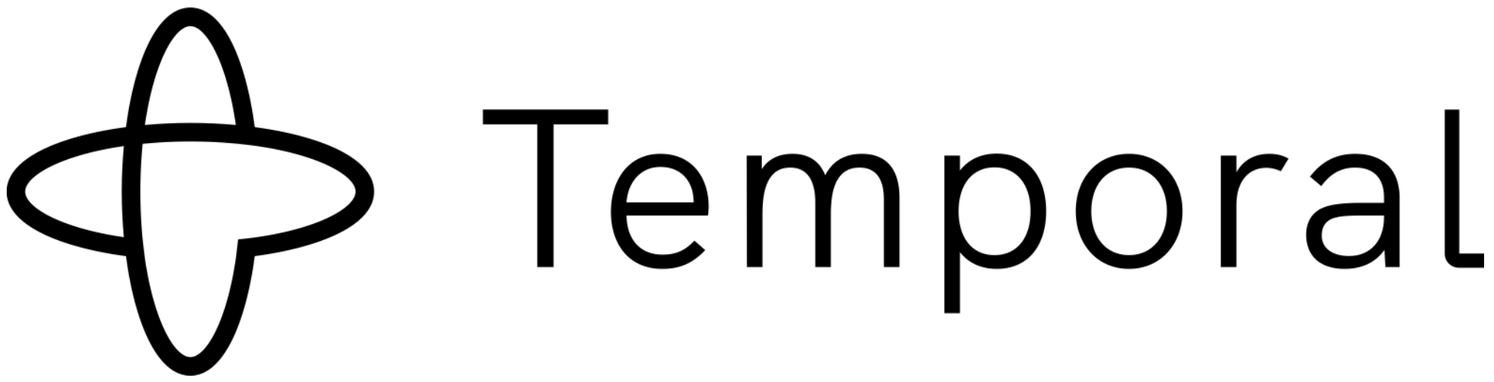
Building invincible systems with



Xpansiv

KONSTANTIN IGNATYEV
Chief Architect
+1 425.233.4536
kignatyev@xpansiv.com

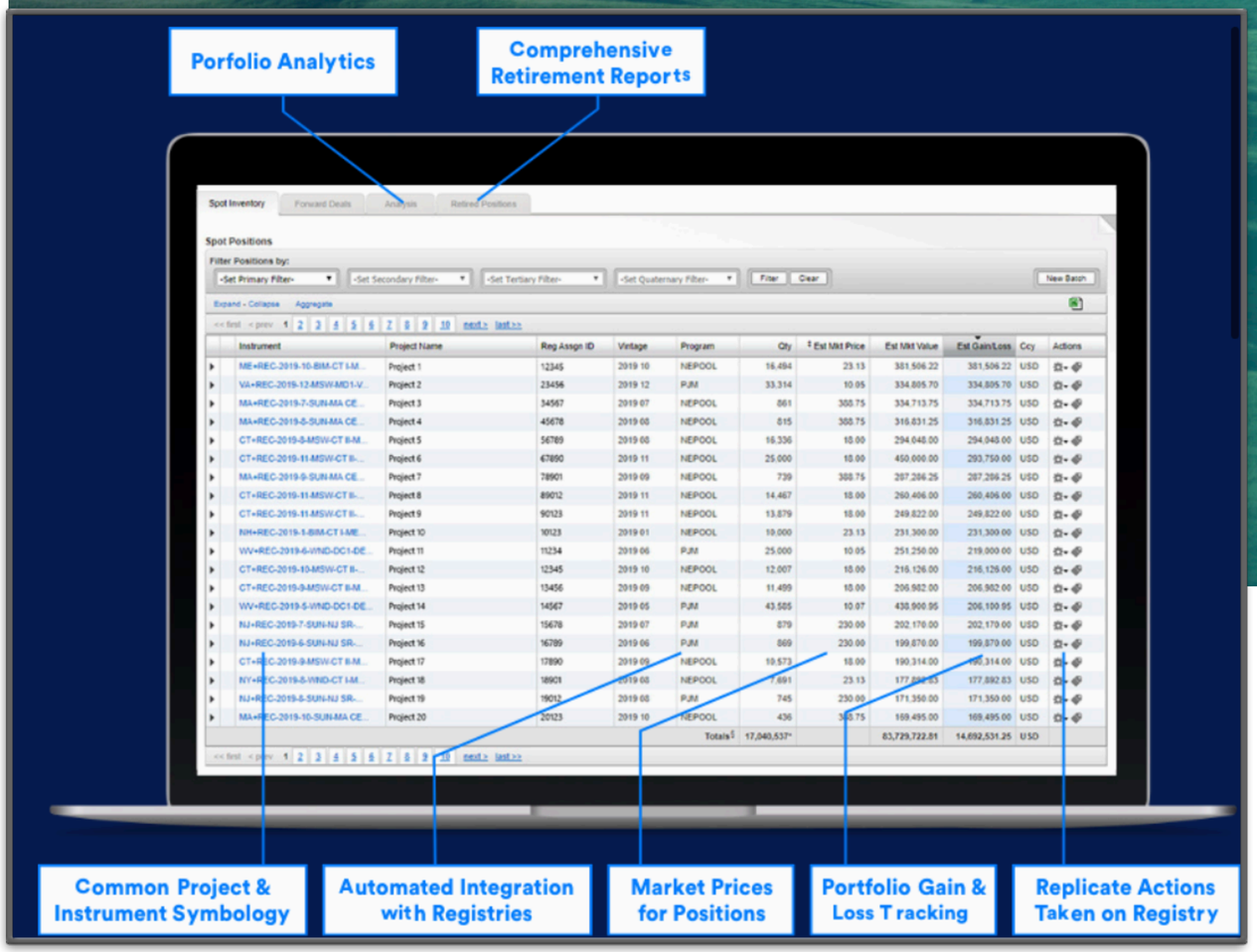
EMA

Xpansiv

Register, Manage, Trade, Settle, Retire, and Analyze Your Environmental Commodities

