IPS®SYSTEMS Integration Strategy

General Document

IPS-Intelligent Process Solutions GmbH, Dr.-Gustav-Adolph-Str. 2, 82049 Pullach (Munich), Germany

Table of Contents

1 Introduction

This document describes basic rules, principles and guidelines for integration between IPS®SYSTEMS and other external systems. It does not describe in detail all interfacing scenarios, mandatory technical requirements neither design specification.

It should be taken into consideration that integration architecture, use cases and interfacing requirements can vary. That's why the purpose of this documents is only to give formal introduction and to explain integration concept on high level.

2 IPS®SYSTEMS Integration Platform

The following figure represents high-level architecture of IPS proposed solution integrated within clients existing infrastructure. Legend is given on the top of the picture.

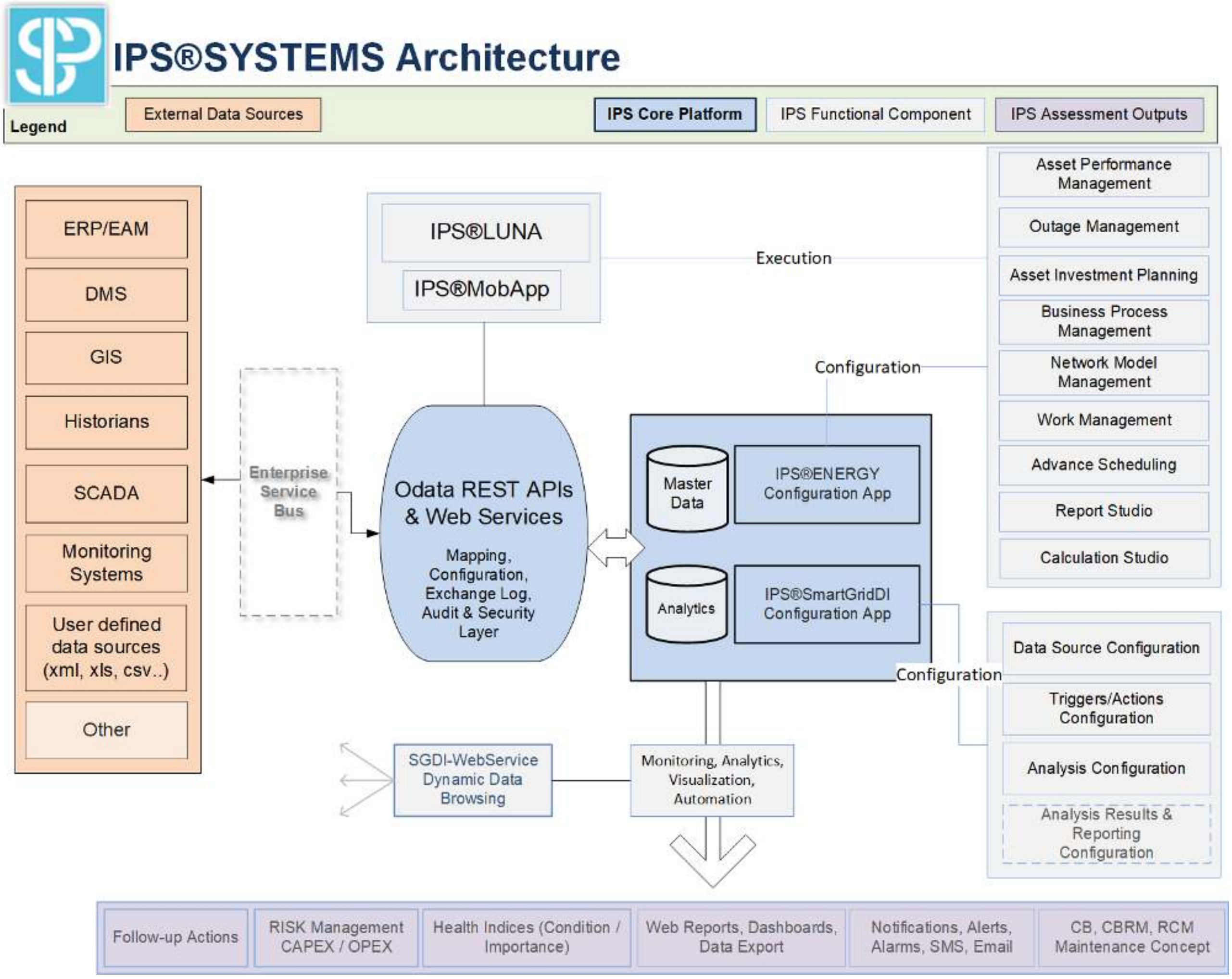

IPS®SYSTEMS Architecture
Legend
External Data Sources
IPS Core Platform
IPS Functional Component
IPS Assessment Outputs
ERP/EAM
DMS
GIS
Historians
SCADA
Monitoring Systems
User defined data sources (xml, xls, csv..)
Other
Enterprise Service Bus
IPS®LUNA
IPS®MobApp
Execution
Configuration
Odata REST APIs & Web Services
Mapping, Configuration, Exchange Log, Audit & Security Layer
Master Data
IPS®ENERGY Configuration App
Analytics
IPS®SmartGridDI Configuration App
Configuration
Asset Performance Management
Outage Management
Asset Investment Planning
Business Process Management
Network Model Management
Work Management
Advance Scheduling
Report Studio
Calculation Studio
Data Source Configuration
Triggers/Actions Configuration
Analysis Configuration
Analysis Results & Reporting Configuration
SGDI-WebService Dynamic Data Browsing
Monitoring, Analytics, Visualization, Automation
Follow-up Actions
RISK Management CAPEX / OPEX
Health Indices (Condition / Importance)
Web Reports, Dashboards, Data Export
Notifications, Alerts, Alarms, SMS, Email
CB, CBRM, RCM Maintenance Concept

Figure 1: IPS®SYSTEMS Solution Architecture

IPS®SYSTEMS Solution Architecture is designed with clear separation between Database Server (central data repository) and Application Server where service-based components are hosted.

Application Server hosts web and windows services and critical application business logic components, while Database Server only stores data used by the system.

As well all automatic condition based and/or time based (background) processing is being done on the application server where smart IPS services are installed.

NOTE: Detail specification for implementation of all required requirements, like interfacing rules and mapping, analytics configurations, data inputs etc., are subject of separate detailed design document.

IPS data models are predefined per asset group (transformer, breaker, protection, linear assets), location type and also data input types (test results, monitoring results, inspections etc.), however, at the same time, the data store models are very flexible and can be easily adjusted or expanded to fit specific requirements.

IPS supports audit of all changes, data quality analysis, conditional notifications and alerts for incoming/outgoing data, statistics etc. and has successful implementation of interfaces with near real time data, monitoring data, test results, ERP systems (SAP/Maximo/Ventyx...), big data systems, OSI Soft, CAPE, PSSE.

