# Modirum MPI Web Service

The solution is to use the Modirum low level Java API wrapped in a simple Java web application. The application have no database and all configuration must be supplied in the method calls or statically in the application.

Requirements (in no particular order):

1. Operations supported:
   a. Check Enrollment
   b. Verify Authentication Result
2. Operations are stateless – client systems have to store any auth proof, handle some state information.
3. Input / Output format
4. Supported deployment options.
5. Configuration of DS and Acquirer certificates.
6. Application configuration in web.xml, environment variables or java VM parameters.
7. Logging.
8. Versioning.

## Terms used in this document

| | |
|---|---|
| **ACS** | Access Control Server, a service at the Issuer that authenticates its cardholders. |
| **CAVV** | Card Authentication Verification Value; encrypted result from the authentication process at the ACS. Is sent in authorization messages and can be verified by the Issuer. |
| **DS** | Directory Server, a service operated by the card schemes (Visa, MC, JCB, Amex, Diners). Used to check enrollment status on a specific card. |
| **ECI** | Electronic Commerce Indicator; used in authorization requests to indicate the type of authentication used. |
| **MPI** | Merchant Plug-In; service used by merchant/service provider to communicate with DS and ACS. The MPI is also responsible for verifying the resulting PARes. |
| **PAReq** | Payer Authentication Request; request to authenticate the cardholder, sent from the MPI to the ACS via the cardholder's device. |
| **PARes** | Payer Authentication Response; response from the ACS to the MPI containing the result of the authentication process. Sent via the cardholder's device. |
| **VEReq** | Verify Enrollment Request; request sent by the MPI to the DS to check enrollment status of a specific card. |
| **VERes** | Verify Enrollment Response; response from the DS to the MPI with the enrollment status for a specific card. |
| **XID** | Transaction identifier that will follow the transaction throughout the 3-D Secure flow from VEReq to authorization. |

## Supported Operations

The application implements three operations; one for checking enrollment status by contacting a DS, one for verifying the result from the authentication process at the ACS and a last one for validating merchant configuration.

### Check Enrollment Operation

This operation takes input about the card and device, a DS is called and the result is returned to the client.
The following URL should be called.
https://<host>/mpi/check_enrollment_status

***MDCoreInterface.verifyEnrollment***(…) is used to call the DS to verify the enrollment status of a card. It takes card, device and merchant information as well as information about what DS to use in the form of an SSLDirectory object.
MPI will select SSLDirectory object by **AcquirerBin** parameter.

The Modirum API creates the VEReq and sends it to the selected DS. The DS will respond with a VERes. The API interprets this response and returns a response object. If the card is enrolled the API will return enrollment status, ACS Url and PAReq. In addition to this the VEReq, VERes and DS URL will also be returned for audit purposes.

### Verify Authentication Result Operation

This operation takes the result from the authentication process at the ACS (PARes) and PaRes validation object as input. PaRes validation object should contain PaReq data like acquirerBin, xid, currencyCode, … etc. so the MPI can match PaReq with PaRes sent. The MPI application uses Modirum's API to unpack and verify the result. The result of the authentication is returned to the client.

The following URL should be called.
https://<host>/mpi/verify_auhtentication_result
MPI will call API methods: ***MDCoreInterface. isPaReqMatchesPaRes***(…) and ***MDCoreInterface. processPARes***(…).

To be able to verify the result the MPI requires acquirer root certificates. The certificates are stored on MPI server file system in standard X.509 (.der or .pem) files. They will be selected by MPI using AcquirerBin that is parsed from PaRes. The MPI returns the result to the client as an authentication result, as well as additional information like ECI, CAVV etc.

### Check Merchant Config

This method is a convenience method for client systems to test if merchant credentials are ok. It should take DS login (and password) as well as AcquirerBin.

The following URL should be called.
https://<host>/mpi/check_merchant_config

MPI tests the credentials on Visa and MC directory servers using predefined test cards.

## Failover and timeouts

MPI can handle the timeout situation when retrieving data from DS (SSLDirectory.timeout). The DS timeout is configurable via DS config XML file referenced from environment variable
`java:comp/env/dsConfigFile=/path/to/dsconfig.xml`
or java VM variable
`-DdsConfigFile=/path/to/dsconfig.xml`

MPI can also fail over automatically to the backup DS URL if available, when a timeout or error occurs. In this case DS URL-s should be configured comma-separated.