EMA

AN XPANSIV SOLUTION



Registies

American Carbon Registry

CLIMATE ACTION RESERVE

ercot

MIRECS

NAR
NORTH AMERICAN RENEWABLES REGISTRY

NC-RETS
North Carolina Renewable Energy Tracking System

NEW YORK STATE OF OPPORTUNITY.
New York Generation Attribute Tracking System

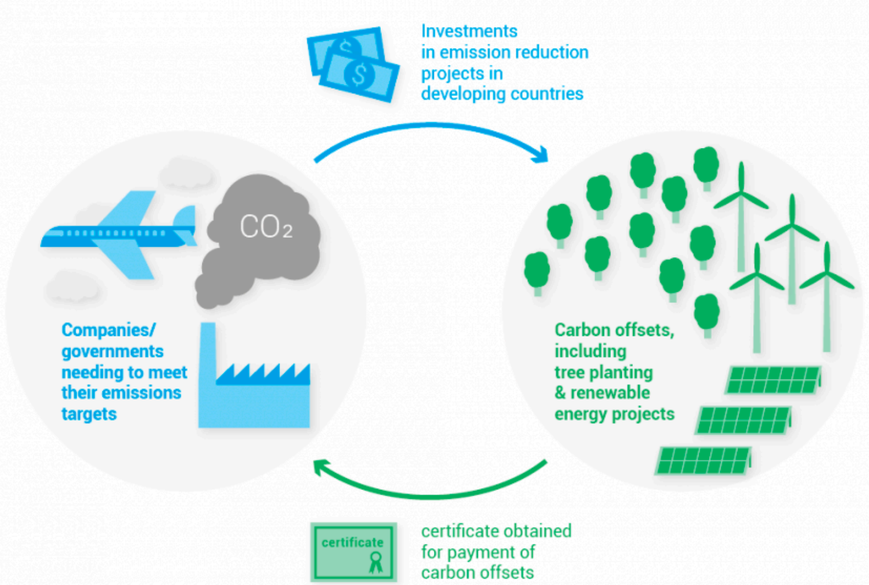
NEPOOL

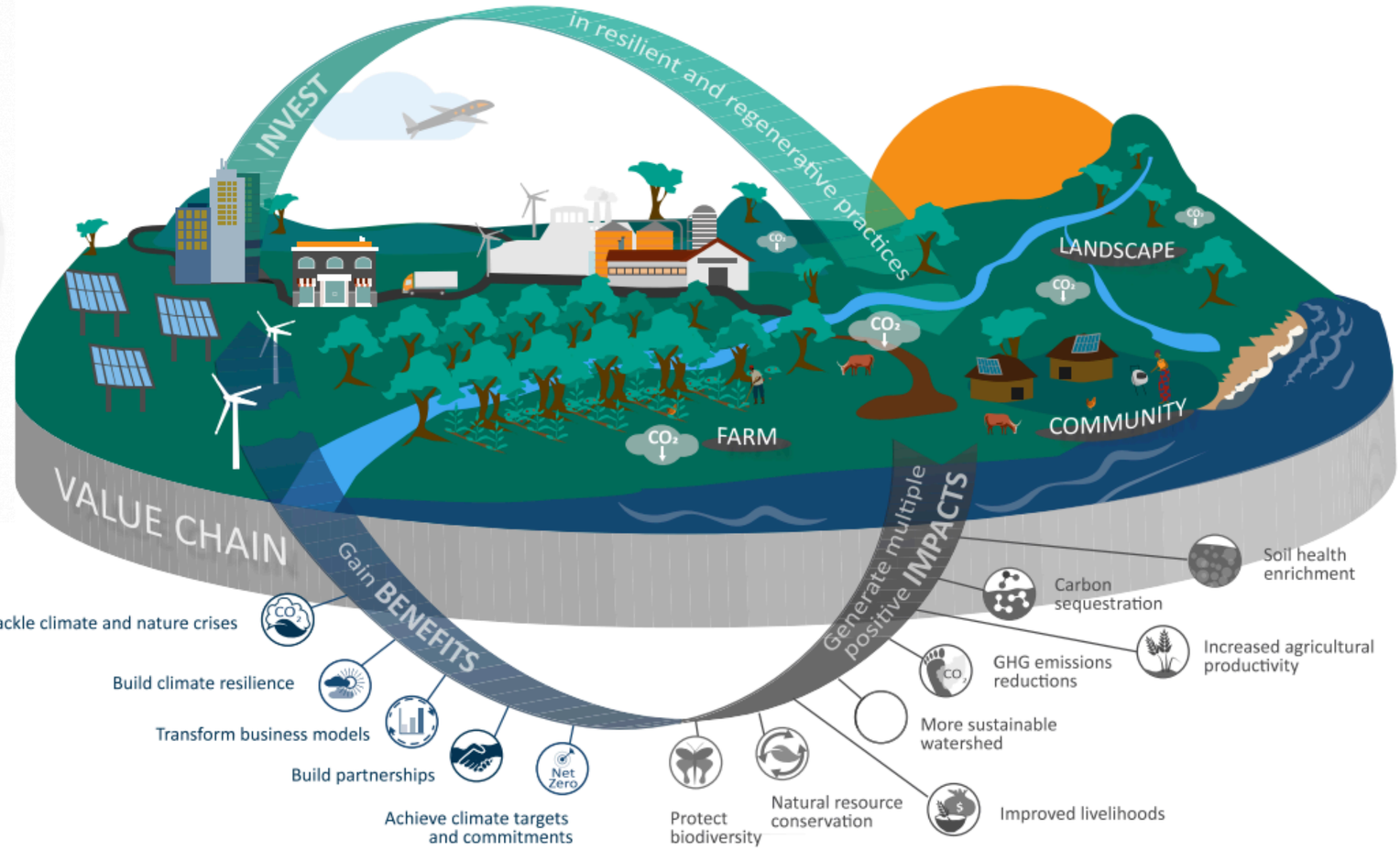
pjm EIS
Environmental Information Services

