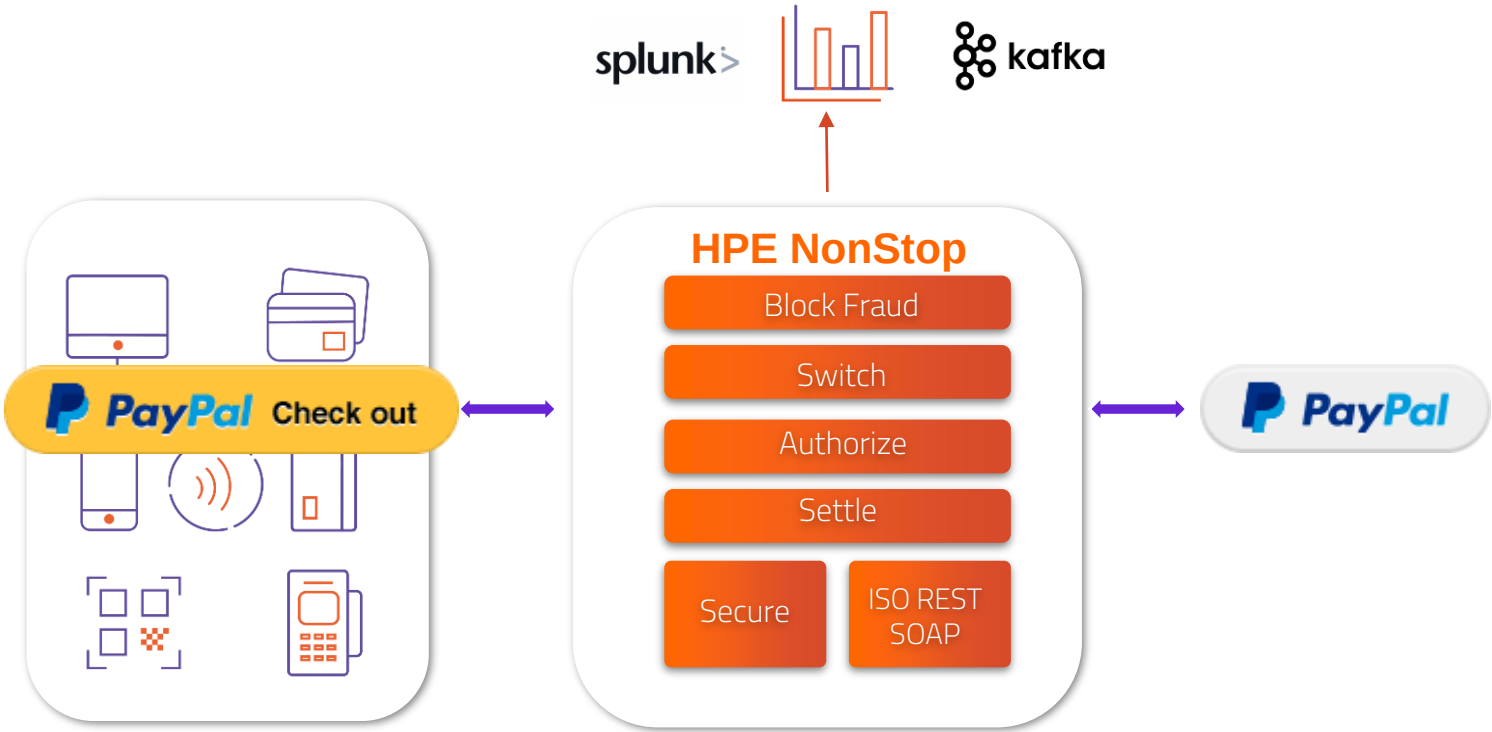

synchrony™
Connecting our NonStop to PayPal
Vikas Katoch, VP POS Services,
Synchrony Bank