IPS strategy for integration with real-time systems - IPS never connects directly to devices but always imports data from other (centralized) systems that are actually data concentrators, monitoring systems or from Historians or from Big Data Lake (e.g., Hadoop Cloudera) for our analysis. However, it can be one or more systems such as these that store different values relevant for analysis in IPS.

Modbus and IEC 61850 are NOT supported - IPS does not support direct connection to specific devices. Modbus and similar protocols are not an option for direct interfacing with IPS.

IPS®SYSTEMS integration platform consists of:

- IPS®SmartGridDI and
- IPS®WEB Services

2.1 IPS®SmartGridDI

IPS®SmartGridDI supports:

- **Data (read/import) pull from external systems**

  IPS®SmartGridDI can monitor data changes in external systems, monitor databases, files, web services etc.
- **Data (export) push to external systems**

IPS®SmartGridDI can push data from its own data store to external sources on schedule or condition basis. Every external source (data connection) from/to which IPS is reading or sending information to is separately licensed.

With IPS Web API it is possible to exchange data only one unidirectional way, from External System (source) to IPS®SYSTEMS (target), whereas, IPS®SmartGridDI is required to enable bidirectional data exchange.

If one data source is used to exchange both via IPS Web API and IPS®SmartGridDI, then this is treated as a single Client License.

2.2 IPS®WEB Services

IPS®WEB Services are based on OData RESTful API, and offer capabilities for:

- IPS database data browsing

  In this way any information from IPS®SYSTEMS database can be read by third party systems.
- IPS database data storing

  In this way any information triggered by third party systems can be stored in IPS.

  IPS is licensing this type of interface per IPS Web API (client) consumer. For example, if IPS is required to feed data from 3 different data sources (SAP, SCADA and GIS) then this would be 3 different client (license) subscribers.

2.2.1 IPS OData API usage

Below is given information about IPS OData API usage.

For testing purposes use LINQPad, Google Postman, Visual Studio or any other tool which can initiate http web request to OData service.

OData protocol defines one end point for exposing a single entity model that is serving. That endpoint is *<serviceRoot>/$metadata*, and the *HTTP GET* request on that endpoint is responding with CSDL document (*Common Schema Definition Language*) confirmative to the specification given at:

http://docs.oasis-open.org/odata/odata-csdl-xml/v4.01/odata-csdl-xml-v4.01.html.

Afore mentioned $metadata CSDL document should explain IPS®ENERGY OData Web API model. This in turn means that any running instance of our service is able to describe it self (its entity model) via simple *HTTP GET* request.

On the following link is IPS document:

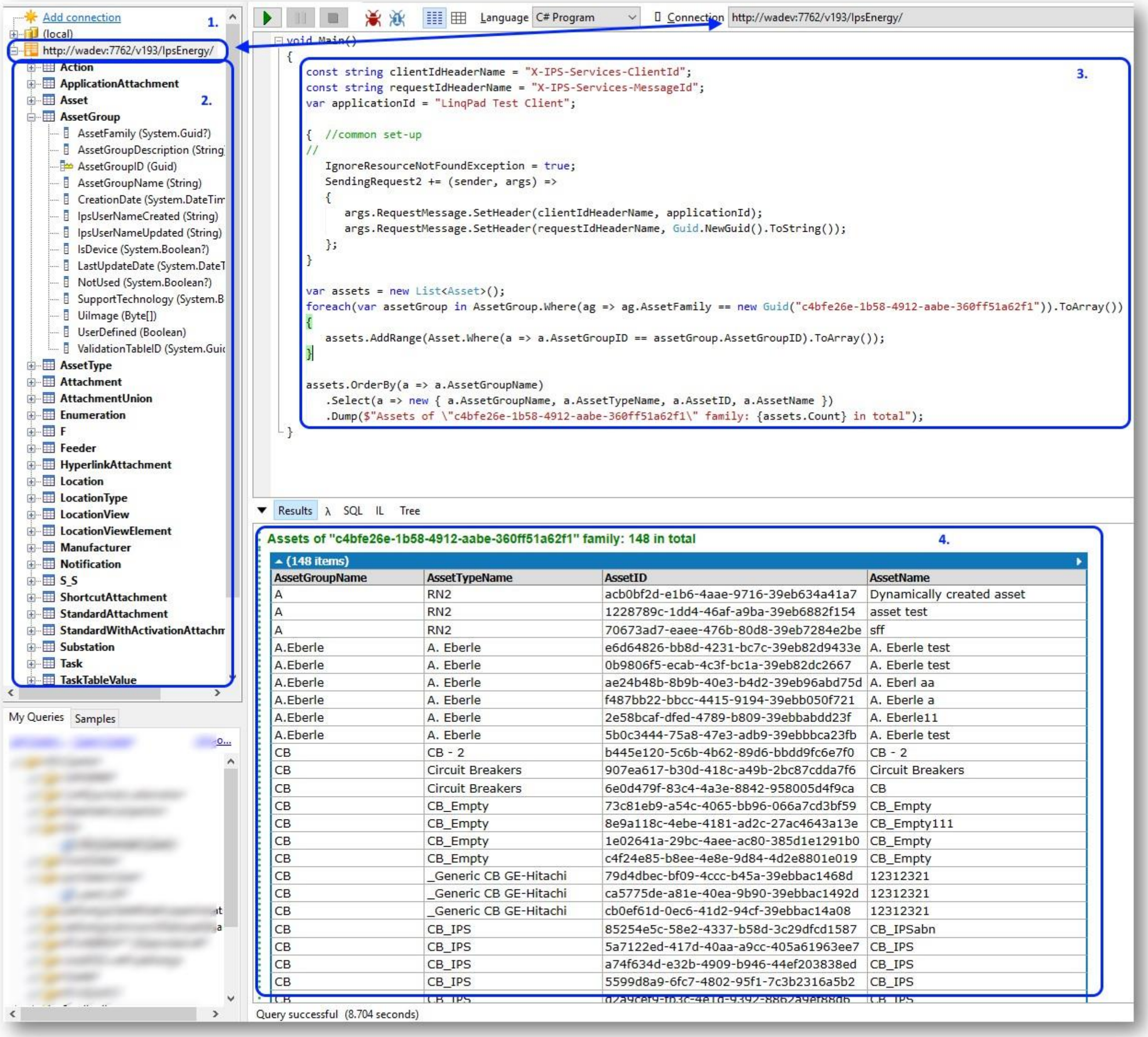
IpsEnergyOdataWebAPI_metadata_SCDL.xml

All the information needed for generating model in any programming/scripting language are present in that XML.

One of the examples of OData clients able to create entity tree out of such metadata documentation is popular (and free) tool:

LINQPad (https://www.linqpad.net/).