Verified Carbon Standard

WECC

How carbon offsets and renewable energy credits work



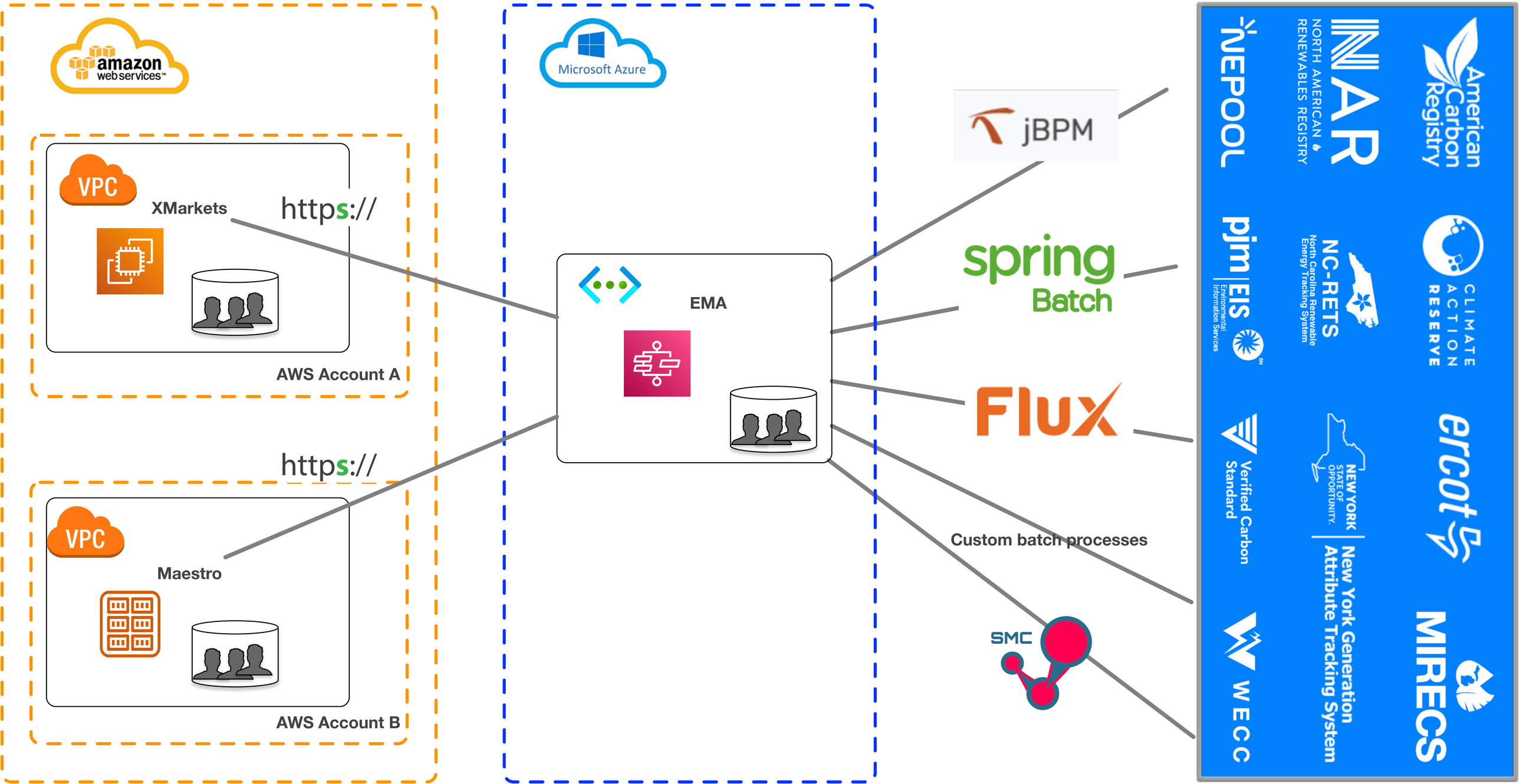


ACR Standards
Approved Methodologies
In Scientific Peer Review
Open for Public Comment
In Development
Inactive
ACR Validation and Verification Standard

APPROVED METHODOLOGIES

Sectoral Scope	Methodology	Version
1. GHG emission reductions from fuel combustion	Truck Stop Electrification	1.1
2. GHG emission reductions from industrial processes	Advanced Refrigeration Systems	2.1
2. GHG emission reductions from industrial processes	Certified Reclaimed HFC Refrigerants, Propellants, and Fire Suppressants	2.0
2. GHG emission reductions from industrial processes	Destruction of Ozone Depleting Substances and High-GWP Foam	1.2
2. GHG emission reductions from industrial processes	Destruction of Ozone Depleting Substances from International Sources	1.0

Different applications built on different technology stacks in the past 15 years, each has own user DB and various batch systems



Why Temporal.IO ?

Camunda Platform

(62) **4.4** out of **5**

Optimized for quick response

Innovating process automation with a standards-based, highly scalable and collaborative approach for business and IT. A community of tens of thousands of users across companies such as Allianz, ING, and Vodafone design, automate and improve mission-critical business processes end-to-end with Camunda, enabling them to build software applications more flexibly, collaboratively and efficiently, gaining the business agility, visibility and scale needed to drive digital transformation.

Reviews | Alternatives

Kogito

Appian

(272) **4.5** out of **5**

Appian provides a leading low-code software development platform that enables organizations to rapidly develop powerful and unique applications. The applications created on Appian's platform help companies drive digital transformation and competitive differentiation. For more information, visit www.appian.com.

Categories in common with Camunda Platform:
Business Process Management

Pega Platform

(257) **4.2** out of **5**

Pega is the only industry-leading platform that allows you to build scalable, enterprise-grade CRM, Digital Process Automation, BPM, Case Management, and AI apps, all on one unified platform.

Categories in common with Camunda Platform:
Business Process Management

Step Functions

IBM Business Automation Workflow (BAW)

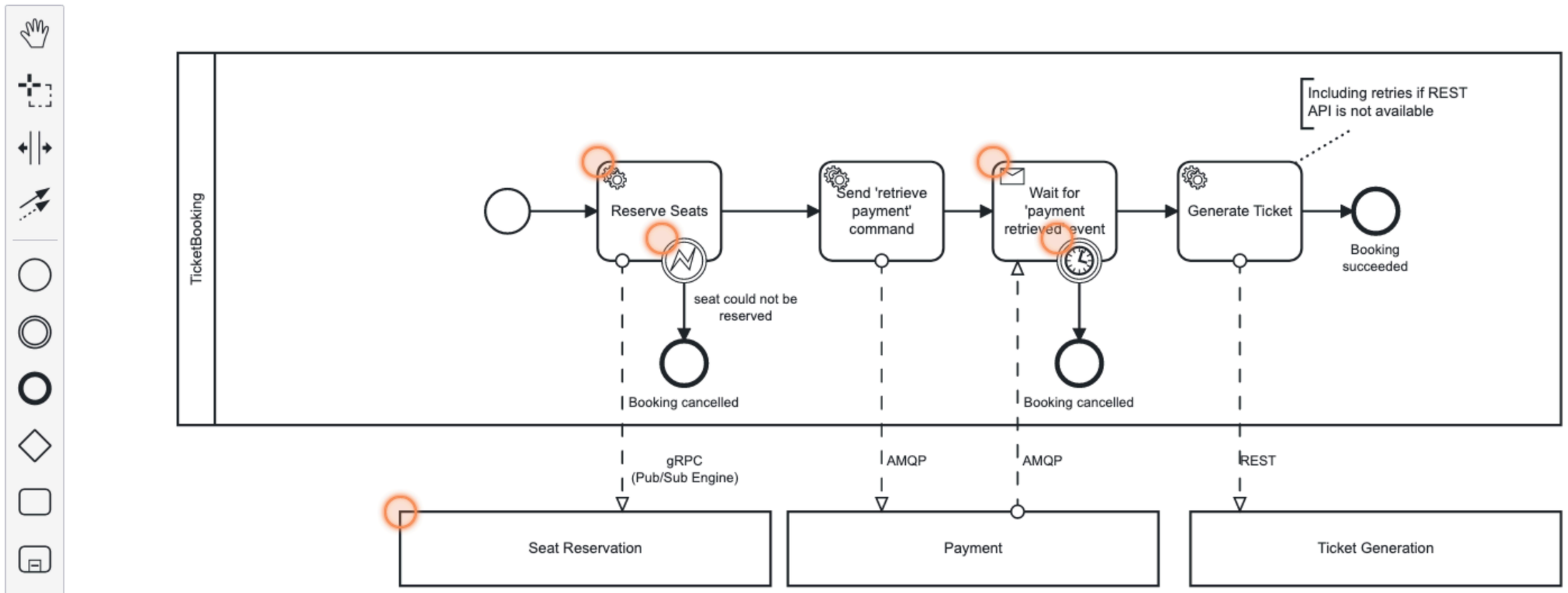
(105) **4.4** out of **5**

Optimized for quick response

Automate your digital workflows, from straight-through or human-assisted processes to managing complex cases be it on-premise or on Cloud