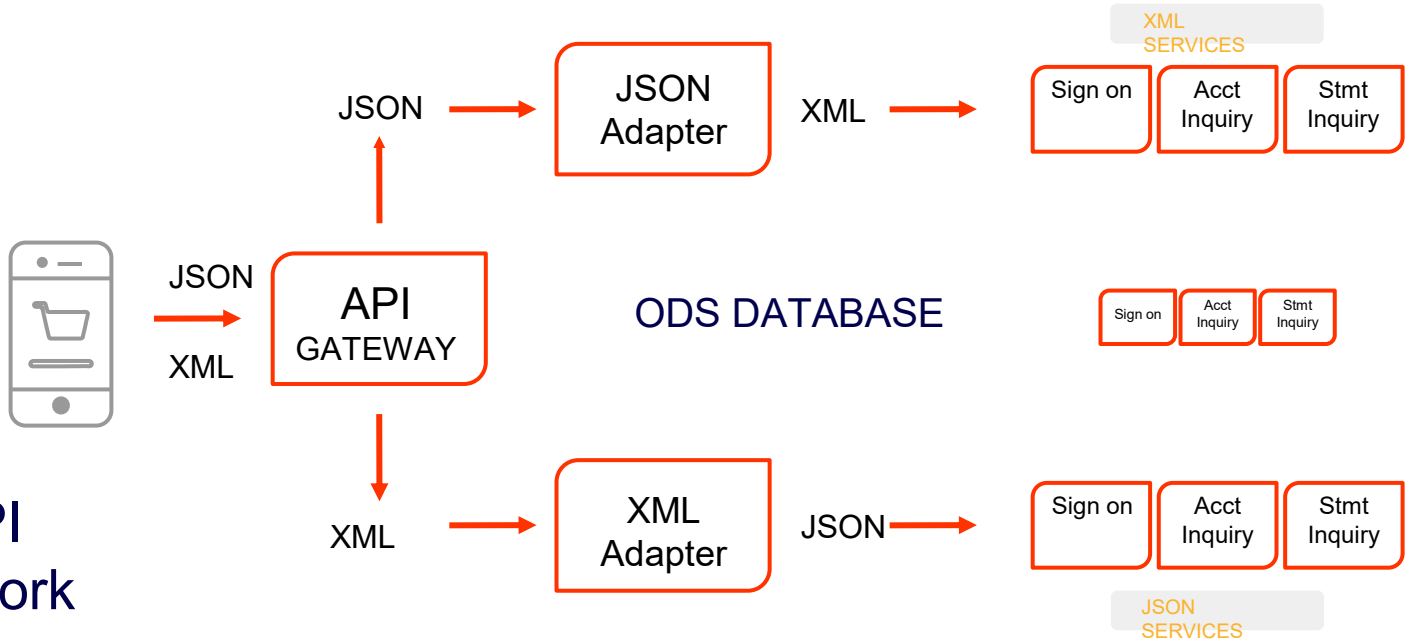
New PayPal Interface Overview

- The PayPal API Adaptor is used to connect our system to PayPal
- It is built using a framework for high availability and high performance
- The API adaptor translates JSON, XML and Custom formats
- It includes OAUTH2 + JWT for application security
- TLS is an optional module to provide additional security

New Solution Architecture

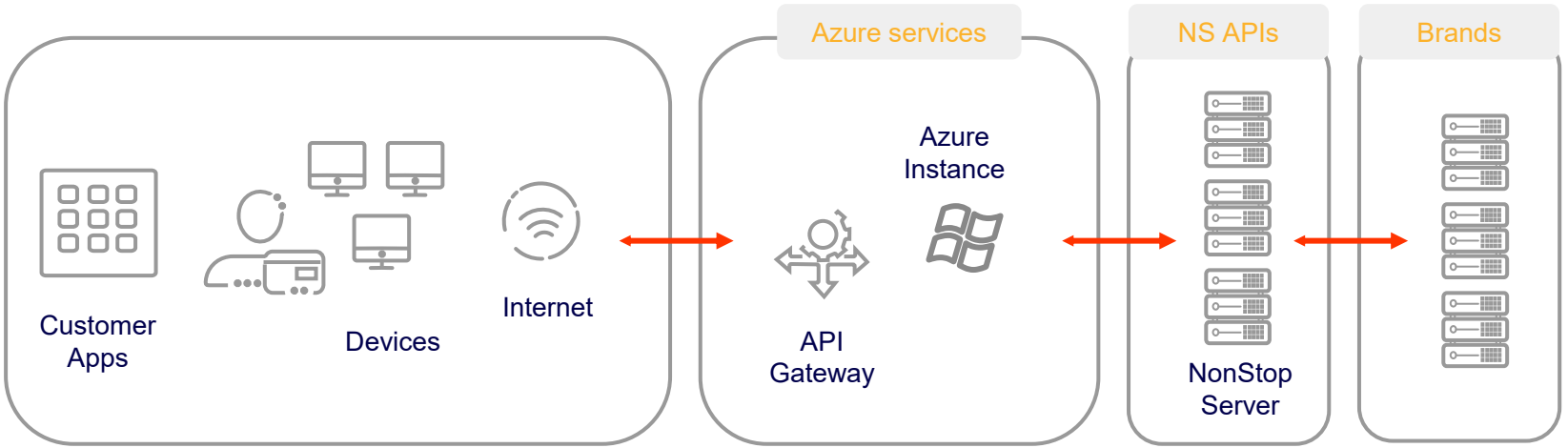


OpenAPI Framework

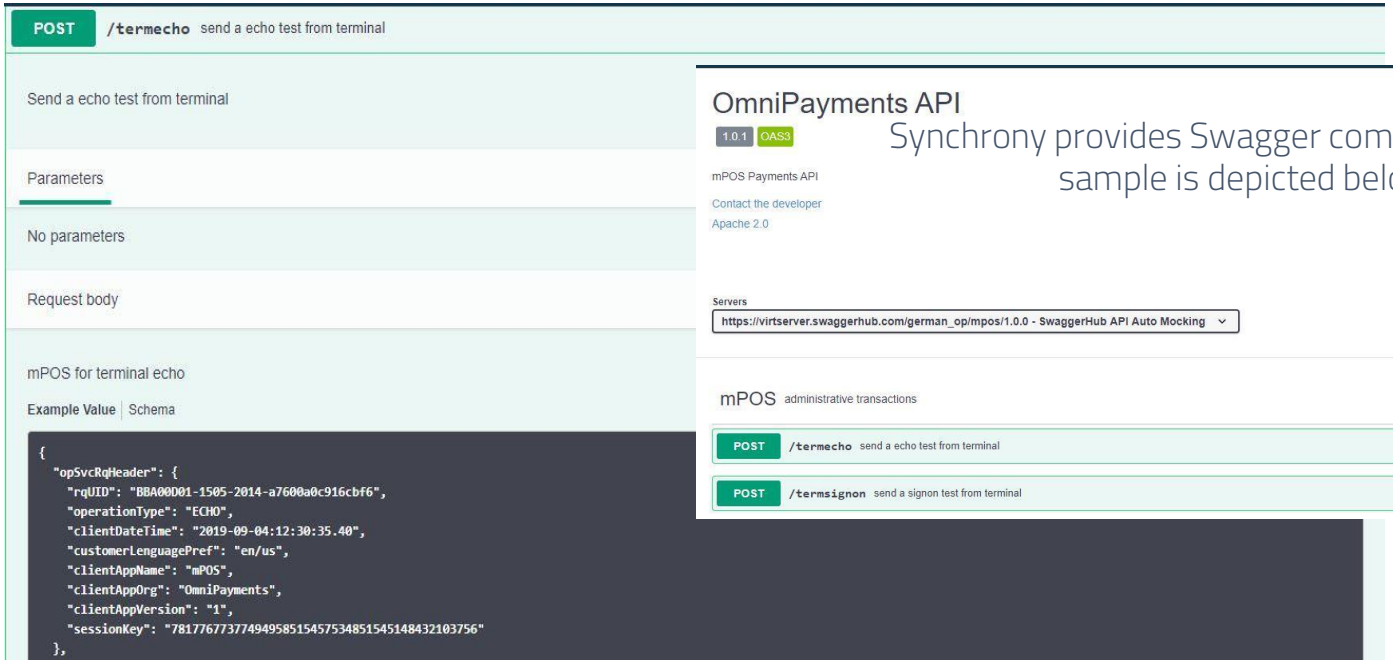


Public Cloud Interface

Client programs can communicate with Azure
Messages are sent to our NonStop and on to PayPal