As an example, on the following figure is given LINQPad 5 screenshot from IPS test environment:



Connection: http://wadev:7762/v193/IpsEnergy/

```
void Main()
{
    const string clientIdHeaderName = "X-IPS-Services-ClientId";
    const string requestIdHeaderName = "X-IPS-Services-MessageId";
    var applicationId = "LinqPad Test Client";

    {  //common set-up
    //
       IgnoreResourceNotFoundException = true;
       SendingRequest2 += (sender, args) =>
       {
          args.RequestMessage.SetHeader(clientIdHeaderName, applicationId);
          args.RequestMessage.SetHeader(requestIdHeaderName, Guid.NewGuid().ToString());
       };
    }

    var assets = new List<Asset>();
    foreach(var assetGroup in AssetGroup.Where(ag => ag.AssetFamily == new Guid("c4bfe26e-1b58-4912-aabe-360ff51a62f1")).ToArray())
    {
       assets.AddRange(Asset.Where(a => a.AssetGroupID == assetGroup.AssetGroupID).ToArray());
    }

    assets.OrderBy(a => a.AssetGroupName)
       .Select(a => new { a.AssetGroupName, a.AssetTypeName, a.AssetID, a.AssetName })
       .Dump($"Assets of \"c4bfe26e-1b58-4912-aabe-360ff51a62f1\" family: {assets.Count} in total");
}
```

Assets of "c4bfe26e-1b58-4912-aabe-360ff51a62f1" family: 148 in total

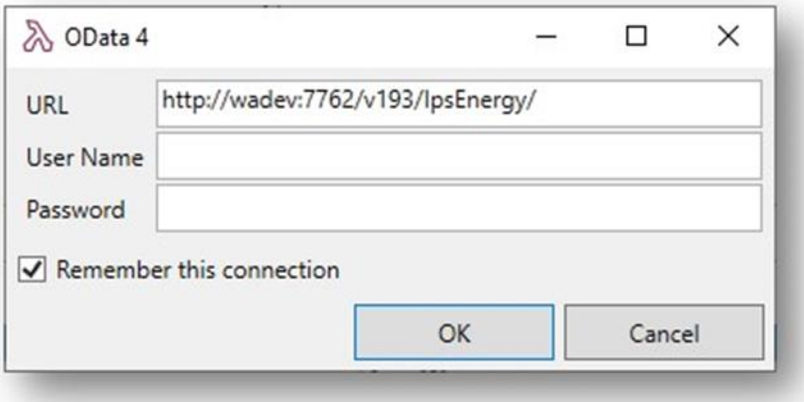
AssetGroupName	AssetTypeName	AssetID	AssetName
A	RN2	acb0bf2d-e1b6-4aae-9716-39eb634a41a7	Dynamically created asset
A	RN2	1228789c-1dd4-46af-a9ba-39eb6882f154	asset test
A	RN2	70673ad7-eaee-476b-80d8-39eb7284e2be	sff
A.Eberle	A. Eberle	e6d64826-bb8d-4231-bc7c-39eb82d9433e	A. Eberle test
A.Eberle	A. Eberle	0b9806f5-ecab-4c3f-bc1a-39eb82dc2667	A. Eberle test
A.Eberle	A. Eberle	ae24b48b-8b9b-40e3-b4d2-39eb96abd75d	A. Eberl aa
A.Eberle	A. Eberle	f487bb22-bbcc-4415-9194-39ebb050f721	A. Eberle a
A.Eberle	A. Eberle	2e58bcaf-dfed-4789-b809-39ebbabdd23f	A. Eberle11
A.Eberle	A. Eberle	5b0c3444-75a8-47e3-adb9-39ebbbca23fb	A. Eberle test
CB	CB - 2	b445e120-5c6b-4b62-89d6-bbdd9fc6e7f0	CB - 2
CB	Circuit Breakers	907ea617-b30d-418c-a49b-2bc87cdda7f6	Circuit Breakers
CB	Circuit Breakers	6e0d479f-83c4-4a3e-8842-958005d4f9ca	CB
CB	CB_Empty	73c81eb9-a54c-4065-bb96-066a7cd3bf59	CB_Empty
CB	CB_Empty	8e9a118c-4ebe-4181-ad2c-27ac4643a13e	CB_Empty111
CB	CB_Empty	1e02641a-29bc-4aee-ac80-385d1e1291b0	CB_Empty
CB	CB_Empty	c4f24e85-b8ee-4e8e-9d84-4d2e8801e019	CB_Empty
CB	_Generic CB GE-Hitachi	79d4dbec-bf09-4ccc-b45a-39ebbac1468d	12312321
CB	_Generic CB GE-Hitachi	ca5775de-a81e-40ea-9b90-39ebbac1492d	12312321
CB	_Generic CB GE-Hitachi	cb0ef61d-0ec6-41d2-94cf-39ebbac14a08	12312321
CB	CB_IPS	85254e5c-58e2-4337-b58d-3c29dfcd1587	CB_IPSabn
CB	CB_IPS	5a7122ed-417d-40aa-a9cc-405a61963ee7	CB_IPS
CB	CB_IPS	a74f634d-e32b-4909-b946-44ef203838ed	CB_IPS
CB	CB_IPS	5599d8a9-6fc7-4802-95f1-7c3b2316a5b2	CB_IPS

Query successful (8.704 seconds)

Figure 2: LINQPad

Marked regions are showing:

1. The URL of the service endpoint. This is the only configuration needed for this query to work (and classes to be generated).

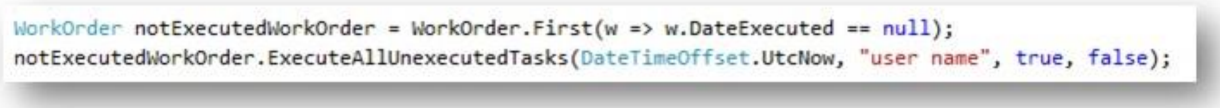


URL http://wadev:7762/v193/IpsEnergy/

Figure 3: URL of the service endpoint

2. Auto generated classes from $metadata document.
3. The C# code example.
4. Result set (displayed by LINQPad unique feature Dump ()).

