CHESS Replacement Project

Software Provider Readiness Working Group

15 September 2021

ASX

Important Information – Competition Law Policy

Working group members are reminded to have regard to their obligations under competition law. In particular, please note changes to the Competition and Consumer Act to prohibit a corporation from engaging with one or more persons in a concerted practice that has the purpose, effect or likely effect of substantially lessening competition.

Agenda

- Objective of the Software Provider Readiness working group
- CDE10
- Spotlight on AMQP
- Q&A
- Forward Schedule

Objective of the Software Provider Readiness working group

ASX

Software Provider Readiness (SPR) Working Group - Objectives

- Open to software providers, referring to third party vendors who supply software to CHESS users or to those CHESS users that develop software in-house that interfaces directly with CHESS
- To provide clarity and certainty around significant areas such as connectivity requirements to environments, security, test tooling and test strategies
- To ensure readiness for the mandatory technical accreditation phase commencing from late April'22
- Bring together relevant individuals who have the knowledge to collectively advise and discuss fundamental elements, themes and challenges on application delivery
- Allow for an open and transparent forum to discuss specific testing requirements covering system testing, functional testing using migrated data and performance testing
- Establish and manage a collaborative forum which will allow for the open discussion of activities in preparation for and throughout industry test phases
- Encourage software providers to generate topics for discussion and highlight areas for future deep dives

Software provider readiness test phases

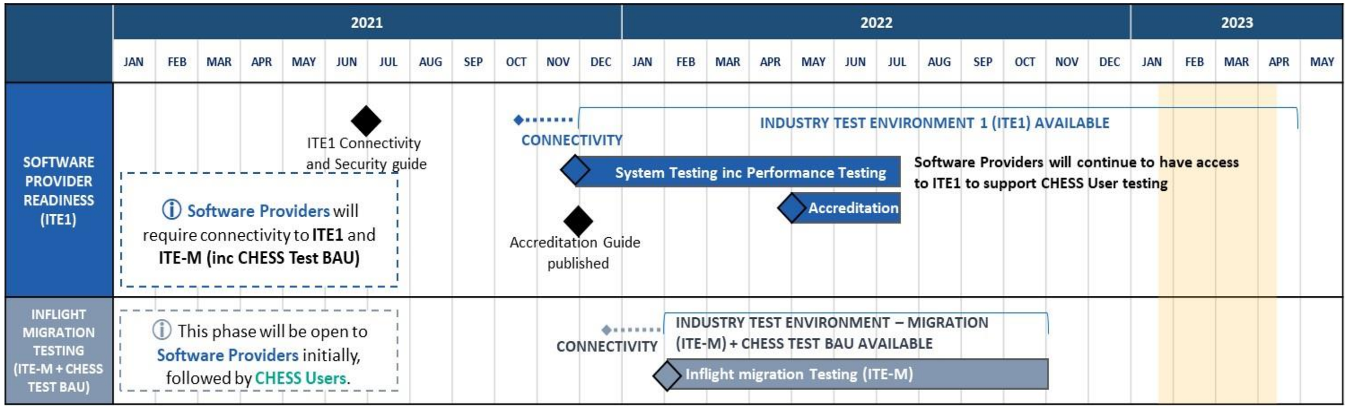
Software provider readiness

Phases of Software Provider Readiness Testing

> Software provider readiness is a stage where software providers have the ability to perform functional and non-functional testing on all functionality

> Software provider readiness testing can be categorised by three main phases:

- System testing (including performance)
- Inflight migration testing
- Technical Accreditation


2021
2022
2023
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC JAN FEB MAR APR MAY
SOFTWARE PROVIDER READINESS (ITE1)
ITE1 Connectivity and Security guide
CONNECTIVITY
INDUSTRY TEST ENVIRONMENT 1 (ITE1) AVAILABLE
System Testing inc Performance Testing
Accreditation
Accreditation Guide published
INFLIGHT MIGRATION TESTING (ITE-M + CHESS TEST BAU)
CONNECTIVITY
INDUSTRY TEST ENVIRONMENT – MIGRATION (ITE-M) + CHESS TEST BAU AVAILABLE
Inflight migration Testing (ITE-M)

Software Providers will require connectivity to **ITE1** and **ITE-M (inc CHESS Test BAU)**

Software Providers will continue to have access to ITE1 to support CHESS User testing

This phase will be open to Software Providers initially, followed by CHESS Users.