## Stateless Operations

Both check enrollment and verify result operations are stateless. No information is stored in the MPI between the calls. This allows the service to scale easily because any instance can process any request.

Since the MPI application is completely stateless, any and all information needed in a dispute situation has to be retained by the client system. The MPI logs all operations to a log file, but the client system has to be responsible for keeping the needed information in an accessible manner. This includes the complete VEReq, VERes, PAReq and PARes messages.

## Input / output format

MPI handles XML input/output messages over HTTPS using JAXB library.
**https://<host>/mpi/check_enrollment_status sample input XML:**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CheckEnrollmentStatusRequest>
  <merchantInfo>
    <merchantName>CoffeeHouse demo shop WS</merchantName>
    <merchantCountryCode>246</merchantCountryCode>
    <merchantUrl>http://www.modirum.com</merchantUrl>
    <dsLoginConfig>
      <acquirerBin>444444</acquirerBin>
      <dsLogin>123456</dsLogin>
      <dsPassword>passwd</dsPassword>
    </dsLoginConfig>
  </merchantInfo>
  <purchaseInfo>
    <amount>1100</amount>
    <amountExponent>2</amountExponent>
    <cardNumber>40160000000002</cardNumber>
    <expMonth>12</expMonth>
    <expYear>99</expYear>
    <currencyCodeNumerical>840</currencyCodeNumerical>
    <deviceCategory>0</deviceCategory>

<httpAcceptHeader>text/html,application/xhtml+xml,application/xml;q=0.9,image/
webp,*/*;q=0.8</httpAcceptHeader>

<httpUserAgent>Mozilla5MacintoshIntelMacOSX192AppleWebKit53736KHTMLlikeGeckoCh
rome33175152Safari53736</httpUserAgent>
    <purchaseDescription>DVD Movies</purchaseDescription>
  </purchaseInfo>
</CheckEnrollmentStatusRequest>
```

**https://<host>/mpi/check_enrollment_status sample output XML:**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CheckEnrollmentStatusResponse>
  <xid>EPnrhfJdThHonnFuxmSapwklsoA=</xid>
  <enrollmentStatus>Y</enrollmentStatus>
  <errorCode></errorCode>
  <errorMessage></errorMessage>
  <acsUrl>https://acs.modirum.com/mdpayacs/pareq</acsUrl>
  <formPaReq>CompressedUsingZLibDeflateAndBase64EncodedPAReq</formPaReq>
  <log>
    <timestamp>2014-04-08T14:23:46Z</timestamp>
    <dsUrl>https://acs.modirum.com/mdpayacs/vereq</dsUrl>
    <veReq>Base64EncodedVEReq</veReq>
    <veRes>Base64EncodedVERes</veRes>
    <paReq>Base64EncodedPAReq</paReq>
  </log>
</CheckEnrollmentStatusResponse>
```

**https://<host>/mpi/verify_auhtentication_result sample input XML:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VerifyAuthenticationResultRequest>
  <paRes>CompressedUsingZLibDeflateAndBase64EncodedPARes</paRes>
  <validationInfo>
      <acquirerBin>444444</acquirerBin>
      <dsLogin>123456</dsLogin>
      <xid>EPnrhfJdThHonnFuxmSapwklsoA=</xid>
      <currencyCodeNumerical>872</currencyCodeNumerical>
      <cardNumberSuffix>0004</cardNumberSuffix>
  </validationInfo>
</VerifyAuthenticationResultRequest>
```

**https://<host>/mpi/verify_auhtentication_result sample output XML:**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VerifyAuthenticationResultResponse>
  <paResValidationResult>PARES_VALID</paResValidationResult>
  <authenticationResult>Y</authenticationResult>
  <errorCode></errorCode>
  <errorMessage>Y status</errorMessage>
  <eci>05</eci>
  <cavv>AAABBJRGEAAAAABZFEYQAAAAAAA=</cavv>
  <cavvAlgorithm>2</cavvAlgorithm>
  <log>
    <timestamp>2014-04-08T14:23:48Z</timestamp>
    <paRes>Base64EncodedPARes</paRes>
  </log>
</VerifyAuthenticationResultResponse>
```

**https://<host>/mpi/check_merchant_config sample input XML:**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CheckMerchantConfigRequest>
  <config>
    <acquirerBin>654321</acquirerBin>
    <dsLogin>123456</dsLogin>
    <dsPassword>passwd</dsPassword>
  </config>
</CheckMerchantConfigRequest>
```

**https://<host>/mpi/check_merchant_config sample output XML:**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CheckMerchantConfigResponse>
  <status>FAILED</status>
  <log>
    <timestamp>2014-04-08T14:14:44Z</timestamp>
    <dsUrl>https://acs.modirum.com/mdpayacs/vereq</dsUrl>
    <veReq>Base64EncodedVEReq</veReq>
    <veRes></veRes>
  </log>
  <errorMessage>ERROR message: Merchant not participating</errorMessage>
</CheckMerchantConfigResponse>
```

## Supported deployment options

The MPI application is distributed as a simple war. There are dependencies to keystore files in the file system, DS config xml file and MPI license file (MPI-license.txt). Other than that everything will be contained in the war.