Categories in common with Camunda Platform:
Business Process Management

Apache Airflow



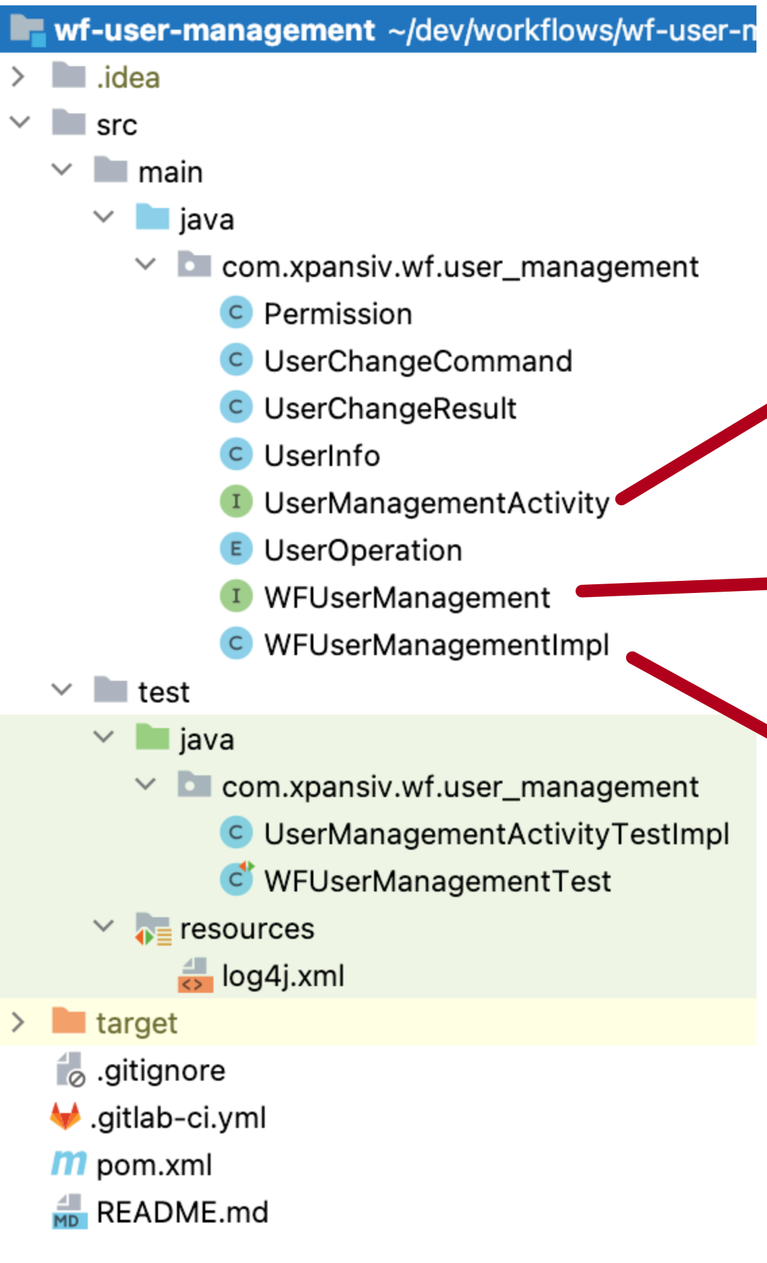
Why Temporal.IO ? - No XML hell for one!

ticket-booking.bpmn

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:di="http://www.omg.org/
spec/DD/20100524/DI" xmlns:zeebe="http://camunda.org/schema/zeebe/1.0" xmlns:modeler="http://camunda.org/schema/modeler/1.0" xmlns:
camunda="http://camunda.org/schema/1.0/bpmn" id="Definitions_1a4gt8m" targetNamespace="http://bpmn.io/schema/bpmn" exporter="Camunda
Web Modeler" exporterVersion="62a42ab" modeler:executionPlatform="Camunda Cloud" modeler:executionPlatformVersion="8.0.0" camunda:
diagramRelationId="5d811647-c4c2-48bc-8d57-3e7d197f91cd">
  <bpmn:collaboration id="Collaboration_0t1vxrh">
    <bpmn:participant id="template-ticket-booking-participant" name="TicketBooking" processRef="template-ticket-booking" />
    <bpmn:participant id="Participant_1l2gxa0" name="Payment" />
    <bpmn:participant id="Participant_0y2e8ja" name="Seat Reservation" />
    <bpmn:participant id="Participant_1i465kq" name="Ticket Generation" />
    <bpmn:messageFlow id="Flow_11rhvrd" name="could be gRPC(Pub/Sub Engine)" sourceRef="Activity_1g89uec" targetRef="
    Participant_0y2e8ja" />
    <bpmn:messageFlow id="Flow_08b5p57" name="could be AMQP" sourceRef="Activity_0lox1kf" targetRef="Participant_1l2gxa0" />
    <bpmn:messageFlow id="Flow_1lm74ik" name="could be AMQP" sourceRef="Participant_1l2gxa0" targetRef="Activity_0h19mxb" />
    <bpmn:messageFlow id="Flow_02bhzx9" name="REST" sourceRef="Activity_0etdda4" targetRef="Participant_1i465kq" />
  </bpmn:collaboration>
  <bpmn:process id="template-ticket-booking" isExecutable="true">
    <bpmn:startEvent id="StartEvent_1">
      <bpmn:outgoing>Flow_19ebf54</bpmn:outgoing>
    </bpmn:startEvent>
    <bpmn:receiveTask id="Activity_0h19mxb" name="Wait for &#39;payment retrieved&#39; event" messageRef="Message_04xnjbt">
      <bpmn:incoming>Flow_0mwrroh</bpmn:incoming>
      <bpmn:outgoing>Flow_0podi1a</bpmn:outgoing>
    </bpmn:receiveTask>
    <bpmn:endEvent id="Event_1ne48r4" name="Booking cancelled">
      <bpmn:incoming>Flow_18sbo4b</bpmn:incoming>
    </bpmn:endEvent>
    <bpmn:endEvent id="Event_05eni0s" name="Booking cancelled">
      <bpmn:incoming>Flow_012b8wg</bpmn:incoming>
    </bpmn:endEvent>
    <bpmn:endEvent id="Event_1mqmjv4" name="Booking succeeded">
      <bpmn:incoming>Flow_19x2sei</bpmn:incoming>
    </bpmn:endEvent>
    <bpmn:boundaryEvent id="Event_1ywtp2y" name="after 5 minutes" attachedToRef="Activity_0h19mxb">
```

Simple User management Workflow
(Unit tested and packaged in a versioned JAR)

1. Naive implementation