CDE Code Drop 10 - Functionality, Test Data and Test Tooling

ASX

CDE 10 - Functionality

- Final functional code drop for CHESS Replacement
- Netting and Settlement Workflow Changes incorporating updates to market trade, netting, reporting and batch settlement.
- The ability to request the following demand reports:
  - Obligation Status Report (OBLG)
  - Counterparty Funds Balance (CFBL)
  - Account Notification Report (ACNT)
- The ability to receive the following end of day reports:
  - Payment Provider Projections
  - Netted Obligation Report (NNDP)
- The ability for a Controlling Participant / Issuer (Registry) to initiate / cancel full and partial DRP/BSP Elections.
- Party Identification support for both a BIC and UIC format is introduced.
- Resolved CDE 9.5 known defects and observed behaviors.

CDE 10 – Testing updates

Category	Change	Change Description
Test Data	Update	Calendar date adjustment to 1 Jan 2023, updates to Issuers & Securities, Account and Holders and Corporate Action seeded data.
CSP Users	Update	Included CDE10 functionality as well as new table mapping UIC/BIC CSP users.
Self service tools	Update	CDE10 introduces enhancements to Self-Service Tool error handling, providing users with Invalid Transaction Notification (comm_807) and Rejected Transaction (comm_808) reject reasons.
CDE Self-Service Account Management	Update	Enhancements to the Self-Service Tool for Bank Account Notifications including: • the ability to specify a Foreign Bank Account.
CDE Self-Service Bulk Account Creation	Update	Enhancement to the Self-Service Tool for Bulk Account Creation: • For each Account successfully created via the Bulk Account Creation Self-Service Tool, the corresponding Account Type, Account Identifier and Holder Id is returned
CDE Self-Service DRP Election and Enquiry	New	New Self-Service Tool provided to Issuer (Registries) to facilitate and perform different DRP Election and Enquiry test scenarios.

CDE 10 – Testing updates

Technical Documentation	Change	Change Description
CDE Self-Service mFund Application	New	New Self-Service Tool provided to provided to PISPs to act as the Controlling Participant to facilitate and perform different mFund Application Order test scenarios.
CDE Self-Service mFund Redemption	New	New Self-Service Tool provided to provided to PISPs to act as the Controlling Participant to facilitate and perform a mFund Redemption Order test scenario.
CDE DRP/BSP Election / Election Enquiry	New	New Auto-Responder functionality, which will act as the Issuer (Registry), responding to a Controlling Participant initiated DRP/BSP Election, DRP/BSP Election Cancellation or DRP/BSP Election Enquiry.
CDE Reserve Bank Information & Transfer System (RITS)	New	New Auto-Responder functionality, which will act as RITS for Bilateral Demand Settlement payments, responding to a RITS Payment Request sent by the CSP to RITS based on the Funds Obligation Amount populated in the Funds Obligation Status Advice.
Known Issues & Limitations	Update	Updated to include CDE 10 known defects and observed behaviors

Focus on AMQP

AMQP Topics

- **AMQP Recap –** Connections, Sessions, Producers, Consumers, Queues, Messages
- **What's New –** Report Notify Queues, Properties
- **Setup Process –** ASX Registration, Certificate Requests, Setup AMQP Client TLS
- **Connection Properties –** URLs, TCP Properties, Reconnection Properties, High Availability/Failover
- **Sending Requests -** AMQP Options, Sample Code
- **Receiving Notifications –** AMQP Options, Duplicate Handling, Sample Code
- **Transactions –** AMQP Transaction Support

AMQP Recap

Connections:

- Single TCP connection, supporting multiple threads (sessions) to access queues,
- Mutual TLS encryption and client authentication,
- Parameters: TCP Timeouts, Reconnections, Security, Set Default-Behaviours. Can often be specified in “Connection URL”

Sessions:

- For ASX, regard as a thread to send request-messages to queue or receive notify-messages from queue
- Specify AcknowledgeMode (“Auto” (for sending) or “Client” (for receiving))