It is possible to deploy war in a standard servlet >= 2.4 container. Specifically Tomcat >= 5.5 and JBoss >= 4.2.3 is supported.

The deployment of the service depends on how the client system is structured. It is possible to deploy the service in a group of dedicated servers behind a load balancer to achieve high availability. Alternatively the service can be deployed on each frontend machine to service the local host only.

## Configuration of DS and Acquirer certificates

To communicate with Directory Servers, the MPI needs access to trusted root certificates that can verify the identity of DS endpoints. If the required certificates are not in the standard cacerts truststore in Java, they have to be either added to cacerts or made available in a custom truststore to the application.

Acquirer specific certificates used to verify the authentication result can be managed by the MPI application. Certificates should be associated with a specific Acquirer Bin and are in X.509 format (.der or .pem) files. Whether one or several certificates are to be used, they are loaded on MPI application startup.

The DS data can be configured in a separate xml config file referenced from environment variable:

```
java:comp/env/dsConfigFile = /path/to/dsconfig.xml
or java VM parameter
-DdsConfigFile = /path/to/dsconfig.xml
```

## Application configuration

Sample DS configuration file structure filled with example data:

```
<?xml version="1.0" encoding="UTF-8"?>
<DirectoryServerConfig>
    <Acquirer>
      <acquirerBin>444444</acquirerBin>
      <keystore>/path/to/keystore1</keystore>
      <keystoreType>JKS</keystoreType>
      <keymanagerFactory>SunX509orIbmX509</keymanagerFactory>
      <keystorePassword>keystorePassword1</keystorePassword>
      <keyAlias>keyAlias1</keyAlias>
      <keyPassword>keyPassword1</keyPassword>
      <keyCertAlias>keyCertAlias1</keyCertAlias>
    </Acquirer>
    <DirectoryServer>
      <acquirerBin>444444</acquirerBin>
      [<acquirerBin>4321</acquirerBin>
      <acquirerBin>4567</acquirerBin>]
      <dsUrl>https://dsec.visa3dsecure.com[,https://dswc.visa3dsecure.com,...]</dsUrl>
      <keystore>/path/to/keystore</keystore>
      <keystoreType>JKS</keystoreType>
      <keymanagerFactory>SunX509orIbmX509</keymanagerFactory>
      <keystorePassword>/path/to/keystore</keystorePassword>
      <keyAlias>keyAlias</keyAlias>
      <keyPassword>keyPassword</keyPassword>
      <keyCertAlias>keyCertAlias</keyCertAlias>
      <truststore>/path/to/truststore</truststore>
      <truststorePassword>truststorePassword</truststorePassword>
      <timeout>truststorePassword</timeout>
      <rootCert>/path/to/x509rootcert1.der</rootCert>
      [<rootCert>/path/to/x509rootcert2.der</rootCert>]
      <!--The card used only on Test Merchant Config operation-->
      <predefinedCard>43500000003</predefinedCard>
    </DirectoryServer>
    <DirectoryServer>
      <acquirerBin>4311</acquirerBin>
      <dsUrl>...</dsUrl>
      ...
    </DirectoryServer>
</DirectoryServerConfig>
```

Please note that more than one DS can be configured in this file. One DS config can be applied to multiple Acquirer BIN-s. In this case multiple acquirerBin tags can be configured as shown above.
An Acquirer tag indicates that specific client certificate will be used for acquirer bin 444444. Application will take acquirerBin=444444 directory server configuration as base and override client keystore with one that is configured in Acquirer.acquirerBin=444444.

## Logging

The application logs using Log4j. There is a log of all requests and responses. Sensitive information is masked.