```java
@ActivityInterface
public interface UserManagementActivity {
    3 usages  1 implementation  Konstantin Ignatyev
    UserChangeResult execute(UserChangeCommand command);
}
```

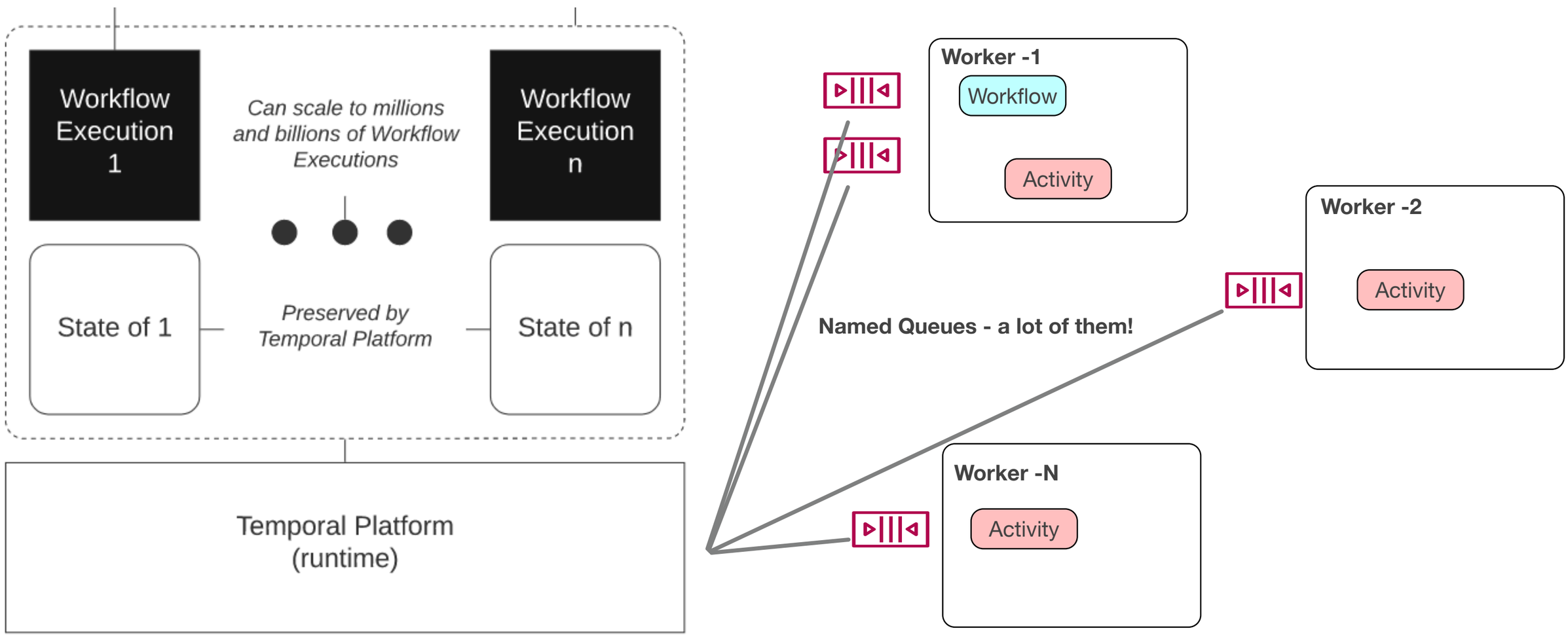
```java
@WorkflowInterface
public interface WFUserManagement  {
    @WorkflowMethod
    void executeCommand(UserChangeCommand cmd);
```

```java
public class WFUserManagementImpl implements WFUserManagement {
    @Override
    public void executeCommand(UserChangeCommand cmd) {
        xmarketsUserManagement.execute(cmd);
        maestroUserManagement.execute(cmd);
        emaUserManagement.execute(cmd);
    }
```

How Temporal.IO works

A Temporal Workflow Execution is a Reentrant Process. A Reentrant Process is resumable, recoverable, and reactive.

- Resumable: Ability of a process to continue execution after execution was suspended on an *awaitable*.
- Recoverable: Ability of a process to continue execution after execution was suspended on a *failure*.
- Reactive: Ability of a process to react to external events.



	Date & Time	Workflow Events	
8	2022-11-28 UTC 20:12:25.59	**WorkflowTaskCompleted**	Scheduled Event ID 6
7	2022-11-28 UTC 20:12:25.57	**WorkflowTaskStarted**	
	Scheduled Event ID 6		
	Identity 5029@KGI-Xpansiv-MBP.hsd1.wa.comcast.net		
	Request ID 6d579df3-7671-42cb-8474-5e14c6dafb0c		
	Suggest Continue As New false		
	History Size Bytes 0		
6	2022-11-28 UTC 20:12:25.56	**WorkflowTaskScheduled**	Task Queue Name asset_transfers_wf
5	2022-11-28 UTC 20:12:25.56	**WorkflowExecutionSignaled**	Signal Name LockSenderFunds
4	2022-11-24 UTC 01:01:08.39	**WorkflowTaskCompleted**	Scheduled Event ID 2
3	2022-11-24 UTC 01:01:08.19	**WorkflowTaskStarted**	Scheduled Event ID 2
2	2022-11-24 UTC 01:01:08.18	**WorkflowTaskScheduled**	Task Queue Name asset_transfers_wf
1	2022-11-24 UTC 01:01:08.18	**WorkflowExecutionStarted**	Workflow Type Name WFAssetsTransfer

Supported SDKs

The following language SDKs are supported. Language tab selection is preserved through a browser cookie. The following language is selected:

Kotlin

Go | Java | PHP | Python | TypeScript

Java is currently selected!

Third-party SDKs

The following third-party SDKs exist but are not supported in the Developer's guide:

- Clojure - from @Manetu
- Scala from @vitaliihonta
- Ruby from @coinbase

SDKs in development

The following SDKs are in alpha/pre-alpha development stages,

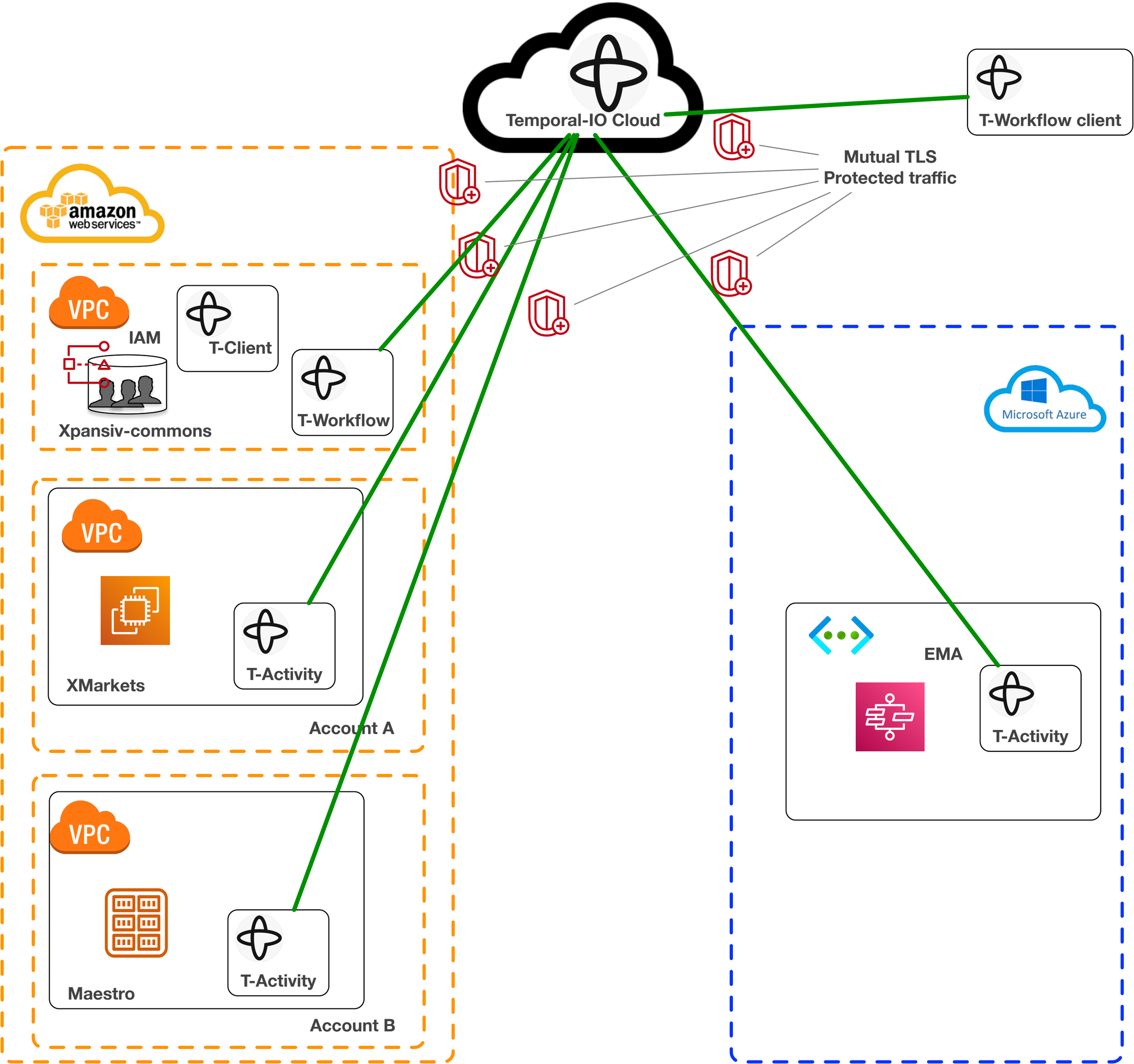
- .NET
- Rust
- Ruby

Any tech stack that supports gRPC can participate.

Even shell scripts via cli (tctl)

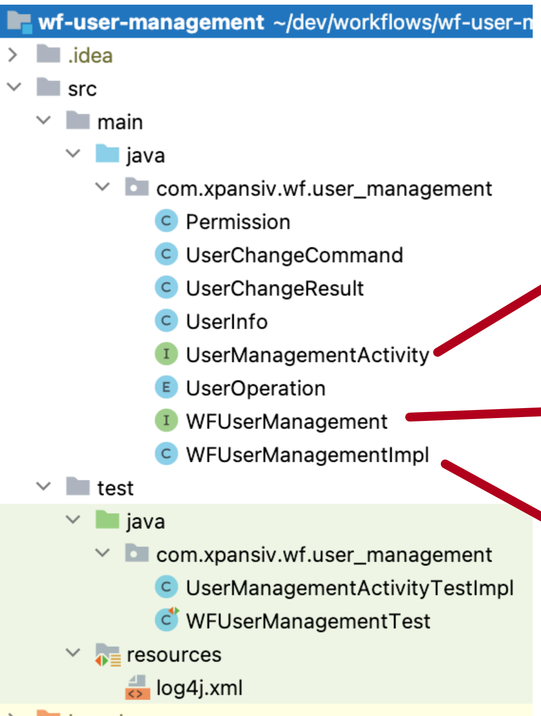
And it is Open Source - anyone can create SDK

Simple User management Workflow



Simple User management Workflow

1.1 Naive implementation (with timeouts)