In other languages there are similar tools. And for Visual studio there are plug-ins for proxy class code generation (VB or C#) from metadata. All exposed methods will also be available in the generated code. Like *ExecuteAllUnexecutedTasks*:



```
WorkOrder notExecutedWorkOrder = WorkOrder.First(w => w.DateExecuted == null);
notExecutedWorkOrder.ExecuteAllUnexecutedTasks(DateTimeOffset.UtcNow, "user name", true, false);
```

Figure 4: Generated code for *ExecuteAllUnexecutedTasks*

Before starting to write code it necessary to understand the mapping between IPS OData attributes and IPS®ENERGY Database schema. For that purpose, relevant is the documentation about IPS®ENERGY Database Model that you can find on the following link:

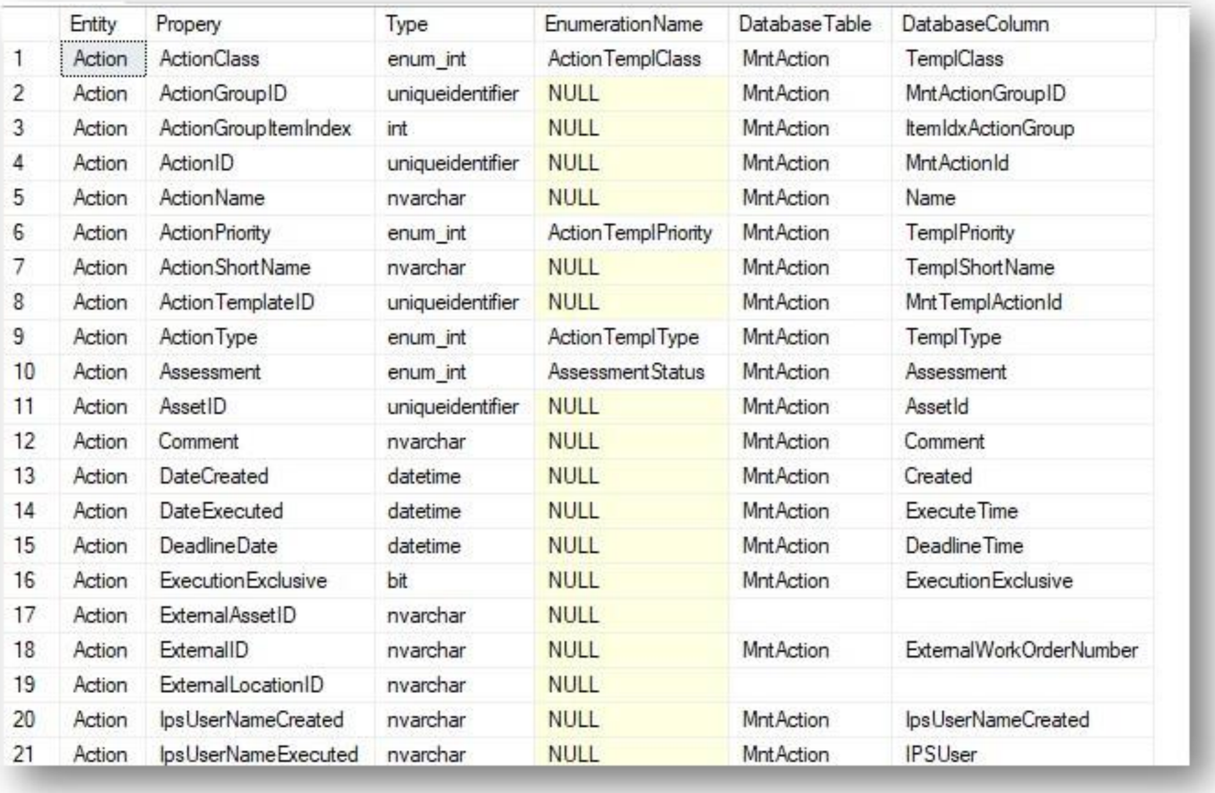
IPS®ENERGY Database Model

Information regarding IPS OData attributes is available in the *IpsEnergyServices_REST* database that is part of IPS delivery.

More specifically, following query will return all the mappings between the OData model and IPS®ENERGY Database:

```
SELECT
       MessageTable as Entity,
       Field as Propery,
       FieldValueType as Type,
       FieldEnumTypeName as EnumerationName,

       IpsEnergyTable as DatabaseTable,
       IpsEnergyColumn as DatabaseColumn

  FROM [DataExchangeMappingProperties]

  WHERE Disabled = 0 AND ApiVersion = 193

  ORDER BY TableType, MessageTable, Field
```

The result of the query:



	Entity	Propery	Type	EnumerationName	DatabaseTable	DatabaseColumn
1	Action	ActionClass	enum_int	ActionTemplClass	MntAction	TemplClass
2	Action	ActionGroupID	uniqueidentifier	NULL	MntAction	MntActionGroupID
3	Action	ActionGroupItemIndex	int	NULL	MntAction	ItemIdxActionGroup
4	Action	ActionID	uniqueidentifier	NULL	MntAction	MntActionId
5	Action	ActionName	nvarchar	NULL	MntAction	Name
6	Action	ActionPriority	enum_int	ActionTemplPriority	MntAction	TemplPriority
7	Action	ActionShortName	nvarchar	NULL	MntAction	TemplShortName
8	Action	ActionTemplateID	uniqueidentifier	NULL	MntAction	MntTemplActionId
9	Action	ActionType	enum_int	ActionTemplType	MntAction	TemplType
10	Action	Assessment	enum_int	AssessmentStatus	MntAction	Assessment
11	Action	AssetID	uniqueidentifier	NULL	MntAction	AssetId
12	Action	Comment	nvarchar	NULL	MntAction	Comment
13	Action	DateCreated	datetime	NULL	MntAction	Created
14	Action	DateExecuted	datetime	NULL	MntAction	ExecuteTime
15	Action	DeadlineDate	datetime	NULL	MntAction	DeadlineTime
16	Action	ExecutionExclusive	bit	NULL	MntAction	ExecutionExclusive
17	Action	ExternalAssetID	nvarchar	NULL		
18	Action	ExternalID	nvarchar	NULL	MntAction	ExternalWorkOrderNumber
19	Action	ExternalLocationID	nvarchar	NULL		
20	Action	IpsUserNameCreated	nvarchar	NULL	MntAction	IpsUserNameCreated
21	Action	IpsUserNameExecuted	nvarchar	NULL	MntAction	IPSUser

Figure 5: Mappings between the OData model and IPS®ENERGY Database

3 Interface Request (IPS IR)

IPS IR (Interface Request) is unidirectional data exchange for one data object from one system to another.

There are 3 different options for IR implementation:

- **Standard integration (most common)** - IPS or IPS partner configure only IPS®SYSTEMS components: IPS®ENERGY, IPS®SmartGridDI and IPS®WEB Services.
- **End to end integration** - IPS or IPS partner configure both IPS®SYSTEMS and External System integration components
- **Install only / own integration** – in this case IPS or IPS partner provides only software package required for integration