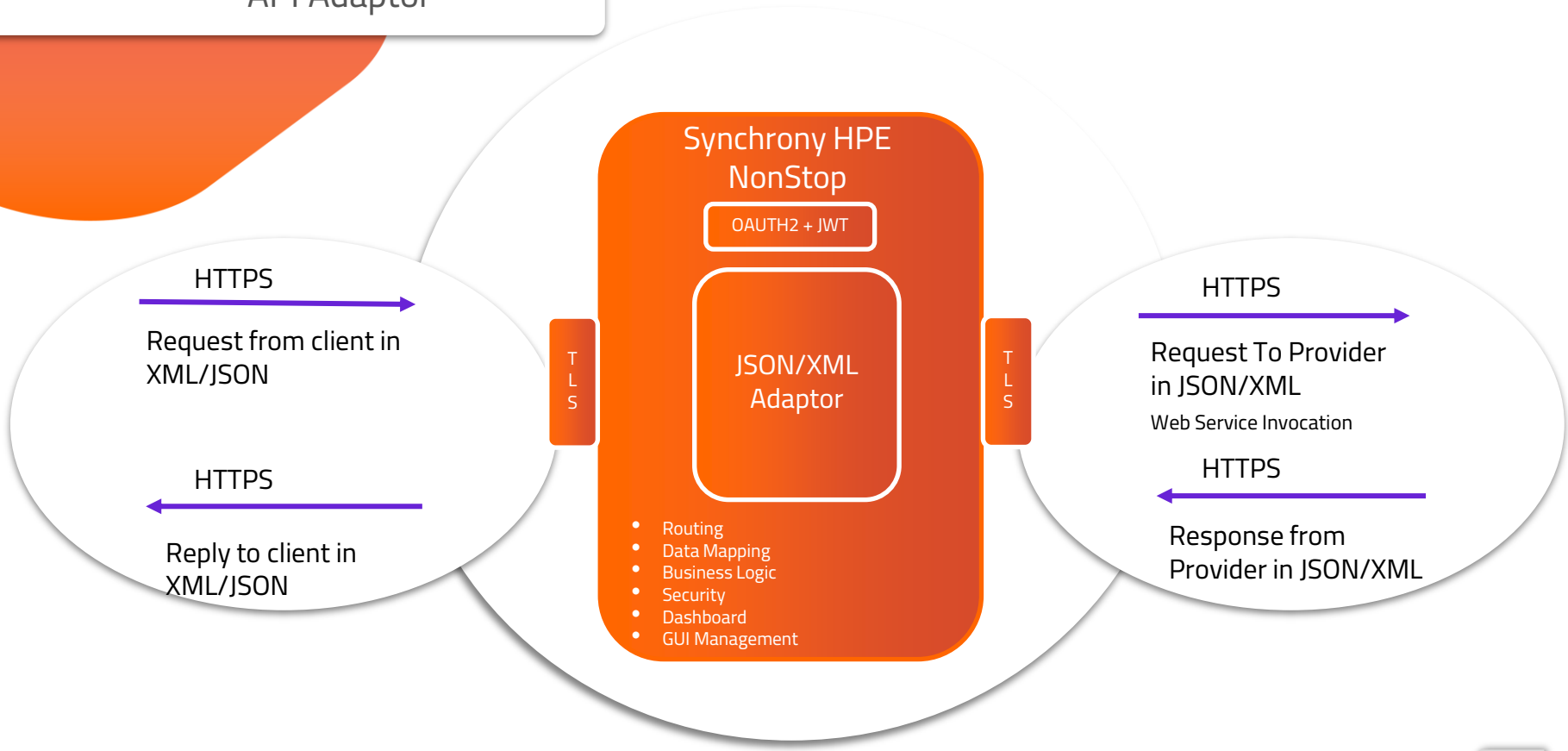


Swagger compliant APIs - sample



Synchrony provides Swagger compliant APIs – a sample is depicted below

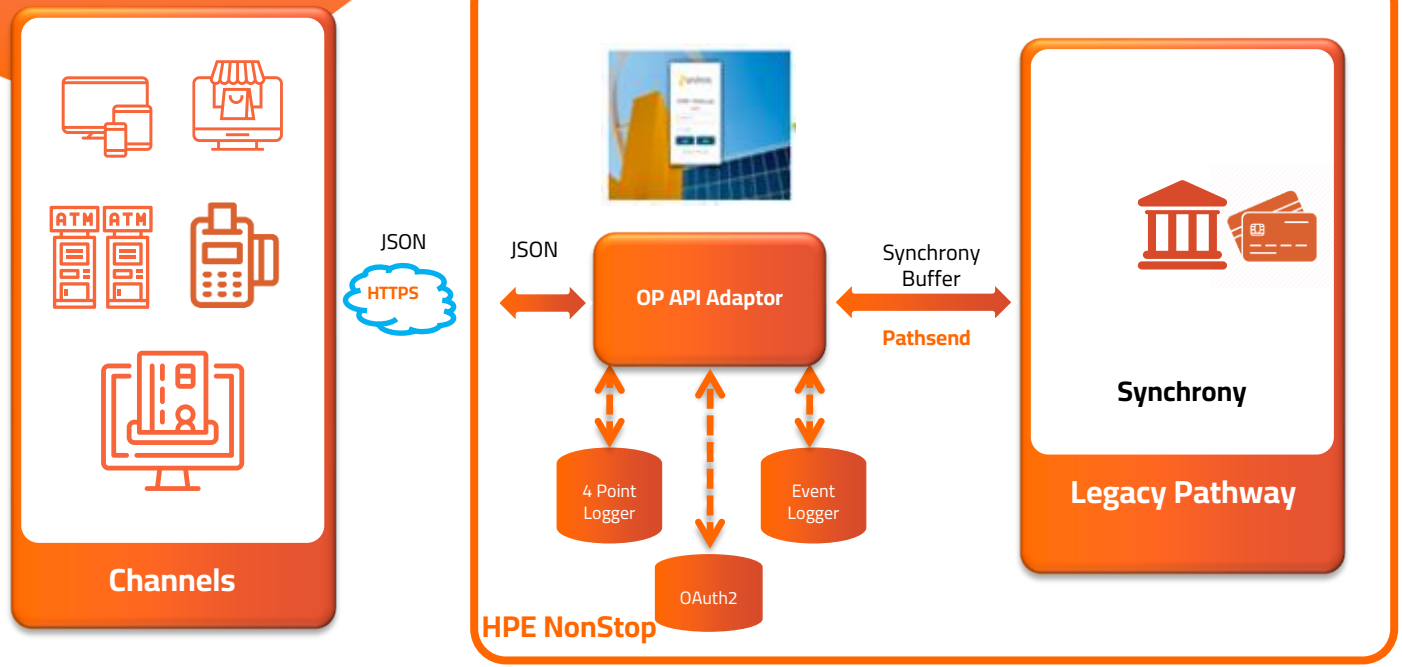
API Adaptor


HTTPS
Request from client in XML/JSON
HTTPS
Reply to client in XML/JSON
T L S
Synchrony HPE NonStop
OAUTH2 + JWT
JSON/XML Adaptor
Routing
Data Mapping
Business Logic
Security
Dashboard
GUI Management
T L S
HTTPS
Request To Provider in JSON/XML
Web Service Invocation
HTTPS
Response from Provider in JSON/XML

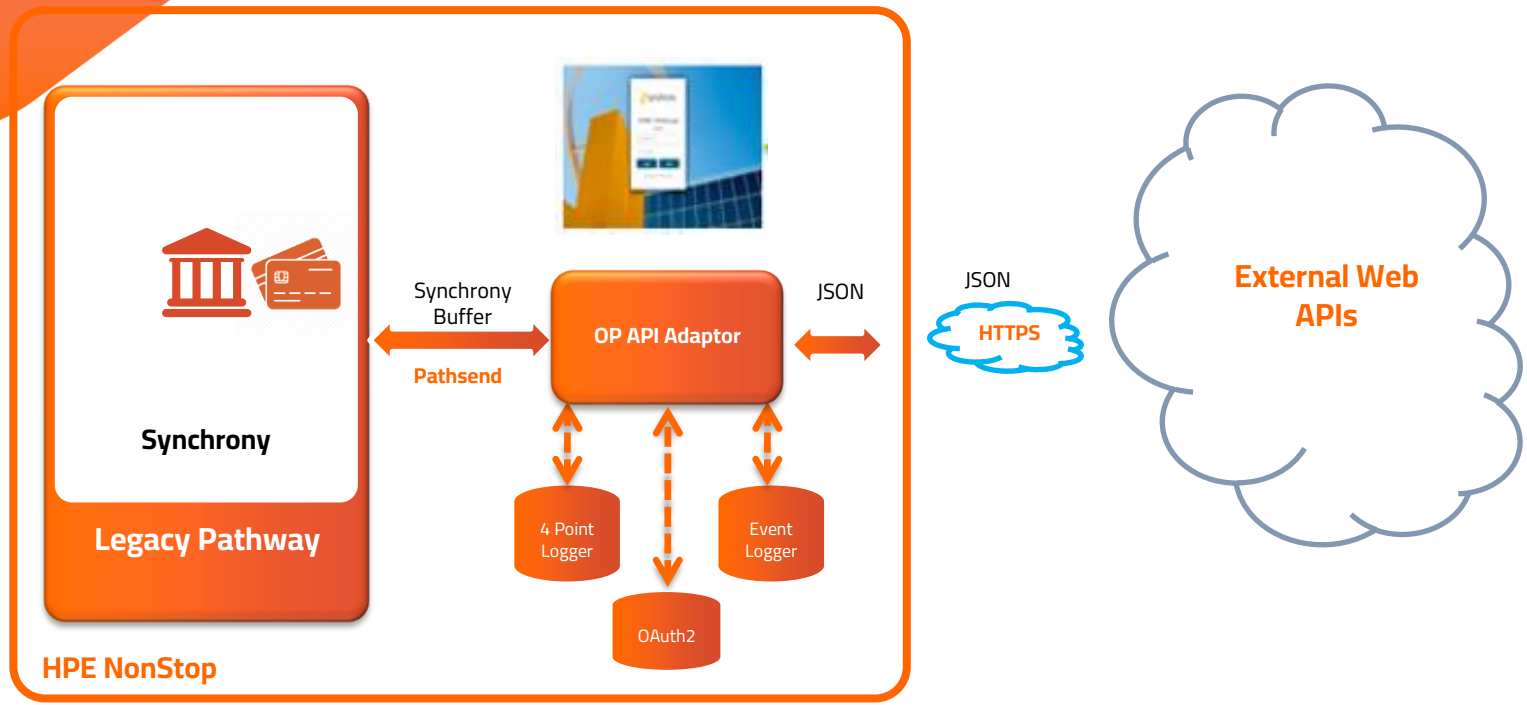
JWT - OAUTH2

- Synchrony supports JWT using OAUTH2
- The Client registers itself with Synchrony setting up a client ID and client secret (password)
- The Client requests an access token
- Synchrony validates the client credentials and issues the access token
- The client includes the access token in every request
- Synchrony processes the API request only if the access token is valid

Modernization - JSON to Legacy Pathway



Modernization - Legacy Pathway to JSON



Modernization – Synchrony Components

TLS - Optional

- Secure connection between client and server
- Data Protection
- Secure tunnel

API Gateway

- HTTP Header validations
- Logical routing based on API resource

JSON TD

- Request/Response mapping
- OAuth2 + JWT
- Data Transformation
- Business Logic
- JSON, XML and Custom mapping