```java
@ActivityInterface
public interface UserManagementActivity {
    3 usages   1 implementation   Konstantin Ignatyev
    UserChangeResult execute(UserChangeCommand command);
}
```

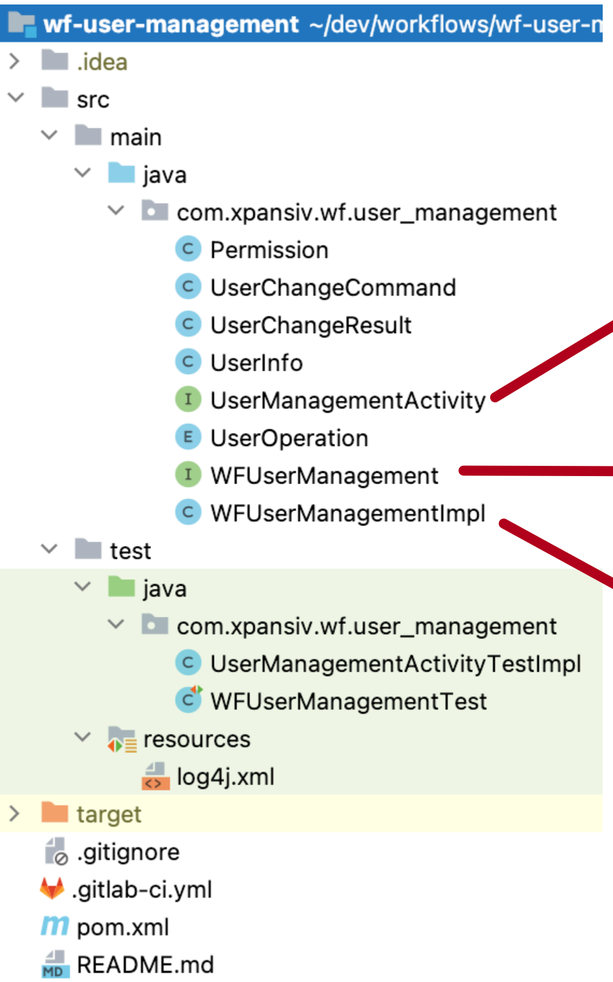
```java
@WorkflowInterface
public interface WFUserManagement  {
    @WorkflowMethod
    void executeCommand(UserChangeCommand cmd);
```

```java
public class WFUserManagementImpl implements WFUserManagement {
    @Override
    public void executeCommand(UserChangeCommand cmd) {
        xmarketsUserManagement.execute(cmd);
        maestroUserManagement.execute(cmd);
        emaUserManagement.execute(cmd);
    }
```

```java
    private final ActivityOptions maestroOptions =buildActivityOptions(WFUserManagement.TASK_QUEUE_MAESTRO);

    private final UserManagementActivity maestroUserManagement =
            Workflow.newActivityStub(UserManagementActivity.class, maestroOptions);


    3 usages   Konstantin Ignatyev
    public ActivityOptions buildActivityOptions( String queueName ){
        return ActivityOptions.newBuilder()
                .setTaskQueue(queueName)
                .setRetryOptions( RetryOptions.newBuilder()
                        .setMaximumAttempts( 10 )
                        .setBackoffCoefficient(1.0)
                        .setInitialInterval( Duration.ofSeconds(1))
                        .build())
                .setStartToCloseTimeout(Duration.ofSeconds(5)).build();
    }
```

Simple User management Workflow

2 Async, parallel, non-blocking implementation




```
6   @ActivityInterface
7   public interface UserManagementActivity {
        3 usages   1 implementation   Konstantin Ignatyev
8       UserChangeResult execute(UserChangeCommand command);
9   }
```

```
6   @WorkflowInterface
7   public interface WFUserManagement  {
18      @WorkflowMethod
19      void executeCommand(UserChangeCommand cmd);
```

