|  |
| --- |
| **DISCOVER**  **MultiChain Evaluation** |
| **Initiation Phase**  **Requirement Specifications** |
| Version Ref : 1.0 |

**Version of the document**

|  |  |  |
| --- | --- | --- |
| **Version No.** | **Date** | **Comments / Modifications** |
| 0.1 | 2017/09/12 | Initial Draft |
| 0.2 | 2017/09/13 | Updated Requirement Specifications. |
| 0.3 | 2017/09/14 | Updated Requirement Specifications. |
| 1.0 | 2017/09/15 | Updated section 2.5 , 2.9 , 2.11 |

**Author(s)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Date** | **Name** | **Function** | **Comments** |
| **Written by** | 2017/09/15 | Vaibhav Keshav Patil | Technical Architect | Draft Version |
| **Checked by** |  | Harminder Singh | BA/Project Manager | Reviewed |
| **Approved by** |  | Rohini Trehan | Delivery Manager | Approved |

**Diffusion list**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Name** | Department | **Validation expected** (Y/N) | **Last date for validation\*** |
| **To :** | Micah Kerr |  | Y |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Copy :** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

\* In case of absence of any comments from you before the identified date, the document will be considered as validated.

**Reference documents**

|  |  |
| --- | --- |
| **Document** (+ short explanation) | Version No |
| MultiChain Streams (Service Bus) PoC.PPTX | v0.0 |
|  |  |

**INDEX**

1. Introduction 4

1.1. Purpose of the document 4

1.2. Definitions, Acronyms and Abbreviations 4

1.3. Scope 4

2. Tasks 5

2.1 ME\_01\_01: Control Creation and publish permissions by an admin node 5

2.2 ME\_01\_02: Development and verification of Stream list capabilities, stream indexing and retrieval capabilities 6

2.3 ME\_01\_03: Demonstrate stream confidentiality capabilities 6

2.4 ME\_01\_04: Demonstrate ability for a reader of a stream to create a blockchain transaction confirming receipt of a transaction from a stream (reflects audit and traceability of a transaction) 7

2.5 ME\_01\_05: Ability to publish to multiple streams simultaneously 7

2.6 ME\_01\_06: End to end Auditing to ensure publishing happened and received by receiving system 8

2.7 ME\_01\_07: Test solution capabilities for preventing subscribers going out of sync due to delay in data confirmation 8

2.8 ME\_01\_08: Adapter/Bridges/API/UI Development 9

2.9 ME\_01\_09: Comparison with RabbitMQ 9

2.10 ME\_01\_010: Performance Testing 10

2.11 ME\_01\_011: Documentation 10

# Introduction

## Purpose of the document

The Purpose of this document is to elaborate overall requirements of the Multichain Evaluation project. There are number of tasks to be performed as a part of multichain evaluation and Discover expects the development and verficiation of the tasks listed in the document. The approach to be followed for each task would also be listed in the document. The document will help in decomposing the high level tasks shared as a part of initial requirements and arriving at the final scope of the project.

The aim of the project is to evaluate the multichain Streams feature for the purpose of using as a messaging system/Service Bus.

## Definitions, Acronyms and Abbreviations

**Acronyms:**

* ***XXX*** : **XXX**.
* ***IT*** : **I**nformation **T**echnology.

**Definitions:**

## Scope

The document captures the software system requirements for Multichain evaliation using the following tasks:

### Task Catalogue

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Priority (1,2,3)** | **Task – Scenario** | **Task ID** |
|  | 1 | ME\_01\_01: Control Creation and publish permissions by an admin node | 2.1 |
|  | 1 | ME\_01\_02: Stream list capabilities, stream indexing and retrieval capabilities | 2.2 |
|  | 2 | ME\_01\_03: Demonstrate stream confidentiality capabilities | 2.3 |
|  | 2 | ME\_01\_04: Demonstrate ability for a reader of a stream to create a blockchain transaction confirming receipt of a transaction from a stream (reflects audit and traceability of a transaction) | 2.4 |
|  | 2 | ME\_01\_05: Ability to publish to multiple streams simultaneously | 2.5 |
|  | 2 | ME\_01\_06: End to end Auditing to ensure publishing happened and received by receiving system | 2.6 |
|  | 2 | ME\_01\_07: Test solution capabilities for preventing subscribers going out of sync due to delay in data confirmation | 2.7 |
|  | 1 | ME\_01\_08: Adapter/Bridges/API/UI Development | 2.8 |
|  | 3 | ME\_01\_09: Comparison With RabbitMQ | 2.9 |
|  | 2 | ME\_01\_10: Performance Measurement | 2.10 |
|  | 3 | ME\_01\_11: Documentation | 2.11 |

# Tasks

## ME\_01\_01: Control Creation and publish permissions by an admin node

Description : Permission to create, write and publish can be controlled by the administrative user and secondary nodes cannot create, write or publish into streams without Admin’s consent.