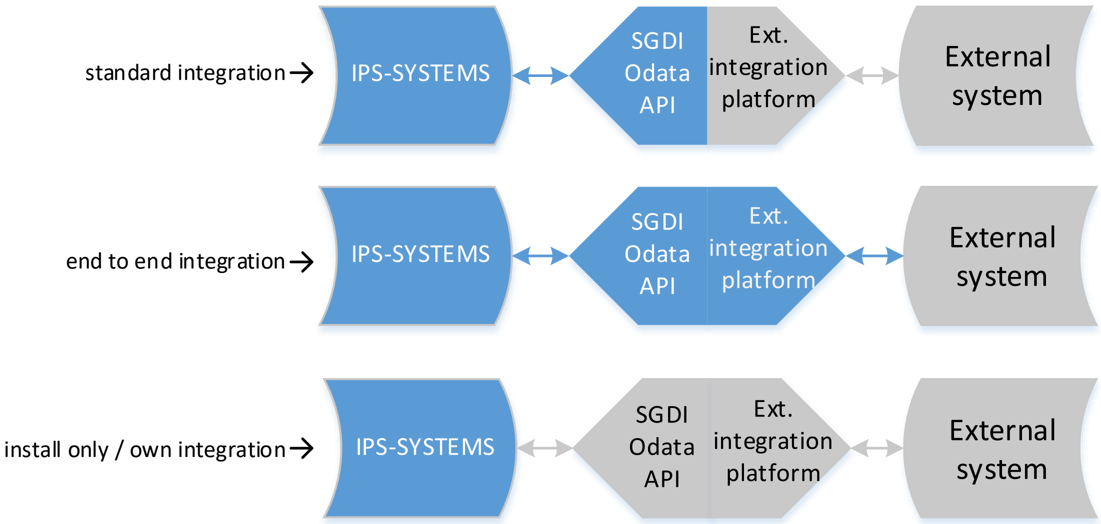


Figure 6: Three options for IR implementation

The number of interface requests (IRs), data objects, the types and technology are being defined in project design phase. In below chapter are given examples.

3.1 SAP-IPS example

In the example list given below are shown a couple of different interfaces and different possibilities of integrating. All the interface requests are described from both sides: source (outbound) and target (inbound).

- Equipment/Assets, Locations, and Work Order data objects and measurement documents are synced unidirectional from SAP to IPS.

Interface request 1: Sending list of newly created/updated/deleted Equipment/Assets

SAP -> IPS, using FTP as a middleware

Interface Request #	From (source system)	To (target system)	Data object	Type	Technology	Description
1	SAP ERP	FTP Server	Equipment	Outbound	SAP ABAP	Custom ABAP development for collecting and exporting corresponding SAP equipment data, creating a CSV file and placing it on the predefined location on the FTP
	FTP Server	IPS	Asset	Inbound	SGDI/IPS Web Services (OData WebAPIs)	SGDI trigger which is checking the FTP location, reading the CSV file, and calling the appropriate IPS OData API for create/delete/update of an asset

Table 1: SAP-IPS Interface Request -Example 1

ABAP (Advanced Business Application Programming) - SAP's proprietary programming language
SGDI - IPS®SmartGridDI

Interface request 2: Sending list of newly created/updated/deleted functional locations

SAP -> IPS, using FTP as a middleware

Interface Request #	From (source system)	To (target system)	Data object	Type	Technology	Description
2	SAP ERP	FTP Server	Location	Outbound	SAP ABAP	Custom ABAP development for collecting and exporting corresponding location data, creating a CSV file and placing it on the predefined location on the FTP
	FTP Server	IPS	Location	Inbound	SGDI/IPS Web Services (OData WebAPIs)	SGDI trigger which is checking the FTP location, reading the CSV file, and calling the appropriate IPS OData API for create/delete/update of locations or SQL stored procedure

Table 2: SAP-IPS Interface Request -Example 2

Interface request 3: Sending SAP PM work orders

SAP -> IPS, using FTP as a middleware

Interface Request #	From (source system)	To (target system)	Data object	Type	Technology	Description
3	SAP ERP	FTP Server	Work Orders	Outbound	SAP ABAP	Custom ABAP development for collecting and exporting executed PM work order based on a predefined criteria, creating a CSV file and placing it on the predefined location on the FTP
	FTP Server	IPS	Work Orders	Inbound	SGDI/IPS Web Services (OData WebAPIs)	SGDI trigger which is checking the FTP location, reading the CSV file, and calling the appropriate IPS OData API or SQL stored procedure

Table 3: SAP-IPS Interface Request -Example 3

Interface request 4: Sending list of measurement documents

SAP -> IPS, using SAP PI/PO as an enterprise service bus

Interface Request #	From (source system)	To (target system)	Data object	Type	Technology	Description
4	SAP ERP	SAP PI/PO	Measurement documents	Outbound	SAP ABAP, PROXY, JDBC	Custom ABAP development for collecting end exporting SAP measurement documents and storing that into temporary IPS SQL staging tables
	SAP PI/PO	IPS	Measurement documents	Inbound	SGDI	SGDI trigger which calls SQL stored procedure and process the staging tables

Table 4: SAP-IPS Interface Request -Example 4

- Notifications are synced bidirectional because some notifications are generated in SAP, and some in IPS.

Interface request 5: Sending SAP PM notifications

SAP -> IPS, using SAP PI/PO as an enterprise service bus

Interface Request #	From (source system)	To (target system)	Data object	Type	Technology	Description
5	SAP ERP	SAP PI/PO	Notifications	Outbound	SAP ABAP, PROXY, JDBC	Custom ABAP development for collecting end exporting SAP PM notification and storing them into temporary IPS SQL staging tables
	SAP PI/PO	IPS	Notifications	Inbound	SGDI	SGDI trigger which calls SQL stored procedure and process the staging tables

Table 5: SAP-IPS Interface Request -Example 5

Interface request 6: Sending IPS notifications

IPS -> SAP, using SAP PI/PO as an enterprise service bus

Interface Request #	From (source system)	To (target system)	Data object	Type	Technology	Description
6	IPS	SAP PI/PO	Notifications	Outbound	SGDI	SGDI trigger which calls export of IPS notification for health index and WO request for Oil Resample and calls SAP WS exposed through SAP PI/PO
	SAP PI/PO	SAP ERP	Notifications	Inbound	SAP ABAP, PROXY, SOAP	SAP ABAP custom development for creating notifications in SAP, using standard SAP BAPI function calls, exposed on SAP PI/PO as a web service

Table 6: IPS-SAP Interface Request -Example 6

3.2 Historian – IPS example

Interface Request #	From	To	Data object	Type	Technology	Description	
7	Historian	IPS	TransformerLoad	Inbound	SGDI / OData WebAPI	Import of AVG, MIN, MAX transformer load per week	SGDI Trigger which reads directly from SQL View and stores in IPS db

Table 7: Historian-IPS Interface Request example

3.3 Oil Laboratory – IPS example

Interface Request #	From	To	Data object	Type	Technology	Description	
8	Oil Laboratory	IPS	DGA	Inbound	SGDI / OData WebAPI	Import DGA analysis results	Web Service hosted for Oil laboratory stores received result in IPS database