```
8   public class WFUserManagementImpl implements WFUserManagement {
34      public void executeCommand(UserChangeCommand request) {
35          promisesMap.put("xmarkets", Async.function(xmarketsUserManagement::execute, request));
36          promisesMap.put("maestro", Async.function(maestroUserManagement::execute, request));
37          promisesMap.put("ema", Async.function(emaUserManagement::execute, request));
38          Promise.allOf(promisesMap.values()).get();
39      }
```

Lets dive deeper:
WorkflowId, RunId, Search Attributes, Sync and Async executions

Recent Workflows

seajug

Advanced Search

Workflow ID | Workflow Type | All | All | UTC

100 | 1–8 of 8

Status	Workflow ID	Type	Start	End
Running	ae4d83f2-4678-486b-bea4-a8e116a0c475	WFAssetsTransfer	2022-11-24 UTC 01:01:08.18	
Completed	e235c44f-22ec-44c0-82f8-b27f71d715bf	WFUserManagement	2022-11-29 UTC 02:11:04.84	2022-11-29 UTC 02:11:25.48
Completed	f2bcb940-a127-4687-b479-c2caa52c6137	WFUserManagement		
Failed	f2bcb940-a127-4687-b479-c2caa52c6137	WFUserManagement		
Completed	u4	WFUserManagement		
Completed	u4	WFUserManagement		

Completed **u4**

History 17 | Workers 1 | Pending Activities 0 | Stack Trace | Queries

Workflow Type: WFUserManagement
Run ID: 2bb6d37c-0f9b-43cb-bb15-03422ef4b890
Start Time: 2022-11-29 UTC 22:16:38.75 **Close Time:** 2022-11-29 UTC 22:16:38.82
Task Queue: user_management
State Transitions: 11