## Versioning

Versioning is supported in such way that wsdl schemas are put into following locations:

```
tomcat/webapps/mpi/WEB-INF/wsdl/v1/
tomcat/webapps/mpi/WEB-INF/wsdl/v2/
etc…
```

Also endpoints are required to have different url-s. For example:

```
  <jaxws:endpoint
          id="mpiserverv1"
          implementor="com.modirum.ws.mpi.web.MpiServicePortTypeV1Impl"
          wsdlLocation="WEB-INF/wsdl/v1/MpiService.wsdl"
          address="/v1">
  </jaxws:endpoint>
  <jaxws:endpoint
          id="mpiserverv2"
          implementor="com.modirum.ws.mpi.web.MpiServicePortTypeV2Impl"
          wsdlLocation="WEB-INF/wsdl/v2/MpiService.wsdl"
          address="/v2">
  </jaxws:endpoint>
etc…
```

As a result version number is included in the service URL, e.g.:

```
https://<host>/mpi/v1/check_enrollment_status
https://<host>/mpi/v1/verify_auhtentication_result
https://<host>/mpi/v1/check_merchant_config

https://<host>/mpi/v2/check_enrollment_status
https://<host>/mpi/v2/verify_auhtentication_result
https://<host>/mpi/v2/check_merchant_config
```

etc.

### Licensing

In order for MPI API to work, license file is needed. It should be referenced from environment variable:

```
java:comp/env/mpiLicenseFile = /path/to/MPI-license.txt
or java VM parameter
–DmpiLicenseFile = /path/to/MPI-license.txt
```

You can obtain the license from support@modirum.com

# API interface

This chapter describes messages Request/Response parameters. The DsUrl, VEReq, VERes, PAReq parameters are informational and should be stored by the client system as proof of the 3-D Secure process.

### Check Enrollment Status

#### Input parameters

| Name | Type | Required | Mapped to in API | Comment |
|---|---|---|---|---|
| CardNumber | 13-19 N | R | pan | |
| ExpMonth | 2 N | R | expiry | Format YYMM |
| ExpYear | 2 N | R | expiry | Format YYMM |
| Amount | 1-12 N | R | purchAmount | Amount in minor units |
| AmountExponent | 1 N | R | exponent | Number of decimals for currency in ISO4217 |
| Currency | 3 N | R | currency | 3 digit currency code from ISO4217 |
| MerchantName | 50 AN | R | merchantName | Might be displayed on the ACS |
| MerchantUrl | 255 AN | R | merchantUrl | Might be displayed on the ACS |
| MerchantCountry | 3 N | R | merchantCountry | ISO3166 3-Numeric code |
| Description | <=125 AN | O | description | Brief |

| | | | | |
|---|---|---|---|---|
| | | | | description of items purchased, determined by the merchant. |
| DsLogin | 32 AN | R | merId | |
| DsPassword | 32 AN | O | merpw | Optional, should be omitted to use certificate. |
| AcquirerBin | 32 AN | R | acqBin | |
| XID | 28 AN | O | xid | Unique transaction identifier, 20 byte statistically unique value that has been Base64 encoded, giving a 28 byte result. Generated if not sent. |
| HttpAcceptHeader | 8192 AN | R | httpAccept | Note, 8Kb is the limit Apache uses, IIS allows for 16Kb |
| HttpUserAgent | 8192 AN | R | httpUserAgent | Note, 8Kb is the limit Apache uses, IIS allows for 16Kb |
| DeviceCategory | 1 N | O | deviceCategory | 0=www (default), 1=mobile, 4=dtv |
| RecurringFrequency | 1-2 N | O | recurFreq | Recurring frequency for PAReq (integer days, 28 means monthly) |
| RecurringEnd | 10 N | D | recurEnd | Recurring end date for PAReq format YYYYMMDD, If recurFreq is |

| | | | | |
|---|---|---|---|---|
| | | | | present then recurEnd is required |
| Installments | 1-3 N | O | installments | Number of Installments for PAReq. Integer value >1 and <=999. *Install and recurring parameters cannot be present at the same time.* |

### Output parameters

| Name | Type | Required | Mapped from in API | Comment |
|---|---|---|---|---|
| XID | 28 AN | R | xid | |
| EnrollmentStatus | 1 A | R | enrolled | Y, N, U, E, X |
| ErrorCode | <50 AN | O | vendorCode | |
| ErrorMessage | <1024 AN | O | mdErrorMsg | |
| ACS Url | <1024 AN | O | formAcsUrl | |
| DsUrl | <1024 AN | R | dsUrl | The URL that was actually used. |
| FormPAReq | <1024 AN | O | formPaReQ | Compressed using ZLIB deflate Base64 encoded request, to be forwarded to the ACS. |
| VEReq | <1024 AN | R | xmlVereq | Base64 encoded xml message. Sensitive data must be masked. |
| VERes | <1024 AN | R | xmlVeres | Base64 encoded xml message. Sensitive data must be masked. |
| PAReq | <1024 AN | O | xmlPareq | Base64 encoded xml message. Sensitive data must be masked. |