Message Producers and Consumers:

- Create “message producer” for a queue then repeatedly create messages and send them via the producer
- Create “message consumer” for a queue, then receive messages from it repeatedly.
- For ASX, just use separate Session for each Producer and Consumer as the processes aren’t related. (ie. not request/reply)

AMQP Recap

Queues:

> ASX will provide one or more sets of queues to each of you, depending on how many queues you currently have:

> To ASX: <Queue-Prefix>.posttrade.csp.amqp.**trx.request**,

> From ASX: <Queue-Prefix>.posttrade.csp.amqp.**trx.notify**

> You will be advised of your queue names upon registration for each environment

Messages:

> AMQP messages have a Body ("Text" type for ASX), Properties (like HTTP Headers) & Attributes (Expiry, Persisted, etc)

> ASX messages are be persisted and non-expiring (ie. Not lost if there is an outage).

> All messages will be signed ISO 20022 XML in the text-message Body.

> You need to sign the "request" XML message using the special purpose certificate downloaded from ASX

> You need to verify that each "notify" message matches the signature sent with it.

What's New

Report Notify Queues:

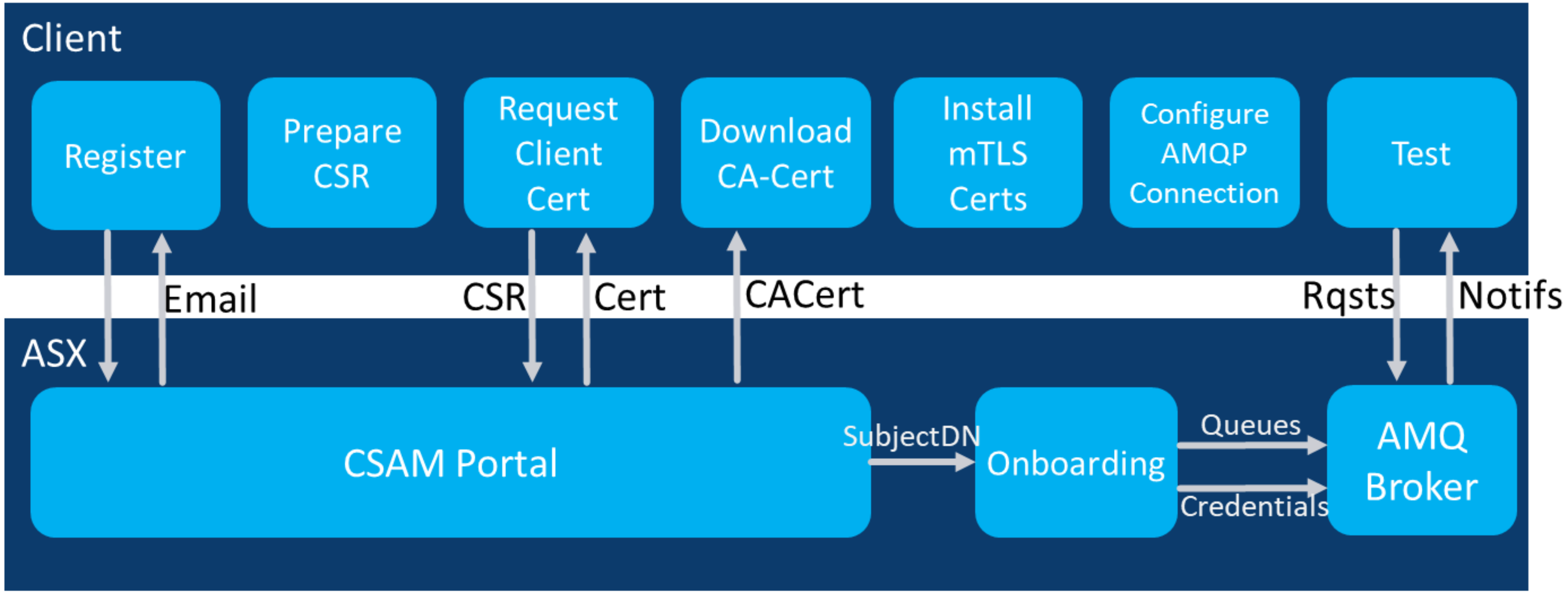
- There will be a new, optional "Report" notify queue where you may elect to receive your reports instead of your main notify queue:
  - From ASX: <Queue-Prefix>.posttrade.csp.amqp.**rpt.notify**

Properties:

- When you send requests, you are now required to set two message properties to help with message tracking:
  - Properties "**BizMsgIdr**" and "**correlationId**" both set to BizMsgIdr from ISO 20022 BAH
  - Setting these two properties will be a mandatory accreditation criterion.