Send XML Request to Provider in JSON

- Invoking API to convert XML request to JSON and send to provider

```
if(inXMLMesageLen >0 && inXMLMesage != NULL)
  {

/*
   Convert Trimmed BLM XML resoponse to JSON using libarary
   If success then apend to template otherwise send header only
*/

   xml2jsonError= trasnformXML2JSON(inXMLMesage,&xml2jsonBuffer,&xml2jsonBufferLen);

   //put service XML2JSON transformed message into json template on success
   //else return status as invalid blm response received.

      if(xml2jsonError !=0 || xml2jsonBufferLen <=0 || xml2jsonBuffer ==NULL)
        {
            if(xml2jsonBuffer)
              {
              free(xml2jsonBuffer);
              xml2jsonBuffer=NULL;
              }

            xml2jsonBufferLen=0;

            memset(inopSvcRsHdrOut.StatusCode,'\0',sizeof(inopSvcRsHdrOut.StatusCode));
            strcpy(inopSvcRsHdrOut.StatusCode,"800");

            memset(inopSvcRsHdrOut.Severity,'\0',sizeof(inopSvcRsHdrOut.StatusCode));
            strcpy(inopSvcRsHdrOut.Severity,"Error");

            memset(inopSvcRsHdrOut.StatusDesc,'\0',sizeof(inopSvcRsHdrOut.StatusCode));
            strcpy(inopSvcRsHdrOut.StatusDesc,"Invalid Response Received from Service");


        }
```

Reply to Client in XML

- Invoking API to get convert the JSON response to XML

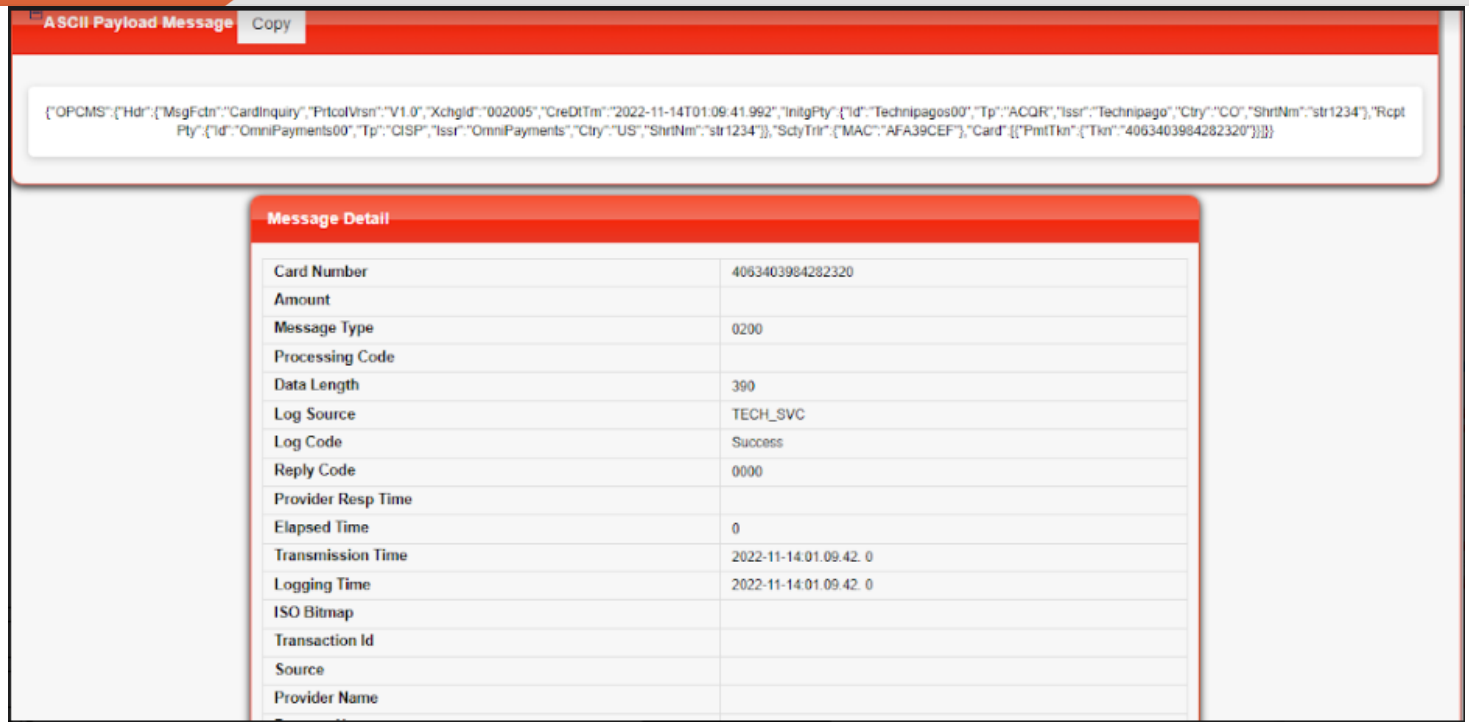
```
/* Call the JSON to XML LIbrary and provide incoming message to it
*/

displayCompleteBuffer("Input to transformJSON2XML : ", incomingAPIMessage);

//Incmoing JSON Message to JSON2XML Library
errReturn=transformJSON2XML(incomingAPIMessage, &outXMLMessage1);

if(errReturn !=0 || outXMLMessage1 ==NULL)
{
    memset(app_msg_txt, '\0', sizeof(app_msg_txt));
    sprintf(app_msg_txt, "Error in transformJSON2XML; Reply back to gateway, error code: %d", errReturn);
    log_emsmsg((int)LOG_CRIHIGH, (int)bizError, (int)LOGD_NOERROR, app_msg_txt);

    errReturn  = prepareErrorResponse(UE_Ctx->apiIndex,&UE_Ctx->mainopSvcRsHrd,&outXMLMessage2);

    ///there shud not be error from this function hard coded error response

    memset(currBuffPtr,'\0',bufflen);
    memcpy(currBuffPtr,outXMLMessage2,strlen(outXMLMessage2));
    *ioCount = strlen(outXMLMessage2);

    displayCompleteBuffer("processJSONRequest  output: ", currBuffPtr);

    OR_Ctx->operationCode = REPLY_CLIENT;
    return (int)BIZ_SEND_FAIL;
}

//displayCompleteBuffer("Output of transformJSON2XML : ", outXMLMessage1);

if(errReturn==0)
    displayCompleteBuffer("Converted XML Template after transformJSON2XML: ", outXMLMessage1);
```