The Primary node (Admin) will create a chain where as other nodes will merely connect to it. Therefore, Streams can only be created by the Admin of chain and only admin will decide who can create, write / publish to streams of that chain. The permission to write and publish can also be revoked by admin.

In addition to this we can also check authorization in the application itself that will be used to connect to multichain.

SubTasks :

1. Creation of Chain.
2. Configuration of Admin Node.
3. Creation of secondary Nodes and configuration of permissions for stream.

High Level Approach:

1. Creation of Chain.
2. Configuration of Admin Node.
3. Creation of secondary Nodes and configuration to join network.
4. Secondary Node request Admin Node to join network.
5. Admin node grants permission to connecting node to join network.
6. Admin creates streams
7. Secondary node subscribes to stream.
8. Admin node may give permission to write / publish to stream.

Feasibility to be checked: NA

## ME\_01\_02: Development and verification of Stream list capabilities, stream indexing and retrieval capabilities

Description : If a node is subscribed to a stream, information can be retrieved from that stream in a number of ways:

* + Retrieving items from the stream in order.
  + Retrieving items with a particular key.
  + Retrieving items signed by a particular publisher.
  + Listing the keys used in a stream, with item counts for each key.
  + Listing the publishers in a stream, with item counts.

Following is the sample body of a stream item :

[

{

"publishers" : [

"1X876A5C565hEWFqs9rq5cYJudPaKhPqithYVP"

],

"key" : "key1",

"data" : "736f6d65206f746865722064617461",

"confirmations" : 5,

"blocktime" : 1505385318,

"txid" : "5a38b698e46ad8c10894ece6f3c3836fcb10933200f871f13dccb48e49b300

68 }]

Streams have Keys associated with each data record. These Keys can be chosen by the publisher. Each published record will have Timestamp and a verification number. Therefore, the publish time can be audited.

Streams can contain multiple items with the same key, and this naturally solves the tension between blockchain immutability and the need to update a database.

Reference:

(<https://www.slideshare.net/coinspark/multichain-private-multicurrency-blockchain-platform>, Slide 10)

SubTasks :

1. Creation of Chain.
2. Configuration of Admin Node.
3. Creation of secondary Nodes and configuration of permissions for stream.

High Level Approach:

1. Creation of Stream by admin node.
2. Subscription by seconday nodes
3. Retiving data from stream based on publisher.
4. Retriving data based on Key
5. Retriving all the data records for a particular key.

Feasibility to be checked:

Multichain allows multiple data records to be published using same Key. This solves the problem where a user may want to update the record. Confirmation would be required in order to allow nodes to publish multiple records for same Key.

Priority (alongwith Remarks): NA

## ME\_01\_03: Demonstrate stream confidentiality capabilities

Description : Stream confidentiality can be achieved by using combination of streams. For example-

* + DataStream (containing encrypted data)
  + AccessStream (containing password that is created using public key).
  + PublicKeyStream (subscriber publishes public key upon subscription)

Data in DataStream can be password protected and kept in AccessStream whereas password in AccessStream would be created based on public key of the reader so that only the reader with that particular public key would be able to read the data.

Reference: <https://www.multichain.com/developers/stream-confidentiality/>

SubTasks :

1. Creation of Chain.
2. Configuration of Admin Node .
3. Creation of Seconday nodes
4. Creation of streams

High Level Approach:

1. Creation of admin and seconday Node.
2. Admin node creates streams and grants permission for PublicKeyStream
3. Secondary Nodes subscribe to streams and publish their public key to PublicKeyStream
4. Admin uses public key to generate passwords and DataStream.
5. Data records published by admin or any other publisher upon permission.
6. Seconday node gets the password using their public key.
7. Seconday node decrypts data record using password.

Feasibility to be checked: NA

## ME\_01\_04: Demonstrate ability for a reader of a stream to create a blockchain transaction confirming receipt of a transaction from a stream (reflects audit and traceability of a transaction)

Description : In order to confirm receipt of data from stream, the subscriber node can use another stream associated only for acknowledgement and publish the acknowledgement status with the Key of data record for which the acknowledgemnt is being generated.

Publishing the acknowledgement in another stream would create the transaction and this transaction detail can be used for audit.

This method would work like a 2 way channel system giving more scope for publisher and subscriber.

Also subscriber is free to choose when to generate acknowledgement, it could be instantly or post processing the data.

This approach can help identify the result upon processing the data.

Data Stream

Subscriber

Publisher

Acknowledgemnt Stream

SubTasks :

1. Creation of Chain.
2. Configuration of Admin Node.
3. Creation of Secondary Nodes.
4. Creation of Data stream
5. Creation of Acknowledgement stream.

High Level Approach:

1. Creation of admin and seconday Node.
2. Admin node creates streams and grants permission
3. Publisher publishes data on stream
4. Subscriber reads data from stream.
5. Subscriber processes data.
6. Subscriber sends positive or negative acknowledgement
7. Publisher reads acknowledgement and may send updated data record.

Feasibility to be checked:

Need to check feasibility of this approach with business requirement.