Setup Process

AMQP / Mutual TLS Setup



Note: Certificates for ISO 20022 Signing must also be downloaded from CSAM and installed for use by your client-code

Setup Process

ASX Registration:

- Register via ASX' **Customer Service Account Management (CSAM) Portal**, part of ASX Online:
    - Obtain Login Details,
    - Register a Service Account for specific ASX Test (eg. ITE1, ITE2) or Production environment,
    - Request/Download Certificates for ISO-Message Signing and AMQP/FIX/Ledger-API Mutual-TLS (See next slide),
    - Download ASX Root Certificate-Authority Certificate to trust AMQP at ASX,
- You will be advised of your queue names upon registration for each environment

Setup Process

Certificate Requests:

> Using a Configuration File and your Private Key, generating a Certificate Signing Request (CSR) file allows ASX to create and sign a certificate without having access to your Private Key

> Instructions and some CSR Configuration requirements will be emailed to you on Registration.

> Upload your CSR to CSAM Portal for validation

> If valid, download your new Client Certificate from Portal. This will be used to authenticate you to AMQP

> Download the ASX Certification-Authority (CA) certificates

> In ITE2 and Prod, Queues and Authentication details will be created automatically in ASX from your Certificate's DN's organisation and all your AICs. (Test queue names will be provided in ITE1)

Generate Pub/Pvt Keys

```
$ openssl genrsa -out my_client.key \
    -passout pass:secret123 2048
```

Create Cert-Request Config as per email

```
$ cat >csr.conf <<EOF
[ req ]
default_bits = 2048
prompt = no
default_md = sha256
distinguished_name = dn
[ dn ]
CN = Shares R Us Pty Ltd
C = AU
ST = Western Australia
L = Perth
O = SHARES R US
OU = SVC-CSP
UID = abc12345-8b8b-99999-565656gggg88
EOF
```

Generate CSR

```
$ openssl req -new -key my_client.key \
    -passin pass:secret123 \
    -config csr.conf -out ite1.csr
```

Upload CSR to CSAM Portal

Setup Process

Setup AMQP Client TLS:

- You need to make the downloaded certificates available to your AMQP Client, either:
  - Add them to Key Store and reference them in your Connection configuration:

```
$ openssl pkcs12 -export -chain -in downloads/asx_client_cert.pem \
          -inkey my_client.key -passin pass:secret123 \
          -CAfile downloads/asx_root_cacerts.pem \
          -out asx_amqpclient_keystore.pkcs12 -passout pass:secret123
```

  Add ASX Root CA Certificate(s) to truststore, to trust ASX-Signed Server certificate:

```
$ openssl pkcs12 -export -nokeys -in downloads/asx_root_cacerts.pem \
          -out truststore.pkcs12 -passout pass:notverysecret
```

Windows: Use file-extension ".pfx" for pkcs12-format stores

  - You might need to use other commands or jars if you use Java, Bouncy Castle, etc
- You will need to define these two to your AMQP Client, which might involve defining the Keystore and Truststore file-paths, passwords and store-type to your connection-URL string or to the connectivity property-window.