GUI – Admin, Troubleshoot and Real time logs

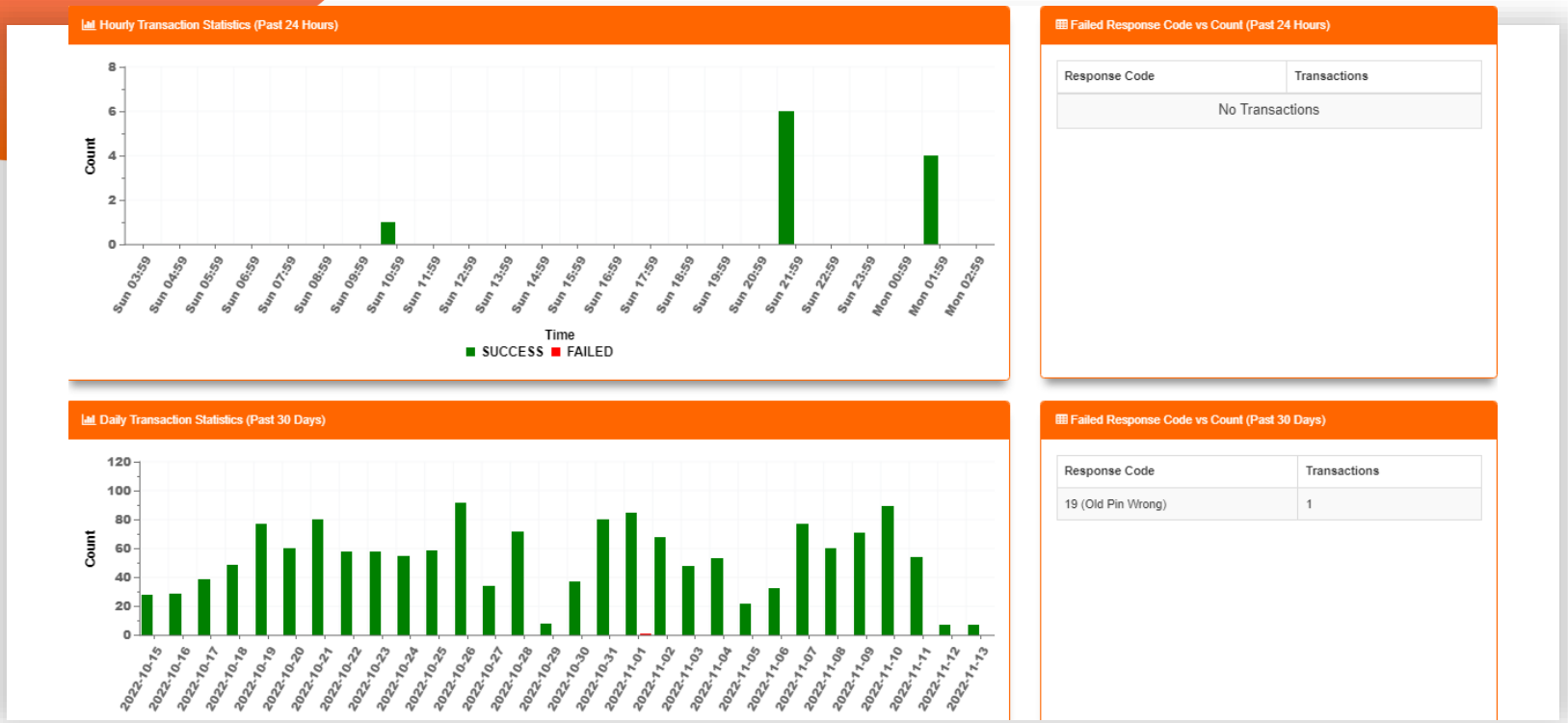
JSON 4 Point Transactions 51

Select	Tran Type	Tkn Num	Req/Rsp	Log Src	Prov Name	LCode	Reply Code	Card Form	Product ID	Log Time	Prov Resp(ms)	Total(ms)
○	CardInquiry	4063403984282320	Response	SVC_TECH	CARD-SRV1	√	0000	Virtual		2022-11-14:01.17.04.000351	5	5
○	CardInquiry	4063403984282320	Request	TECH_SVC		√	0000			2022-11-14:01.17.04.000346		0
○	CardInquiry	4063403984282320	Response	SVC_TECH	CARD-SRV1	√	0000	Virtual		2022-11-14:01.10.02.000768	4	4
○	CardInquiry	4063403984282320	Request	TECH_SVC		√	0000			2022-11-14:01.10.02.000764		0
○	CardInquiry	4063403984282320	Response	SVC_TECH	CARD-SRV1	√	0000	Virtual		2022-11-14:01.09.42.000004	4	4
○	CardInquiry	4063403984282320	Request	TECH_SVC		√	0000			2022-11-14:01.09.42.000000		0
○	GenCard	4063403984282320	Response	SVC_TECH	CARD-SRV1	√	0000	Virtual	000	2022-11-14:01.09.39.000367	208	208
○	GenCard		Request	TECH_SVC		√	0000	Virtual	000	2022-11-14:01.09.39.000160		0