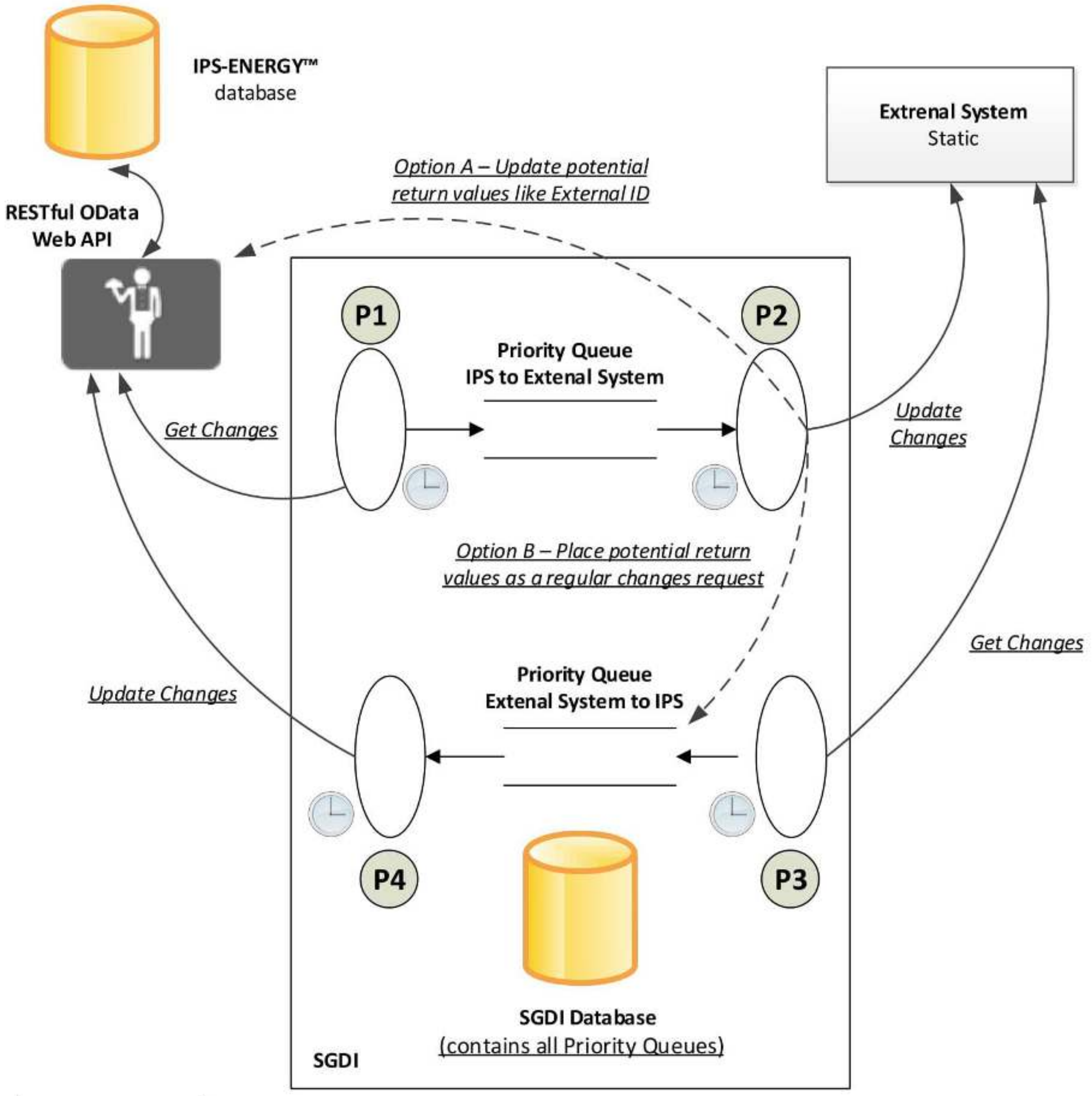
Table 8: Oil Laboratory -IPS Interface Request example

NOTE: Client is required to approve final list of interfaces before implementation.

Interfacing licenses are not including configuration services of the interfaces. Configuration services can be related to IPS®SYSTEMS Configuration OR to End-To-End Configuration services where IPS offers configuration of the third-party system as well.

IPS®WEB Service must be installed if any data exchange with IPS®SYSTEMS is required. IPS is always adopting interfacing technology related to requirements, performance and efficiency of implementation.

4 IPS®ENERGY System Integration - Scenarios



SCENARIO 1

All data exchange process in both directions are initiated by SGDI

Note: *The IPS-ENERGY RESTful OData Web API* ***never initiates*** *any exchange process*

