FRAMEWORK DEVELOPER MANUAL

KATUN

BLOCKCHAIN

GENERAL INFORMATION

Document Information.	
Tittle	Framework Developer Manual
Version	1
Status	Released
Release Date	June 3th. 2019
Author	Valeria Hernández
Contributors	

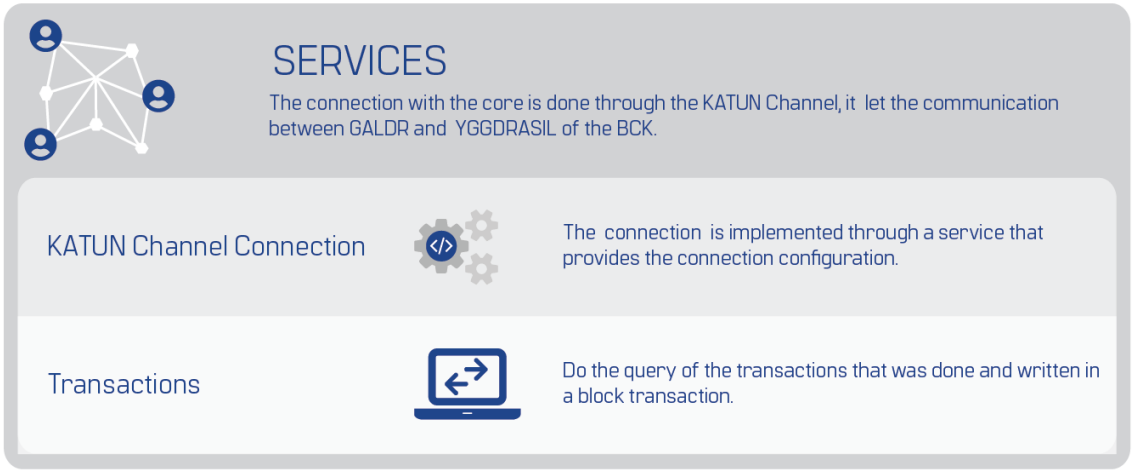
CHANGE CONTROL

Version	Description.	Author	Change date.
1.0	Base document	Massiel Pasaflores	

CONTENT

1. ANTECEDENTS

Katun framework is a reusable software component that saves developers time by providing access to code that performs a programming task. This work tool was created to connect Katun blockchain with different technologies that the user will use, using the tool in an intermediary way in the JAVA language.



2. OBJECTIVE

To know the services that the framework provides to the developer can use it.

3. SCOPE

Explain framework functionalities in a developer context, define the events they can perform with the BlockChain Katun, and define the constrains they must follow to successfully implement a development with this framework.

4. COMPONENTS

Below are defined the functionalities supported by the framework as the developer can use to implement the tools, He or the organization needs to consume some of the Blockchain Katun features.

4.1. Framework

The framework has control over the decisions that take part in its elaboration. Therefore, the user can connect to the Katun blockchain and can receive requests through it.

The objective is to design a framework that facilitates the connection with Katun Blockchain. When creating a network server, there are several problems to consider:

- Accept connections
- Access the sequences represented by a connection
- Process the incoming data in some way and send a response.

4.1.1. Connection

The framework is implemented to use the network through a service that provides the connection configuration. This type of connection is called **Distributed** and returns the information of the any available node in the Blockchain katun to be used to perform the events.

There is a Default connection configuration the developer can use only if is known the node information to be used to perform the events.

The developer can choose the type of configuration and other parameters to work with the framework in the Configuration File.

4.1.2. Configuration File

The framework includes a file called ConnectSettings.json that define the configurations the developer can use to build their applications with the next parameters:

The next arguments might be defined in all cases (distributed or manual connection type) in the ConnectSettings.json

- CONN_TYPE: Defines how to connect with the katun network.
- DIR_URL: Defines the url of the distributed connection, the developer can use to access to the blockchain katun network.

- USER_NAME: Defines the username of the distributed connection, the developer can use to access to the blockchain katun network.
- PASSWORD: Defines the password of the distributed connection, the developer can use to access to the blockchain katun network.

