**Integrate IHM with CureMD Domain API’s**

**Add following dll’s references in your project:**

**DLL’s are available in CureMD.Domain.API’s project:**

**Path: /src/Infrastructure/ExternalReferences/IHM**

* Elastic.Clients.Elasticsearch.dll (8.15.6.0) (**if using Configurator.Core no need to add**)
* Elastic.Transport.dll (0.4.22.0) (**if using Configurator.Core no need to add**)
* RabbitMQ.Client.dll (6.8.1.0) (**If using EventBusRabbitMQ in your project, no need to add**)
* RestSharp (112.0.0.0) (**immunization/history forecast/commonwell/dentalsoft flow only**)
* NHapi.Base.dll (**immunization/history forecast/commonwell/dentalsoft flow only**)
* NHapi.Model.V23.dll (**immunization/history forecast/commonwell/dentalsoft flow only**)
* NHapi.Model.V25.dll (**immunization/history forecast/commonwell/dentalsoft flow only**)
* NHapi.Model.V231.dll (**immunization/history forecast/commonwell/dentalsoft flow only**)
* NHapi.Model.V251.dll (**immunization/history forecast/commonwell/dentalsoft flow only**)
* EncoderParserCore.dll
* IHubTransactionManager.BL.dll
* IHubTransactionManager.Data.dll
* IHubTransactionManager.dll
* IHubTransactionManager.Models.dll

**appsettings.json**

**Add below object in appsettings.json.**

"IHub": {

"IHM": {

"ApplicationConstants": {

"IHAuthenticateUrlTestHubV2": "2539353D3271626E223839243930532E24387851263F2426321C303F206E0238463622242C222E1E4731067007292425225B27296D2D26232F2E35467C242D252C57",

"IHAuthenticateUrlProdHubV2": "2539353D3271626E223E38236535473F2426321C3D2835641F5C27353F272A35573B252024252E38463A243B612E2E25385730246D3A26262425",

"ApiBasePathTest": "2539353D3271626E223839243930532E24387851263F2426321C303F206E0238463622242C222E1E473106700F3D2864",

"ApiBasePathProd": "2539353D3271626E223E38236535473F2426321C3D2835641F5C27353F272A35573B25202C312279"

}

},

"RabbitMQ": {

"Server": {

"Type": "Local",

"Host": "localhost",

"Port": "5672",

"Username": "guest",

"Password": "guest"

},

"Queue": [

{

"Type": "Reprocess-Outbound",

"Name": "HubConnect-Reprocess-Outbound-11x",

"Direction": "Outbound"

},

{

"Type": "Realtime-Outbound",

"Name": "HubConnect-Realtime-Outbound-11x",

"Direction": "Outbound"

}

]

}

}

**Program.cs**

**1- Add the required references**:

using IHubTransactionManager;

2-**Register Services:**

builder.Services.RegisterIHMServices(builder.Configuration);

**The class where the function is actually called.**

**3. Add the required references**

Make sure to include the following namespaces at the top of your file:

using IHubTransactionManager.HL7MessageHandler.Interfaces;

using IHubTransactionManager.Models.RequestParameterView;

using IHubTransactionManager.Models.Enums;

**4. Declare a private read-only field in the class**

Add a private read-only field for the IProcessor interface in your class. This field will be used to interact with IHM dll.

private readonly IProcessor \_processor;

**5. Inject the IProcessor interface in the constructor**

Inject the IProcessor interface via the constructor to access the IHM transaction manager. This allows the class to handle dependency injection.

public PatientDemo(IServiceProvider serviceProvider, IProcessor processor)

{

\_processor = processor;

}

**6. Call the IHM function with required parameters**

Within your logic for saving a patient, you'll call the IHM function after the patient details have been saved successfully. Below is an example of how to do this:

public async Task<bool> Save (PatientDetails PatientDetails)

{

//Patient Save Logic

// If Patient Save logic successfully executed then trigger HL7 message

// IHM Integration

**9. Create the IHM transaction request model**

Next, create a TransactionRequestView model and pass the appropriate parameters to the IHM SendAsync function. This is where you define the context for the HL7 message, such as the user ID, database name, and component type.

try

{

TransactionRequestView requestView = new()

{

UserId = PatientDetails.UserId, // Logged-In User Id

DatabaseName = PatientDetails.DatabaseName, //Current Practice Database Name

DatabaseServerName = PatientDetails.ServerName, //Current Server Name

ComponentId = PatientDetails.ComponentId, //In patient demo case it is PatientId

ComponentType = ComponentTypeEnum.PatientDemo,

PartnerCategory= PartnerCategoryEnum.HIE, //default value HIE, if not selected.

Trigger = "Add Patient" //It represents from which flow HL7 message triggers

};

await \_processor.SendAsync(requestView);

}

catch(Exception ex)

{

//use file logging for debugging purpose.

}

}

**Note:**

If you need to handle the response, call this function with await; otherwise, call it without await to let it run synchronously.

await \_processor.SendAsync(requestView);

**Mappings:**

-**ComponentTypeEnum and PartnerCategoryEnum Enums mapping**

|  |  |  |
| --- | --- | --- |
| **Flow/Page** | **ComponentTypeEnum** | **PartnerCategoryEnum** |
| Patient Demographics | PatientDemo | HIE |
| Scheduler | PatientAppt | HIE |
| Sign Provider Note | ProvideAndRegisterDocumentSet | HIE |
| ESuper-Bill | FinancialTrans | HIE |
| Lab Orders | LabOrder | Lab |
| Radiology Orders | RadiologyOrder | Radiology |
| Immunization | Immunization | StateRegistry/HIE |
| HistoryForecast | HistoryForecast | StateRegistry/HIE |
| Lab Results | LabResult | HIE |
| Radiology Results | RadiologyResult | HIE |
| Clinical Notes (Progress Notes) | ClinicalNotes | HIE |

For reference please review this pull request:

[Pull Request 53118](https://devops.curemd.com/CureMD10g/11g/_git/CureMD%20Domain%20APIs/pullrequest/53118): Integrate IHM with CureMD Domain API's