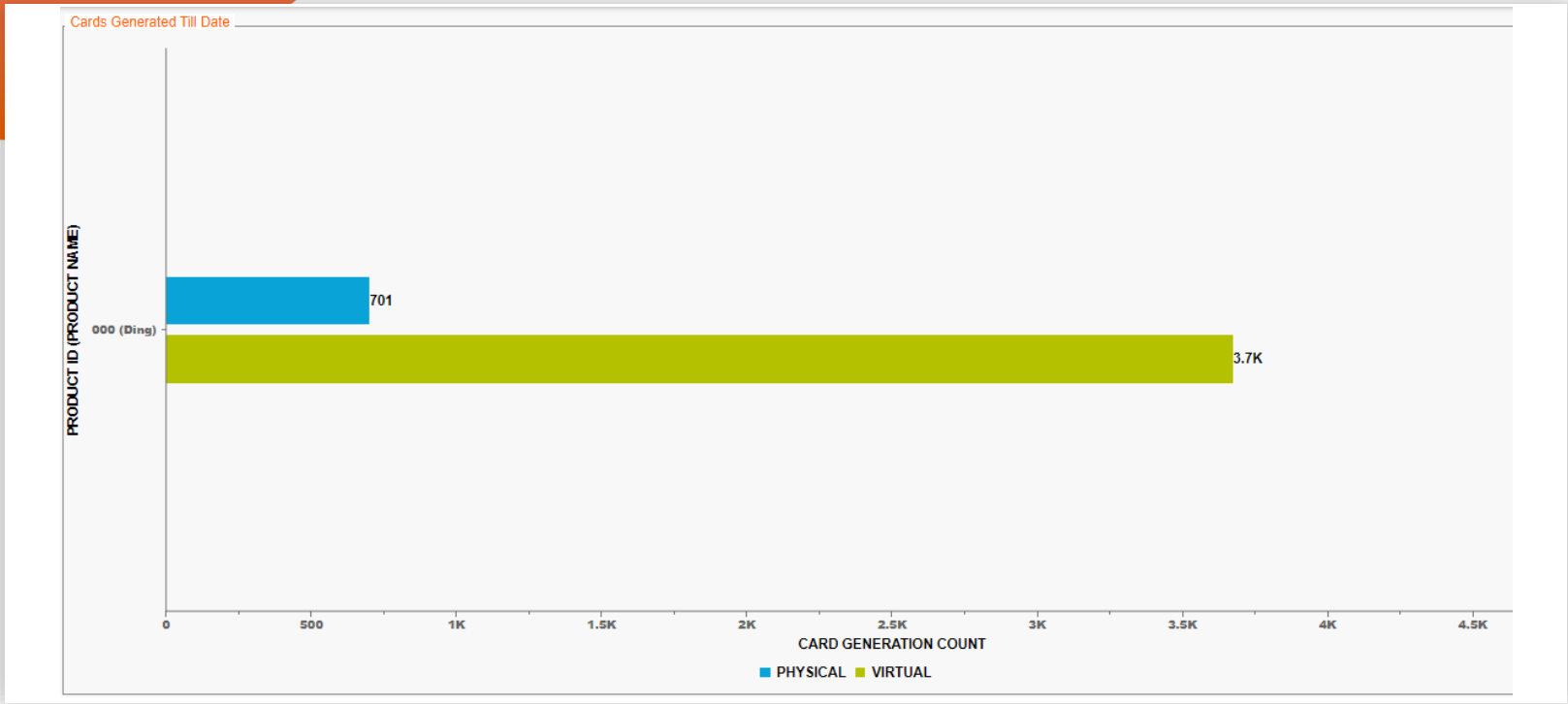
GUI – Payload - Tokenized



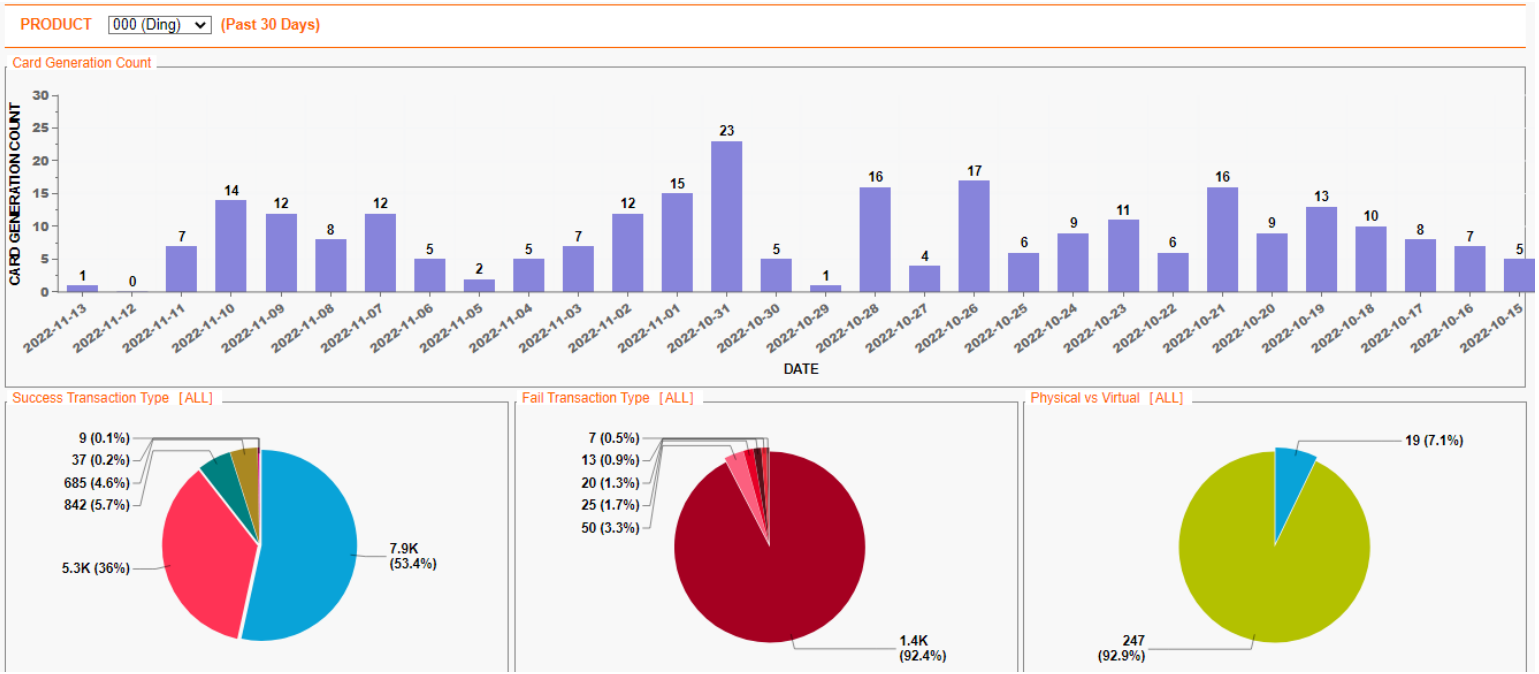
Dashboard – Realtime Monitoring on Linux – Optional