Figure 7: IPS®ENERGY System Integration - Scenario 1

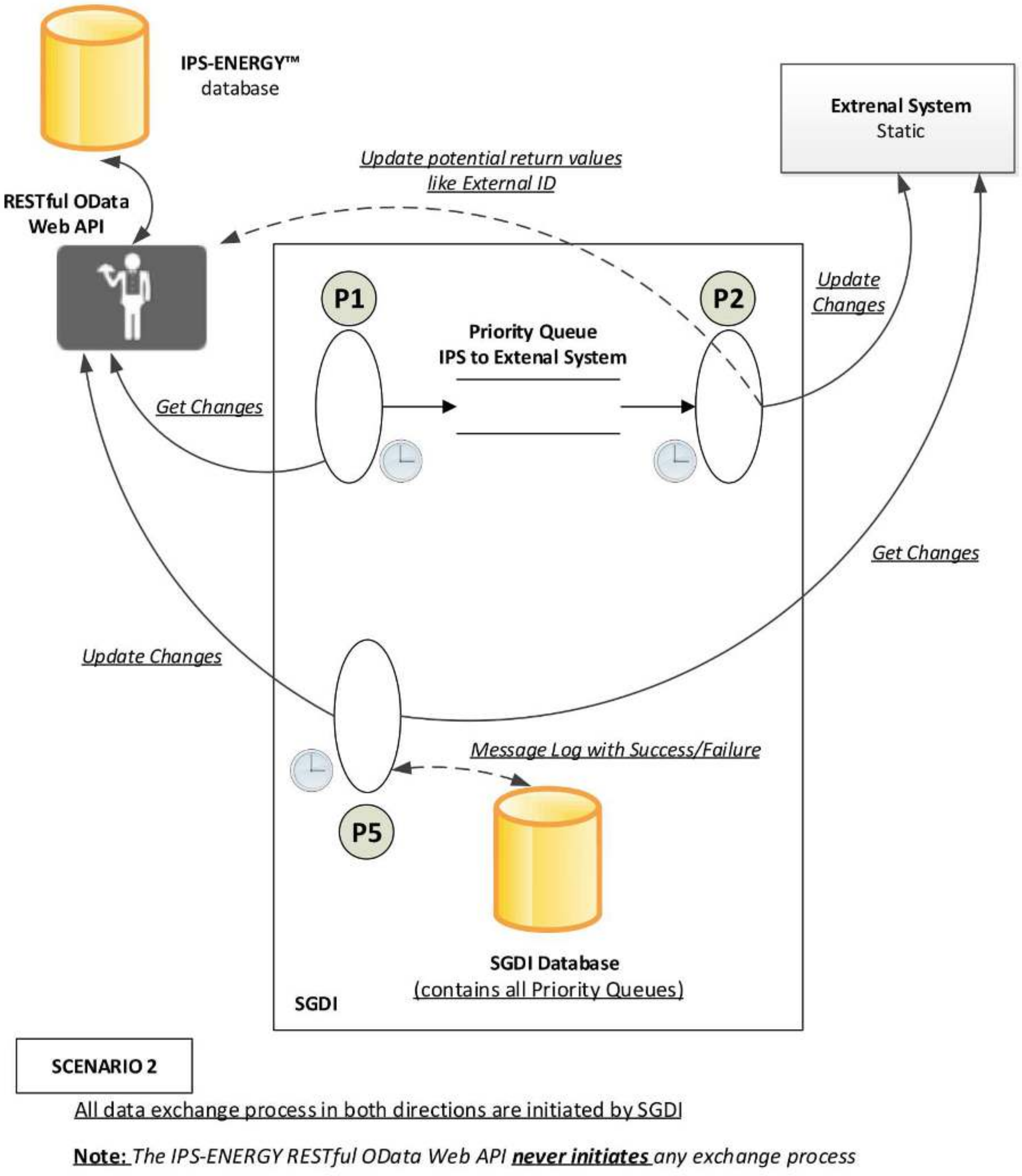
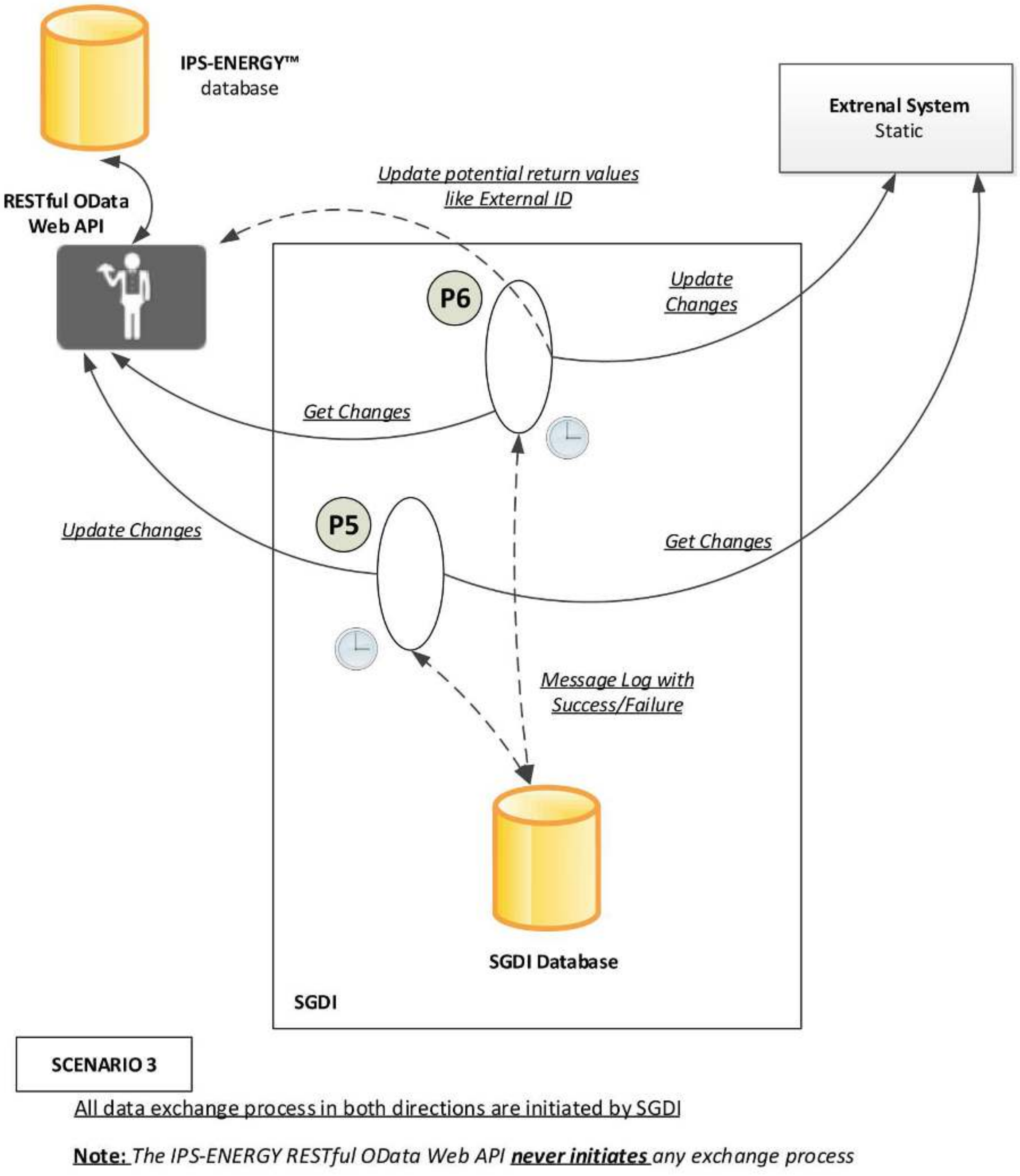

IPS-ENERGY™
database
Extrenal System
Static
Update potential return values
like External ID
RESTful OData
Web API
P1
P2
Priority Queue
IPS to Extenal System
Update
Changes
Get Changes
Get Changes
Update Changes
Message Log with Success/Failure
P5
SGDI Database
(contains all Priority Queues)
SGDI
SCENARIO 2
All data exchange process in both directions are initiated by SGDI
Note: The IPS-ENERGY RESTful OData Web API never initiates any exchange process