Usually this Manual configuration is set when the developer works with and specific node and Katun provides

- CA_CORP_URL: Certificate Authenticator URL
- ORDERER_URL: URL of Orderer services in Blockchain Katun.
- CORP_PEER: Peer DNS/URL related to the node the developer uses to work with the framework.
- CORP_PEER_URL: Peer DNS/URL related to the node the developer uses to work with the API.

The next in an example of how to build a ConnectSettings**.json** connection File:

```
{
    "CONN_TYPE": "Distributed",

    "DIR_URL": "18.216.237.194",
    "USER_NAME": "diskatun1",
    "PASSWORD": "G3n3sis1",

    "CA_CORP_URL": "http://3.17.227.26:7054",
    "ORDERER_URL": "grpc://3.17.227.26:7051",
    "CORP_PEER": "peer0.corp001.katundomain.com",
    "CORP_PEER_URL": "grpc://3.17.227.26:7051"
}
```

4.2. Example Package

katun.network: Set of Class *KatunChannel* that shows the method called *connectKatunChannel* that let connect with the katun channel and the can to developer events with the connection success.

5. Katun Examples Services

5.1 Block Info

The developer can consult all transactions through the connection with the katun channel in the BlockTransactionsExample class, this classes have some method that returns a string with the all transactions saved in the block before.

Also, can consult some information like the height of the block, the block number, the current hash of the block and the previous hash of the block.

```
 */
public class BlockTransactionsExample {

    KatunBlockInfo kbi = new KatunBlockInfo();

    public static long getKatunBlockNumber() {...6 lines }

    public static long getKatunBlockHeight() {...6 lines }

    public static String getKatunBlockCurrentHash() {...6 lines }

    public static String getKatunBlockPreviousHash() {...6 lines }

    public static String getKatunBlockTransactions() {...5 lines }
```

```
run:
log4j:WARN No appenders could be found for logger (katun.helper.Config).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
jun 30, 2019 11:37:04 PM katun.client.KatunCAClient enrollAdminUser
ADVERTENCIA:  admin is already enrolled.
katunBlockNumber:399
jun 30, 2019 11:37:07 PM katun.client.KatunCAClient enrollAdminUser
ADVERTENCIA:  admin is already enrolled.
blockHeight:401
jun 30, 2019 11:37:08 PM katun.client.KatunCAClient enrollAdminUser
ADVERTENCIA:  admin is already enrolled.
currentHash:6f51851e3aa119f7d009c5f29e96fc0008d12fc0683cd63954821b31c247ee93
jun 30, 2019 11:37:08 PM katun.client.KatunCAClient enrollAdminUser
ADVERTENCIA:  admin is already enrolled.
previousHash:afd9870ea1f2ae41d510959f55c8cefca58801b6e1f4bf2114d412f4fea47e86
jun 30, 2019 11:37:09 PM katun.client.KatunCAClient enrollAdminUser
ADVERTENCIA:  admin is already enrolled.
blockTransactions:{"transactions":[{"TransactionId":"f6d0a9e327e10b906788fcfdc0fe0df5a1b850800b8d844bcf7cea91f22ac036","time
BUILD SUCCESSFUL (total time: 37 seconds)
```

Example log from class BlockTransactionsExample.

5.1 Connection katun channel.

The developer can consult he connection through the connection with the katun channel .