## ME\_01\_05: Ability to publish to multiple streams simultaneously

Description : Multichain has given functionality to publish to multiple distinct streams in one single command. So items can be published to multiple distinct streams in one single command execution. However, it is not possible to create multiple items for the same stream in one transaction but can be controlled programmatically. There can be multiple publishers for one stream, but each publisher must have permissions assigned by admin in order to publish to any stream.

The list of streams in network can be retrived and based on business logic, the publisher may publish to multiple streams simultaneously.

Reference: <https://www.multichain.com/developers/raw-transactions/>

SubTasks :

1. Creation of Chain.
2. Configuration of Admin Node.
3. Cretion and configuration of Secondary Nodes
4. Creation of Streams
5. Permission and Subscription to streams.

High Level Approach:

1. Creation of Chain.
2. Configuration of Admin Node.
3. Cretion and configuration of Secondary Nodes
4. Creation of Streams
5. Permission and Subscription to streams.
6. Publish data records to multiple streams simultaneously.

Feasibility to be checked: NA

## ME\_01\_06: End to end Auditing to ensure publishing happened and received by receiving system

Description : Each record published will have a hashcode in response confirming that a transaction has been created. The stream will have attributes like- publisher, timestamp, key, verification, transaction id. In return the reader can send acknowledgement via another stream and this would create a transaction with timestamp that would be useful in audit. Hence an end to end auditing can be achieved.

SubTasks :

1. Creation of Chain.
2. Configuration of Admin Node.
3. Cretion of Secondary Nodes
4. Creation of Streams
5. Permission and Subscription to streams.
6. Publisher Publishes data records to streams
7. Suubscriber processes data and confirms with ACK.

High Level Approach:

1. Creation of Chain configuration of Admin Node
2. Cretion of Secondary Nodes
3. Creation of Streams
4. Permission and Subscription to streams.
5. Publisher Publishes data records to streams and transaction is created.
6. Suubscriber processes data and confirms with ACK.
7. Transaction is created for ACK.

Feasibility to be checked: NA

## ME\_01\_07: Test solution capabilities for preventing subscribers going out of sync due to delay in data confirmation

Description : All nodes in Blockchain are always in synchronization with others. Blocks are ordered accoreding to timestamp in ascending order.

A custom approach of centrally controlled keys may also help in tracking unsync data.

SubTasks :

1. Creation of Chain.
2. Configuration of Admin Node
3. Cretion of Secondary Nodes
4. Creation of Streams
5. Permission and Subscription to streams.
6. Publisher Publishes data records to streams and transaction is created.
7. Suubscriber processes data and confirms with ACK.
8. Transaction is created for ACK.

High Level Approach:

1. Creation of Chain.
2. Configuration of Admin Node.
3. Cretion of Secondary Nodes
4. Creation of Streams
5. Permission and Subscription to streams.
6. Publisher Publishes data records to streams and transaction is created.
7. Suubscriber processes data and confirms with ACK.
8. Transaction is created for ACK.
9. End to End Auditing can be seen on **View** page.

Feasibility to be checked: NA

## ME\_01\_08: Adapter/Bridges/API/UI Development

Description :

Adapters:

Verify the feasibility to transfer data to external applications using streams: In order for external application to read data, JSON-RPC API can be used.

UI:

1. Admin page for creation of streams and controlling its WRITE permissions.
2. Basic UI page to trigger the transaction and display the messages processed by each Node.
3. Capture the message triggering and reception timestamps and acknowledgement.

SubTasks :

1. Creation of Mock external system.

Creation of simple UI template.

High Level Approach:

* Simple UI to display all the operation results

## ME\_01\_09: Comparison with RabbitMQ

Description : Understand and compare how a MultiChain streams based Event/Service Bus compares to RabbitMQ :

In RabitMQ the publish request goes to “Exchange” and then routes to a particular Queue. Data is kept on this Queue until the consumer sends an acknowledgement. Once acknowledgement is received, data is removed from Queue.

In multichain Streams work as the combination of Exchange and Queue. Data is shared with all the subscriber but can never be removed. Acknowledgement can be achieved by using dual stream architecture.

Parameters for comparision of Multichain with RabbitMQ will be listed later on.

High Level Approach:

1. List down the features in RabbitMQ and Multichain
2. Compare features and performance.

## ME\_01\_010: Performance Testing

Description : Identify any limiting factors/restrictions with using MultiChain Streams speed, data limits, etc. Potential limiting factors/restrictions and potential workarounds, if available

SubTasks :

1. Testing in terms of speed.
2. Testing in terms of Data Bandwidth.
3. Testing in terms of data limit.
4. Testing with standard 10k Payload.
5. Testing overall application in terms of potential bugs.

## ME\_01\_011: Documentation

**Description :** Creation of templates of the documents that would record the results of each task and would be shared with Discover for validation.

Templates for documenting the findings would be provided later.

SubTasks : Document on following will be created :

1. Setup and Installation.
2. Solution Architecture.
3. Application flow Description.
4. Comparision report
5. multichain list features.