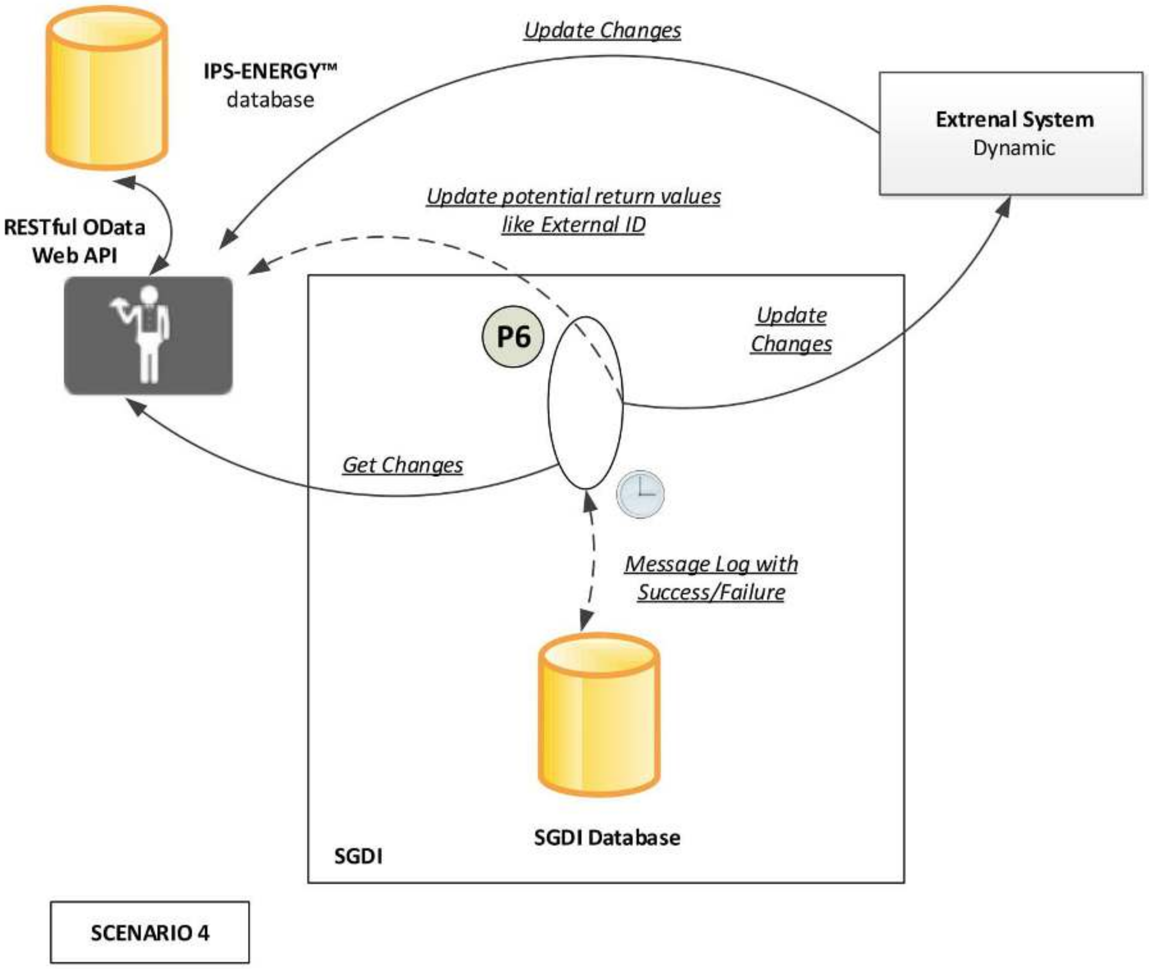
Figure 8: IPS®ENERGY System Integration - Scenario 2



All data exchange process in both directions are initiated by SGDI

Note: *The IPS-ENERGY RESTful OData Web API* ***never initiates*** *any exchange process*

Figure 9: IPS®ENERGY System Integration - Scenario 3



Data exchange process from External system to IPS is initiated by External system and from IPS to External system is initiated by SGDI

Note: *The IPS-ENERGY RESTful OData Web API **never initiates** any exchange process*

Figure 10: IPS®ENERGY System Integration - Scenario 4

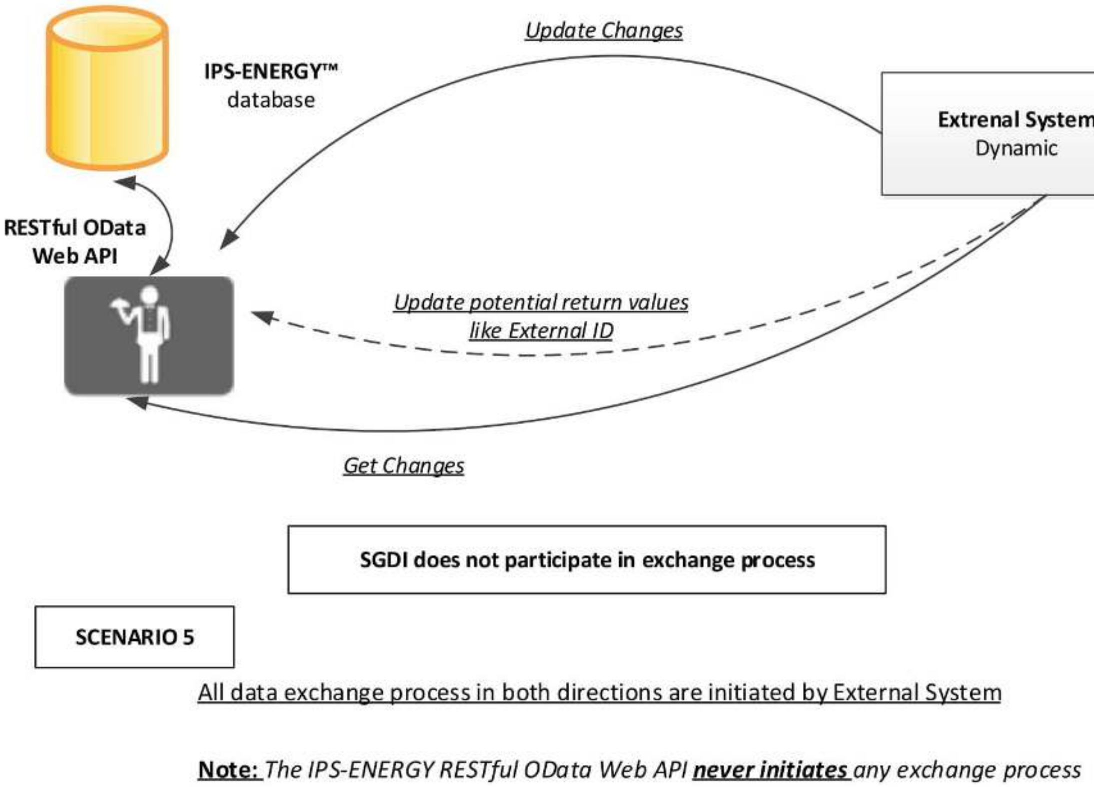

Update Changes
IPS-ENERGY™
database
Extrenal System
Dynamic
RESTful OData
Web API
Update potential return values
like External ID
Get Changes
SGDI does not participate in exchange process
SCENARIO 5
All data exchange process in both directions are initiated by External System
Note: The IPS-ENERGY RESTful OData Web API never initiates any exchange process

Figure 11: IPS®ENERGY System Integration - Scenario 5

In general, the preferred option for IPS would be to set up a web service on the IPS side that some external service can call to store measurements in IPS. If for that external service, SQL database table (via ODBC, for example) is fitting better, or perhaps file base exchange instead of web service, IPS can provide it as well.

If it is expected that IPS pulls the data from any source, instead of waiting for data to be pushed into IPS, we can also do any options mentioned above that would fit. IPS can pull the data via Web Service, API (Rest or SOAP), or pull from any readable file (xml, xls, csv), or pull directly from some table (SQL, Oracle, Access).

List of Figures

List of Tables

Appendix

Appendix A)

For information purposes, besides this document, the following information is also available:

- IPS Interfacing with Web Services
- IPS Enterprise Asset Management Integration with SAP on the following link:

  IPS®EAM_Integration_with_SAP_ENU_v1.3