Connection Properties

URLs:

> You will be advised of the AMQP connection URL and Port when you register for a specific environment. eg

host	`ite1-amqp.asx.com.au`
port	`4005`

Connection Properties

TCP Properties:

> You will be advised of the AMQP connection URL and Port when you register for a specific environment. eg

tcpKeepAlive	TRUE
trustStoreLocation	truststore.pkcs12
trustStorePassword *	notverysecret
trustStoreType	pkcs12
keyStoreLocation	asx_amqpclient_keystore.pkcs12
keyStorePassword *	secret123
keyStoreType	pkcs12
contextProtocol	TLS

*** Generate your own trust and key store passwords**

ASX

Connection Properties

Reconnection Properties:

> If your AMQP Client supports Reconnection or Failover properties, ASX recommends the following settings:

initialReconnectDelay (ms)	**3000**	(Wait 3 seconds before reconnecting)
reconnectDelay (ms)	**3000**	(Retry every 3 seconds, unless…)
useReconnectBackOff	**TRUE**	(…Increase wait time between attempts)
reconnectBackOffMultiplier	**2.0**	(…double wait time after each attempt)
maxReconnectDelay (ms)	**30000**	(…until wait time is 30 seconds, then just retry every 30secs)
maxReconnectAttempts	**-1**	(Keep retrying forever)
startupMaxReconnectAttempts	**10**	(Only try 10 times if connection has never been successful)
warnAfterReconnectAttempts	**10**	(Only log warning-message every 10 attempts)
randomize	**FALSE**	(Not applicable to AMQP at ASX)
amqpOpenServerListAction	**IGNORE**	(ASX Server might suggest other hosts or ports. Ignore them)

Connection Properties

High Availability/Failover:

- To connect to AMQP, there is a single URL/Port per environment.
- The ASX AMQP implementation provides synchronous multi-site redundancy.
- Most failures of ASX AMQP infrastructure will be handled transparently within ASX.
- If your AMQP Client supports reconnection parameters (see previous slide), their use will help your AMQP Client survive network and server glitches transparently, without you having to write special code
- The most common AMQP Client, QPID, offers reconnection parameters as part of its "Failover" facility.
    - Normally, if a server fails, the "failover" protocol attempts to reconnect to the next server in a list
    - Only one ASX AMQP server URL is required as any failover is handled internally by ASX. However, the reconnection properties still work and handle network glitches for you automatically.

Connection Properties

Sample QPID AMQP Client Connection URL:

> Sample URL without reconnection options (Format: "amqps://<host-domain>:<port>?<tcp-query-parameters>"):

```
amqps://ite1-amqp.asx.com.au:4005?
transport.tcpKeepAlive=true&transport.contextProtocol=TLS&
transport.trustStoreLocation=truststore.pkcs12&transport.trustStorePassword=notverysecret&
transport.keyStoreLocation=asx_amqpclient_keystore.pkcs12&transport.keyStorePassword=secret123
```

> Sample URL with reconnection options (Format: "failover:(<amqp-url-with-TCP-options>)?<failover-reconnection-options>")

```
failover:(amqps://ite1-amqp.asx.com.au:4005?
transport.tcpKeepAlive=true&transport.contextProtocol=TLS&
transport.trustStoreLocation=truststore.pkcs12&transport.trustStorePassword=notverysecret&
transport.keyStoreLocation=asx_amqpclient_keystore.pkcs12&transport.keyStorePassword=secret123)?
failover.initialReconnectDelay=3000&failover.reconnectDelay=3000&
failover.maxReconnectDelay=30000&failover.useReconnectBackOff=true&
failover.reconnectBackOffMultiplier=2.0&failover.maxReconnectAttempts=-1&
failover.startupMaxReconnectAttempts=10&failover.warnAfterReconnectAttempts=10&
failover.randomize=false&failover.amqpOpenServerListAction=IGNORE
```

ASX

Sending Requests

AMQP Options – Session Options:

- When sending requests, you need to create a Session, which can either be Transactional or Non-Transactional:
  - Transactional: This is not necessary for ASX "request" messages.
    More details later
  - Non-Transactional:
    Session session = connection.**createSession**(**false**, Session.**AUTO_ACKNOWLEDGE**);
    - If the message cannot be stored by ASX, an error is returned to the AMQP Client. You may need to try resending or advise your source of messages (eg. your database or your input queue) not to remove the message so it can be resent later. This might involve throwing an exception or issuing a roll-back.
    - ASX will check and discard duplicate messages so there is no harm if you resend a request-message unnecessarily.
    - If you can use the "reconnection" options (see previous slide), the AMQP Client will transparently keep retrying automatically, failing only if the problem cannot be fixed by retrying.
    - Otherwise, you will need to detect the problem and resend the message yourself.