## Verify Authentication Result

### Input parameters

| Name | Type | Required | Mapped to in API | Comment |
|---|---|---|---|---|
| PARes | <1024 AN | R | pares | Compressed using ZLIB deflate, Base64 encoded pares xml |
| acquirerBin | 32 AN | O | | acquirerBin from PaReq to match with PaRes |
| dsLogin | 32 AN | O | | dsLogin from PaReq to match with PaRes |
| xid | 28 AN | O | | xid from PaReq to match with PaRes |
| currencyCodeNumerical | 3 N | O | | currencyCodeNumerical from PaReq to match with PaRes |
| cardNumberSuffix | 4 N | O | | cardNumberSuffix from PaReq to match with PaRes |

### Output parameters

| Name | Type | Required | Mapped from in API | Comment |
|---|---|---|---|---|
| paResValidationResult | A | R | | PARES_VALID, SIGNATURE_INVALID, PARAMETER_MISMATCH, ERROR |
| AuthenticationResult | 1 A | R | txstatus | Y, A, N, U |
| errorCode | < 50 AN | O | | Error Code from the MPI Vendor. |
| errorMessage | < 1024 AN | O | | Error Message from the MPI Vendor. |
| Eci | 1 N | O | eci | Note, only populated in case it was received from the ACS. |
| Cavv | <32 AN | O | cavv | Only present if result is Y or A. |
| CavvAlgorithm | 1 N | O | cavvAlgorithm | Required if Cavv is present. |

Notes about paResValidationResult:

| Value | Desc | Result Params | Error Params |
|---|---|---|---|
| **PARES_VALID** | Signature is valid and PaRes is | Y | N |

| | | | |
|---|---|---|---|
| | correctly formatted and has passed parsing. Clients can expect valid values in result params. | | |
| **SIGNATURE_INVALID** | Signature validation failed. No result params and error info in errorMsg/Code. | N | Y |
| **PARAMETER_MISMATCH** | One or more of the parameters in paResValidationInfo does not match values from PaRes. Information provided in errorMsg/Code. | N | Y |
| **ERROR** | General error. Information provided in errorMsg/Code. | N | Y |

"result params" are AuthenticationStaus, Eci, Cavv and CavvAlg.

The client should first evaluate the paResValidationResult. If it's valid, process the result according to the authenticationResult. ErrorMsg/Code is not present. If it's not valid, the result params are excluded and the only possible parameters are errorCode and errorMessage.

## Test Merchant Config

This method uses the input parameters and connection to the DS using pre-defined cards for each DS. The idea is to verify that the merchant config is correct and registered at the DS, the enrollment result is not of interest.

### Input parameters

| Name | Type | Required | Mapped to in API | Comment |
|---|---|---|---|---|
| DsLogin | 32 AN | R | merId | |
| DsPassword | 32 AN | O | merpw | Optional, should be omitted to use certificate. |
| AcquirerBin | 32 AN | R | acqBin | |

### Output parameters

| Name | Type | Required | Mapped from in API | Comment |
|---|---|---|---|---|
| ConnectionResult | A | R | | SUCCESSFUL, |

| | | | | |
|---|---|---|---|---|
| | | | | FAILED |
| ErrorMessage | <1024 AN | O | | Error message |
| DsUrl | <1024 AN | R | dsUrl | The URL that was actually used. |
| VEReq | <1024 AN | R | xmlVereq | Base64 encoded xml message. Sensitive data must be masked. |
| VERes | <1024 AN | R | xmlVeres | Base64 encoded xml message. Sensitive data must be masked. |
| PAReq | <1024 AN | O | xmlPareq | Base64 encoded xml message. Sensitive data must be masked. |

## Official release package verification

Official release packages are signed with Modirum private key. In order to verify the package validity run this command:

```
openssl dgst -sha1 -verify mdcodesign2014-pub.pem -signature mdpaympiws-
[version].zip.sig mdpaympiws-[version].zip
```

mdcodesign2014-pub.pem is located inside of distribution package.