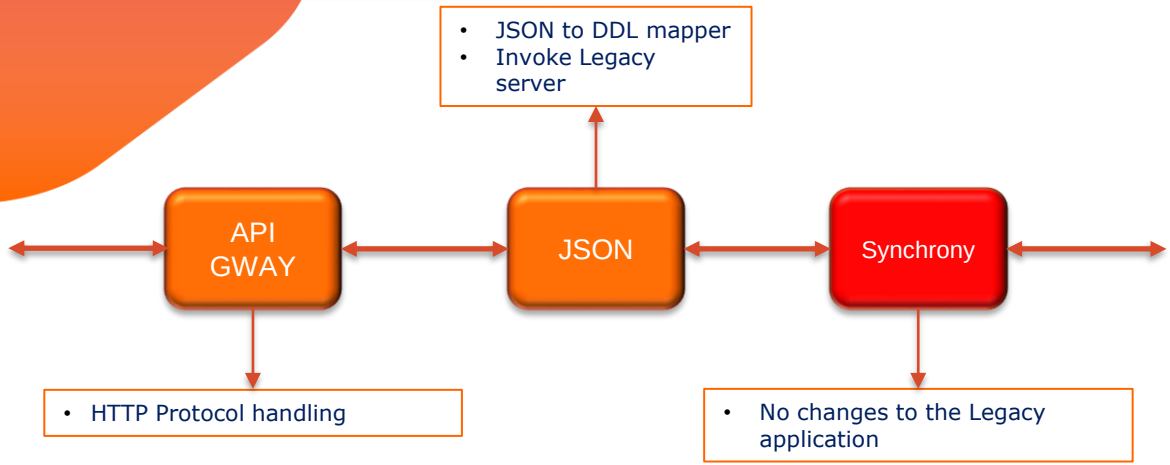
Dashboard – Realtime Monitoring



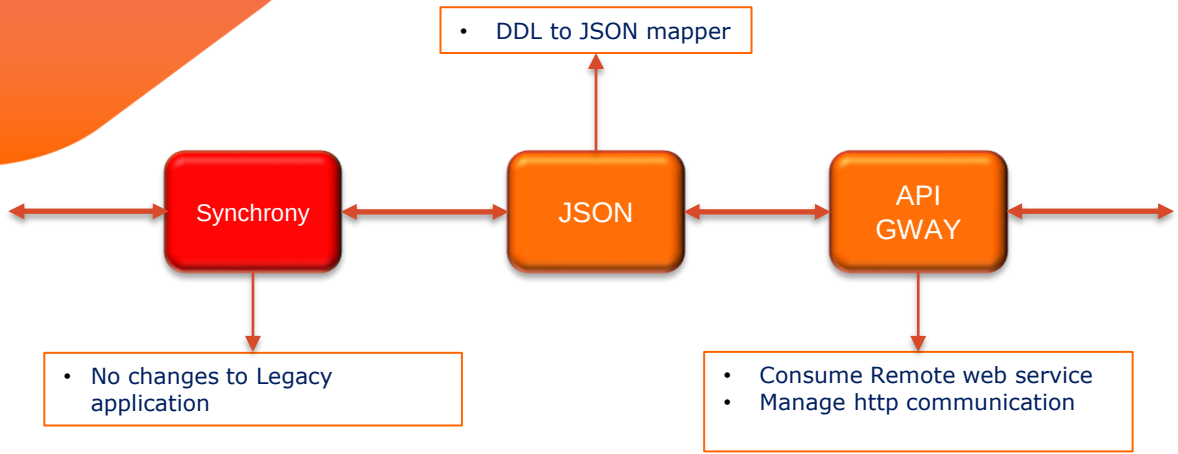
Dashboard – Realtime Monitoring – Statistics for the Interface



Message Flow: Inbound JSON



Message Flow: Outbound JSON



HTTP Response Codes

Response Code	Response code description
200	HTTP Service is OK
405	Invalid input
404	Input not found
500	Internal server error

- We support the common HTTP response codes
- Additional codes can be added

HTTP Response Codes

Common Response Codes from APIs

Response Code	Response code description
00	Operation Completed Successfully
01	Operation Declined
1024	Tag is missing
1120	Tag Value is Blank
2558	Institution Record is not Present
4	Invalid Request Received/Parse Error. Tag value cannot be blank
200	Tag value length Invalid
1020	Tag is Missing. Wrong operation type is received in the request

- These common API response codes are supported
- Additional response codes can be added

OAUTH2 Response Codes

OAUTH2 Response codes	
Response Code	Response code description
0	OAUTH OK
1	OAUTH Valid
2390	Parameter Absent
2391	Parameter Rejected
2392	Invalid Client
2393	Invalid User/Password
2394	Invalid Request
2395	Unauthorized Client / Authorization is not given in Request
2396	Server Error
2397	Token Expired
2398	Invalid Token
2399	Invalid Grant