Sending Requests

AMQP Options – AMQP Message Options:

> Create an AMQP "Text Message" (ie. the body of the message is to be XML text, not binary or JSON)
> `TextMessage amqpTextMessage = session.createTextMessage(myIsoMsg.getXmlText());`

> You are now required to set two message properties to help with message tracking, both to the BizMsgIdr of the message:

> > **"BizMsgIdr":** `amqpTextMessage.setStringProperty("BizMsgIdr", myIsoMsg.getBizMsgIdr());`
>
> > **"correlationId":** `amqpTextMessage.setStringProperty("correlationId", myIsoMsg.getBizMsgIdr());`

> ASX will later return a notify message with "**correlationId**" set to request's BizMsgIdr but "**BizMsgIdr**" will be a new value, unique for the notify message.

AMQP Options – Message Producer Options:

> Create your Message Producer just after creating your session:
> `MessageProducer messageProducer = session.createProducer(new JmsQueue("sharesrus.54321.ite1.posttr……trx.request"));`

> Just send the AMQP TextMessage with no options. eg.
> `messageProducer.send(amqpTextMessage);`

> When sending a signed ISO 20022 request XML message, ASX will persist it across two data centres to protect against message loss.

> Do not set message expiry or change persistence to "non-persistent" or the message may be lost.

Sending Requests

Sample code

```
try {
    ConnectionFactory factory =
        new JmsConnectionFactory(“failover:(amqps://ite1-amqp.asx.com.au:4005?sslEnabled=true&keyStorePath=.....");
    Connection connection = factory.createConnection();
    connection.setExceptionListener(new MyExceptionListener());
    connection.start();

    Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
    MessageProducer messageProducer = session.createProducer(new JmsQueue(“sharesrus.54321.ite1.posttr……trx.request”));

    SignedIso20022Message myIsoMsg = getNextSignedIsoMsgRequests();          // GET FIRST ISO REQUEST MSGS FROM SOMEWHERE
    while (isoMsg != null) {
        TextMessage message = session.createTextMessage(myIsoMsg.getXmlText());
        message.setStringProperty("BizMsgIdr", myIsoMsg.getBizMsgIdr());     // GET BIZ-MSG-ID OF ISO20022 REQUEST
        message.setStringProperty(“correlationId", myIsoMsg.getBizMsgIdr()); // SAME BIZ-MSG-ID TO CORRELATE NOTIFICATION
        messageProducer.send(message);
        LOG.info("Sent id=" + message.getJMSMessageID() + ",body=" + myIsoMsg.getXmlText());
        myIsoMsg = getNextSignedIsoMsgRequests();                            // GET NEXT ISO REQUEST MSGS FROM SOMEWHERE
    }
    session.close();
    connection.close();
} catch (Exception exp) {
    LOG.error(exp.toString());
    System.exit(1);
}
```

Receiving Notifications

AMQP Options – Session Options:

- When receiving, you need to create a Session, which can either be Transactional or Non-Transactional:
  - Transactional: This is not necessary for ASX "notify" messages.
    More details later
  - Non-Transactional:
    `Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);`
    - Acknowledging received messages is similar to "committing" them so that they are consumed/removed from the ASX notify queue.
    - Use "**CLIENT_ACKNOWLEDGE**" option to explicitly advise AMQP only after you have permanently saved or processed the notification. Otherwise, AMQP will delete the message from ASX before you have processed it.
    - Alternately, you can receive and process a number of messages then acknowledge the last one. AMQP will then send the Acknowledgements for it and the earlier messages and delete them. You might want to do this just after you save the batch of messages to, say, a database, then commit your database changes. This is just the same as for a Transaction.