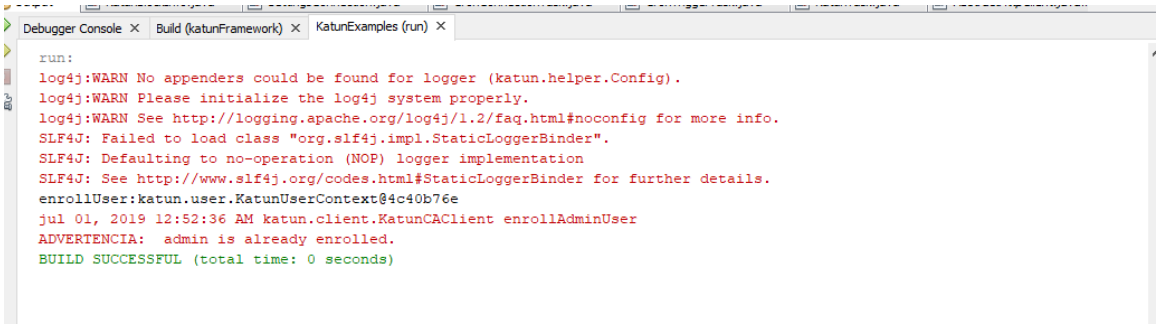
```java
import java.util.logging.Level;
import java.util.logging.Logger;
import katun.client.KatunChannelClient;
import katun.network.KatunChannel;

/**
 *
 * @author Valeria-Katun
 */
public class KatunConnectionExample {

    public static KatunChannelClient getKatunConnection() {...11 lines }

    public static void main(String[] args) throws Exception {
        KatunChannelClient channel = getKatunConnection();
        System.out.println("channel:" + channel);
    }
}
```

```
run:
log4j:WARN No appenders could be found for logger (katun.helper.Config).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
jul 01, 2019 12:10:00 AM katun.client.KatunCAClient enrollAdminUser
ADVERTENCIA:  admin is already enrolled.
channel:katun.client.KatunChannelClient@4b7dc788
jul 01, 2019 12:10:02 AM katun.network.KatunChannel connectKatunChannel
INFORMACIÓN: Is Connected to katunChannel.
BUILD SUCCESSFUL (total time: 2 seconds)
```

5.1 Enrollment User katun .

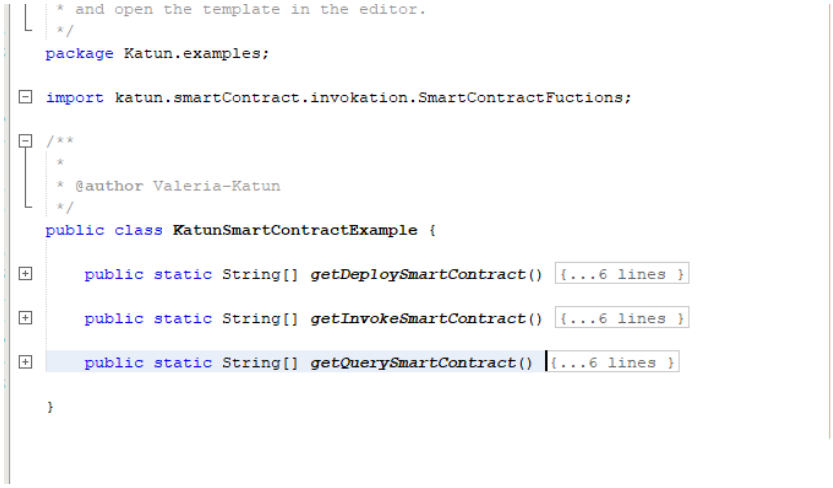
The developer can consult the user context from the connection with the katun channel.

```java
package Katun.examples;

import java.util.logging.Level;
import java.util.logging.Logger;
import katun.network.EnrollUserKatun;
import katun.user.KatunUserContext;

/**
 *
 * @author Valeria-Katun
 */
public class UserEnrrollmentExample {
    public static KatunUserContext getKatunUserEnroollment(){...10 lines }

    public static void main(String[] args) throws Exception {
        KatunUserContext enrollUser = getKatunUserEnroollment();
        System.out.println("enrollUser:" + enrollUser);
    }
```



5.1 Smart Contract Functions

The principal function In smart contract are: deploy the smart contract to will use, Invoke or execute the smart contract and finally query the information of the smart contract.



Katun Tech	
NOMBRE CARGO/ ROL	**NOMBRE** CARGO/ ROL
NOMBRE CARGO/ ROL	**NOMBRE** CARGO/ ROL