripts read the connection string from `MES_PG`. If it is empty they fall back to the environment variable `MES_PG_CONNSTR` on the server.

Build the plant model

The template ships the types, not your plant. Create one tag per plant element using the MES UserTypes:

Level	UserType	Key members
Plant	`MES_Plant`	Code, TimeZone, DefaultShiftPattern
Area	`MES_Area`	Code, PlantCode
Line	`MES_Line`	Code, AreaCode
Equipment	`MES_Equipment`	Code, TypeCode, ProdRate, RateUnit, Mode, Status, KpiProduction, KpiHour, KpiShift, KpiDay
Variable (optional)	`MES_Variable`	Code, EquipmentCode, HistorianSourceTagname, EngUnits

The **Code** member is the key every event and KPI row uses, so give each equipment a unique code and do not change it after events are recorded. Bind `Mode` and `Status` to your PLC through device points, so the equipment state is live.

Turn on the OEE schedule

Write the equipment codes, comma-separated, to the tag `MES/Runtime/KpiEquipmentList` (for example `FILLER01,CAPPER01`). The task `MES_OeeScheduler` runs every 5 minutes and computes OEE for the current hour for each code in the list. With the list empty, it does nothing.

Shifts

`MES/Shifts/Shift_Default` is a single 24/7 shift. For a multi-shift plant, create one `MES_Shift` tag per shift under `MES/Shifts` and set:

- **StartTime**: local start, `HH:mm`.
- **DurationMin**: length in minutes.
- **DaysOfWeek**: seven characters starting on Monday, `-` for days off. `MTWTF--` is weekdays only.
- **TimeZone**: leave blank to use the server time zone.

At each shift change, `MES_ShiftTransition` closes every open production run and opens a new one with the same product, so production totals split by shift.

Record events from your scripts

The MES classes are static and are called with their full namespace from Server-domain scripts (Script Tasks or Server classes). Every write takes an `actor`: the user or service account responsible, which goes to the Audit Trail.

Downtime

```
// Machine stopped: open a downtime event and keep its id
long id = MES.Event.Event.OpenDowntime("FILLER01", DateTime.UtcNow, "ConveyorJam",
"MES_Service_EventDetection");
@Tag.Line1/Filler/DowntimeId = id;

// Machine running again: close it
MES.Event.Event.CloseDowntime(@Tag.Line1/Filler/DowntimeId, DateTime.UtcNow, "MES_Service_EventDetection");
```

Other calls: `GetOpenDowntimes(equipmentCode)`, `GetDowntimesByTimeRange(equipmentCode, from, to)`, `GetDowntimePareto(equipmentCode, from, to)` (pass an empty code for all equipment).

Production runs

```
// Start a run (one open run per equipment)
long run = MES.Event.Production.Production.Open("FILLER01", DateTime.UtcNow, "SKU-500ML",
"MES_Service_EventDetection");

// Add counts as they arrive (deltas, not totals)
MES.Event.Production.Production.IncreaseCounter(run, 120, 3, "MES_Service_EventDetection");

// End the run
MES.Event.Production.Production.Close(run, DateTime.UtcNow, "MES_Service_EventDetection");
```

`Open` fails if the equipment already has an open run. `IncreaseCounter` accepts only an open run and deltas of zero or more, not both zero.

Corrections

`MES.Curation.Curation.Recategorize(eventId, newReasonCode, actor)` changes the reason of a downtime event and recomputes that hour's OEE. `MES.Curation.Curation.Edit(eventId, fieldName, newValue, actor)` changes `reason_code`, `operator_name` or `notes`. Both record the before and after values.

How OEE is calculated

OEE = Availability × Performance × Quality, per equipment and per time bucket. The results are stored in the MES database.

Factor	Formula
Availability	(planned production time - unplanned downtime) / planned production time. Downtime with a reason in the Planned category is removed from planned production time; any other downtime counts as unplanned.
Performance	Not yet computed from production counts in 10.1.6. See *Known limitations*.
Quality	good count / (good + reject) over the production runs in the bucket. 1.0 when there is no production data.

To compute on demand: `MES.OEE.OEE.Compute(code, "Hour", start, end)`. To read the last stored value: `MES.OEE.OEE.GetLatestOEE(code, "Hour")`.

The starter screens

Display	Shows
`MES_PlantModel`	Asset tree rooted at `MES`. Selecting an equipment sets the client context used by the other two screens.
`MES_OEE_Dashboard`	Latest hourly OEE and its three factors for the selected equipment (query `Q_MES_KPI_Latest`). This is the startup page of a new solution.
`MES_Downtime_List`	Downtime events of the selected equipment in the client time range (query `Q_MES_Downtimes`). The Edit and Recategorize buttons are placeholders: wire them to `MES.Curation` calls in your own dialog.

AI tools

The class `MesRuntimeTools` exposes MES read and write operations (plant model, equipment state, downtime and production lists, OEE, Pareto, reason tree, open/close downtime, corrections) as tools for AI clients connected to the FrameworX AI Runtime. Custom tools must be enabled in the solution settings for the runtime to load them. In 10.1.6 the class compiles only in Multi-Platform solutions.

Known limitations

Limitation	What to do
The template does not create the MES database tables (downtime events, production events, KPI data, curation log).	Contact Tatsoft support for the database schema before running the MES functions.
The OEE Performance factor is a placeholder value between 0.85 and 1.0, not a calculation from production counts.	Treat OEE and Performance values as indicative only.
Reason categories (Planned, Unplanned) are not read from the `MES/Reasons` tree. Downtime is counted as unplanned unless its reason is in the built-in planned list.	Expect Availability to be lower than actual when planned stops are recorded.
In Windows (.NET Framework 4.8) solutions, the `MesRuntimeTools` class fails to build with error CS0234 (`System.Text.Json` is not available). The other MES classes build normally.	If you do not use the AI tools, delete the `MesRuntimeTools` class and rebuild. To use them, create the solution as Multi-Platform.
The starter screens have no navigation menu, and the downtime Edit and Recategorize buttons are not wired.	Add navigation and your own correction dialog.

Troubleshooting

Symptom	Cause	Fix
Every MES call fails with a connection-string error in the trace	`MES_PG` not configured and no `MES_PG_CONNSTR`	Set the connection string (see *Connect the database*).
OEE screens stay empty	`MES/Runtime/KpiEquipmentList` is empty, or no equipment selected	Fill the list; select an equipment in `MES_PlantModel`.
"already has an open production run"	`Open` called twice for the same equipment	Close the open run first, or reuse its id.
Errors after plugin import	Scripts not rebuilt	Build with Rebuild All.