Receiving Notifications

AMQP Options – Message Consumer Options:

> When creating a Consumer, only specify the notify-queue name. No other parameters are required.

```
MessageConsumer messageConsumer =
                    session.createConsumer(new JmsQueue(“sharesrus.54321.ite1.posttr……trx.notify”));
```

> Receive a notify-message from the queue by invoking the Consumer’s receive(wait-time) method. You will need to invoke this within a loop to retrieve messages. If there are no messages, specifying a wait time will allow the client to wait for a while for a message to arrive, reducing network traffic if you just kept trying with no wait time. If there are still no messages after the wait-time, it will return null.

```
Object msgObj = messageConsumer.receive(30000L); // wait 30 seconds in case there are no messages
```

> Normally, just loop forever, although you may want to include some mechanism to stop looping when you want to shut down your Notification receival-process.

> Don’t forget to verify the notification against the signature!

Duplicate Handling:

> It is possible that a duplicate notification will be received. Extract the “BizMsgIdr” message-property and check if you have already received this notification. Discard it and acknowledge the message if you’ve already processed it successfully.

Receiving Notifications

Sample Code:

```
try {
    ConnectionFactory factory =
        new JmsConnectionFactory(“failover:(amqps://ite1-amqp.asx.com.au:4005?sslEnabled=true&keyStorePath=....."); 
    Connection connection = factory.createConnection();
    connection.setExceptionListener(new MyExceptionListener());
    connection.start();

    Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
    MessageConsumer messageConsumer = session.createConsumer(new JmsQueue(“sharesrus.54321.ite1.posttr……trx.notify”));

    while(checkIfConsumptionToContinue()) {                                 // CUSTOM CODE TO DECIDE CONTINUE VS STOP NOW
        Object msgObj = messageConsumer.receive(30000L);                   // WAIT 30 SECONDS IF NO NOTIFICATIONS
        if (msgObj != null && msgObj instanceof TextMessage) {
            TextMessage message = (TextMessage) msgObj;
            LOG.info(“Received correl=" + message.getStringProperty(“correlationId”)
                    + “BizMsgIdr=“ + message.getStringProperty(“BizMsgIdr”) + ",body=" + message.getText());
            processMsg(message);                                            // OMIT DUPLICATES, CHECK SIGNATURE, PROCESS/SAVE
            message.acknowledge();                                          // ACKNOWLEDGE REMOVES MSG FROM QUEUE
        } // TextMessage
    }

    session.close();
    connection.close();
} catch (Exception exp) {
    LOG.error(exp.toString());
    System.exit(1);
}
```

Transactions

AMQP Transaction support

> AMQP Transactions are not required for ASX.

> AMQP supports only “local” transactions. You cannot commit both AMQP messages and, say, database updates such as via XA.

> When sending “request” messages, there might be a minor performance improvement if you send a number of messages, then commit the AMQP session just before committing the source of your messages (eg. Your database). Your AMQP client may buffer the messages locally and send them to ASX in one transmission only when you commit your session, avoiding the network overhead of a transmission per message. Behaviour might be different with different AMQP clients.

> When receiving “notify” messages, there is really no advantage of using transactions compared with acknowledging individual or batches of messages. Just ensure you commit your database or ensure you have processed the notifications before acknowledging your AMQP messages.

AMQP Checklist for ITE1 Entry

- ✓ Completion of AMQP development
- ✓ ISO Message signing mandatory in ITE1 (optional in CDE)
- ✓ Inclusion of new properties in AMQP header
- ✓ Implement recommended settings
- ✓ Client Applications must be able to discard duplicates (BizMsgId)
- ✓ Give consideration for inclusion of optional reporting queue
- ✓ Give consideration for multiple sets of queues for client side load balancing or business reasons (i.e. share registries, listed vs non-listed, account participants)
- ✓ Reach out to CHESSReplacement@asx.com.au if you have any questions including details of your specific AMQP Client library

ASX

Q&A

Forward schedule

Next steps

Review and feedback:

- Slides and Q&A will be published on the CHESS Replacement webpage
- Provide feedback by contacting **CHESSReplacement@asx.com.au** – use "SPR – WG" as the subject heading

Next Software Provider Readiness Working Group meeting:

- Date: Wednesday 13 October '21

Forward Schedule

- ITE1 Test Kit and test tooling overview
- ITE1 onboarding
- ISO updates (pagination & usage of pilot (!p))
- Ledger API – Token Acquisition
- In-flight migration testing
- Technical Accreditation deep dive
- CSAM

Thank you.

ASX

ASX

Disclaimer