```
40          val wfStub = temporalio.client.newWorkflowStub( WFUserManagement::class.java,
41              WorkflowOptions.newBuilder()
42                  .setTaskQueue(WFUserManagement.WF_TASKS_QUEUE)
43                  .setWorkflowId( cmd.userInfo.id) //we can reuse workflow ID
44                  .build()
45          )
46  // synchronized WF start
47  //            wfStub.executeCommand(cmd)
48  //async start
49          val wfExecution = WorkflowClient.start(wfStub::executeCommand, cmd)
50          println("Started workflow: ${wfExecution.workflowId}")
51      }
```

Namespace: seajug

Details

Description:
Owner: Unknown
Global? No
Retention Period: 3 days
History Archival: Disabled
Visibility Archival: Disabled
Failover Version: 0
Clusters: active (active)

Versions

Temporal Server Version: 1.18.5
Temporal UI Version: 2.8.3

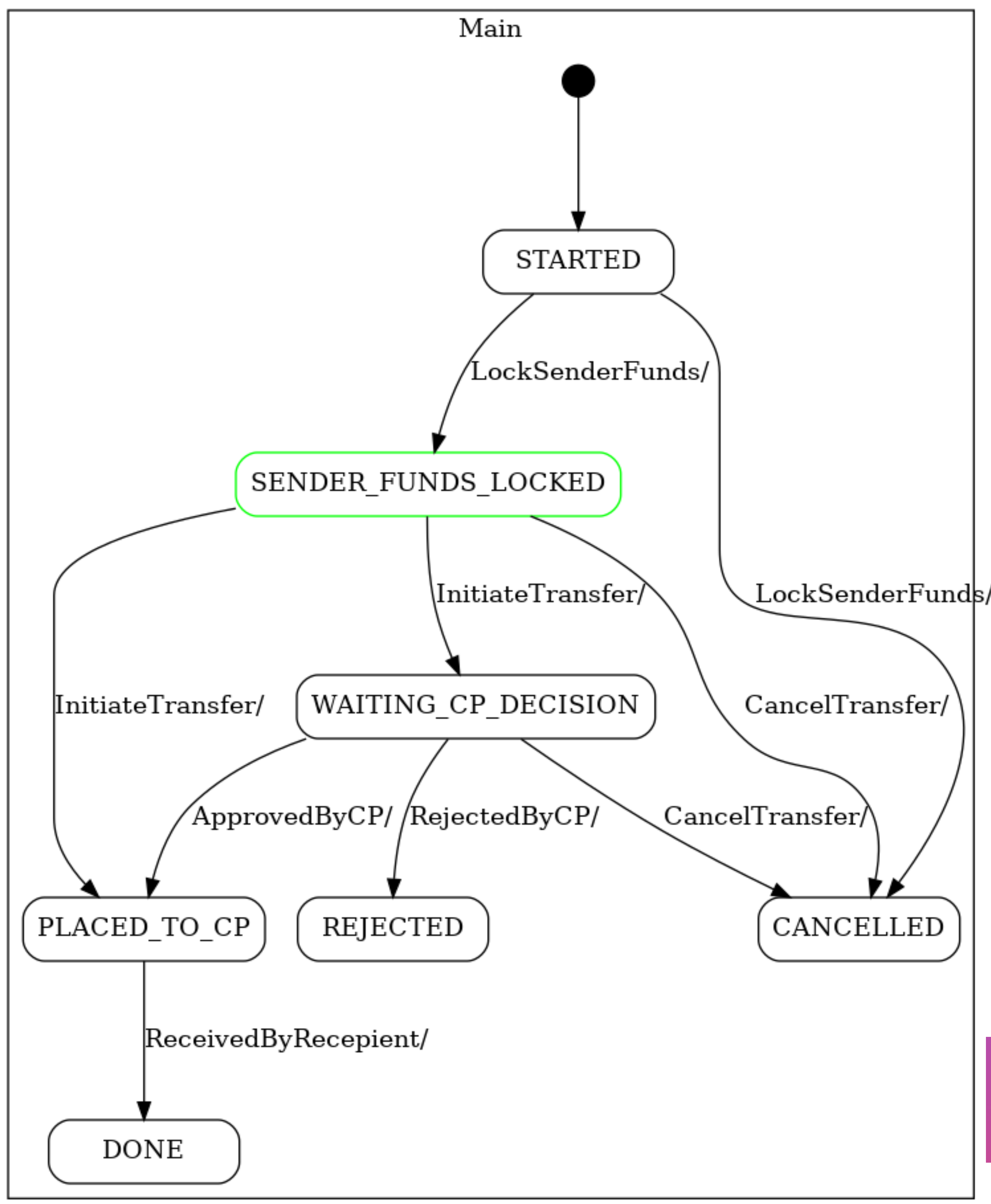
Search Attributes

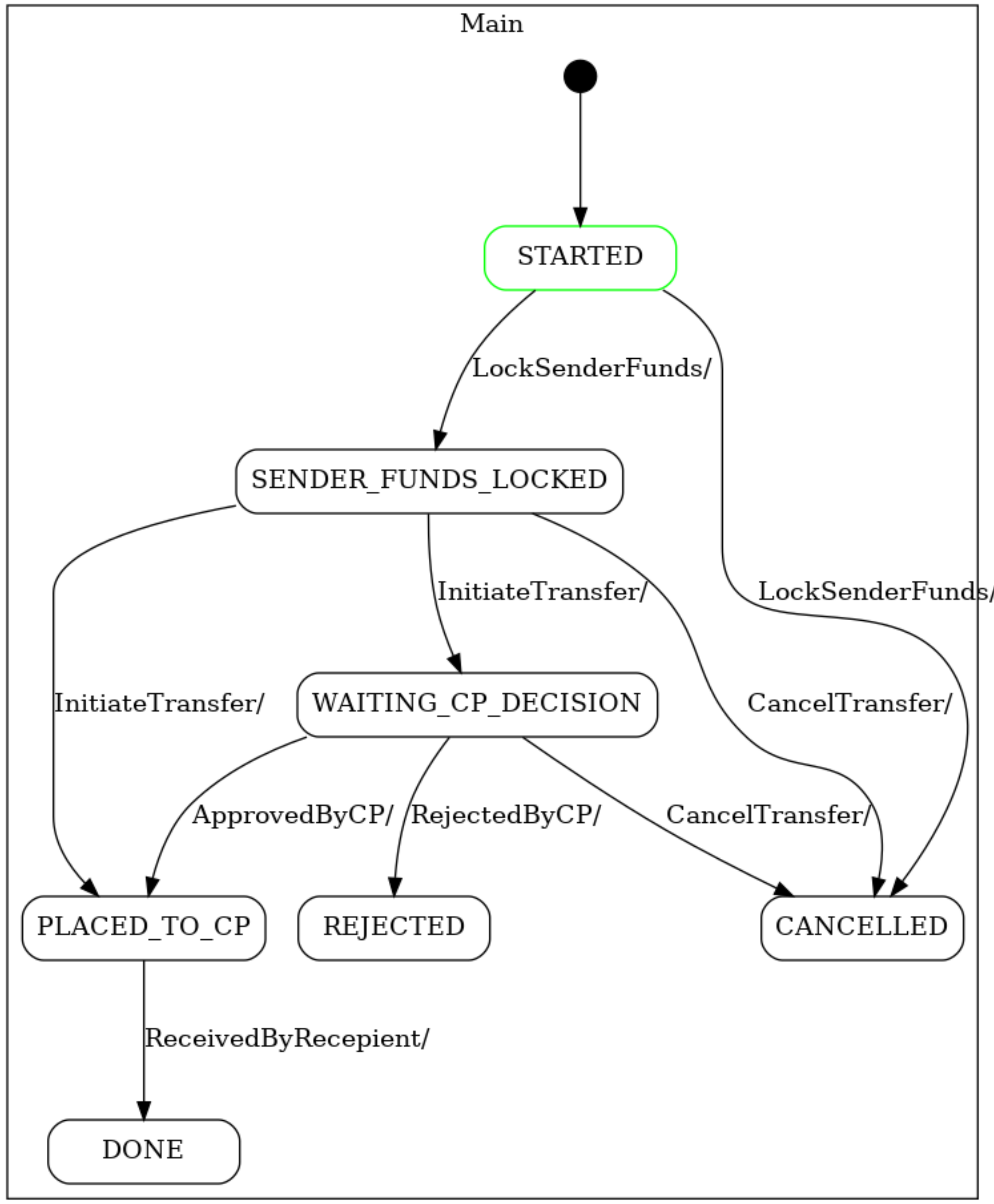
Key	Type
BatcherNamespace	Keyword
BatcherUser	Keyword
BinaryChecksums	Keyword
CloseTime	Datetime
CustomBoolField	Bool
CustomDatetimeField	Datetime
CustomDoubleField	Double
CustomIntField	Int
CustomKeywordField	Keyword

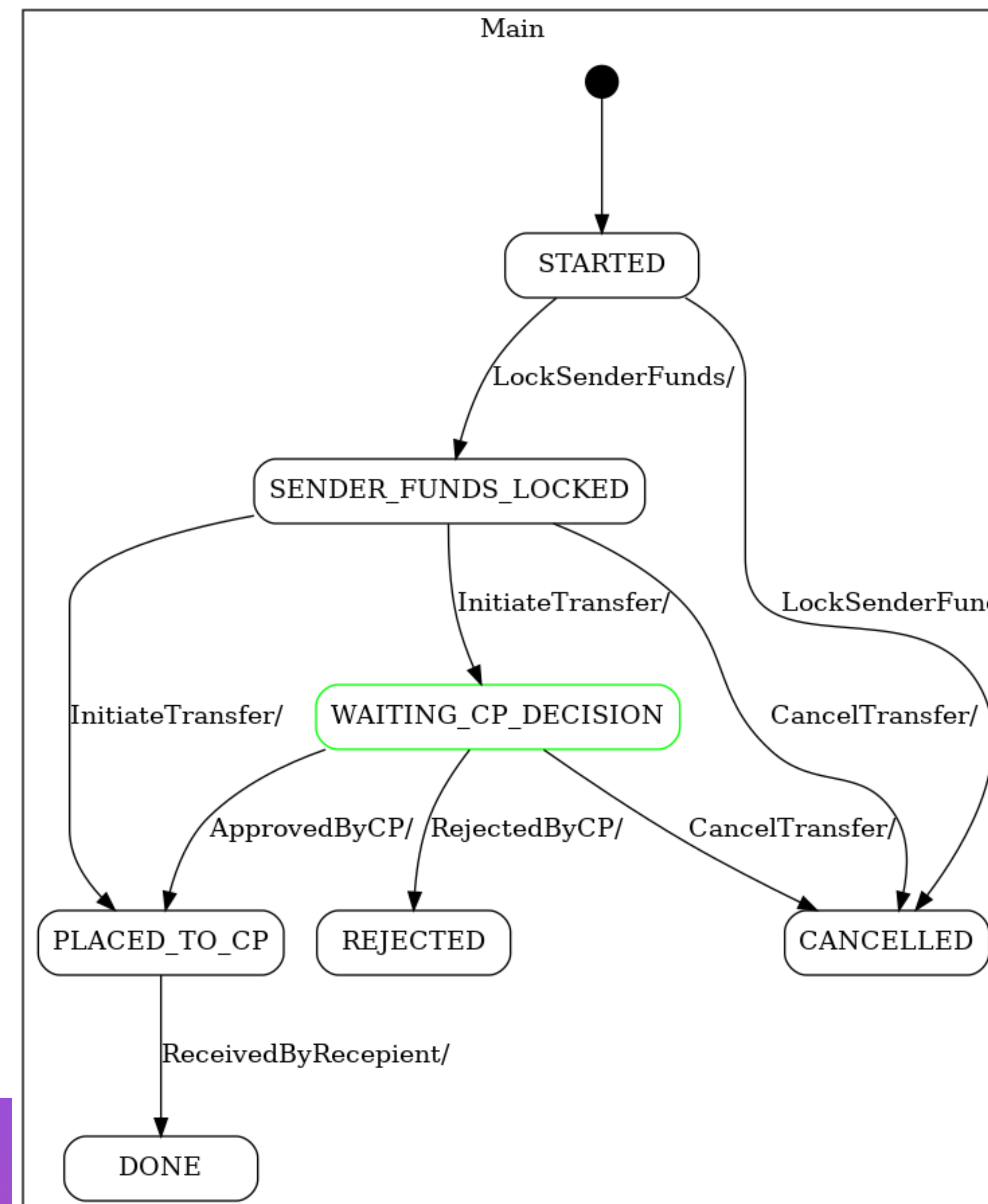
Lets dive deeper: Signals, and Queries

```kotlin
@WorkflowInterface
interface WFAssetsTransfer:VisualWF {
    Konstantin Ignatyev
    @WorkflowMethod
    fun transferAssets(senderId:String, recipientId:String, ein:String, quantity:Long)
    Konstantin Ignatyev
    @SignalMethod
    fun LockSenderFunds()
    Konstantin Ignatyev
    @SignalMethod
    fun InitiateTransfer()
    Konstantin Ignatyev
    @SignalMethod
    fun ApprovedByCP()
```

```kotlin
interface VisualWF {
    Konstantin Ignatyev
    @QueryMethod
    fun currentStateName():String
    Konstantin Ignatyev
    @QueryMethod
    fun describeWorkflowInDot():String
    Konstantin Ignatyev
    @QueryMethod
    fun getFSMDefinition():String
}
```





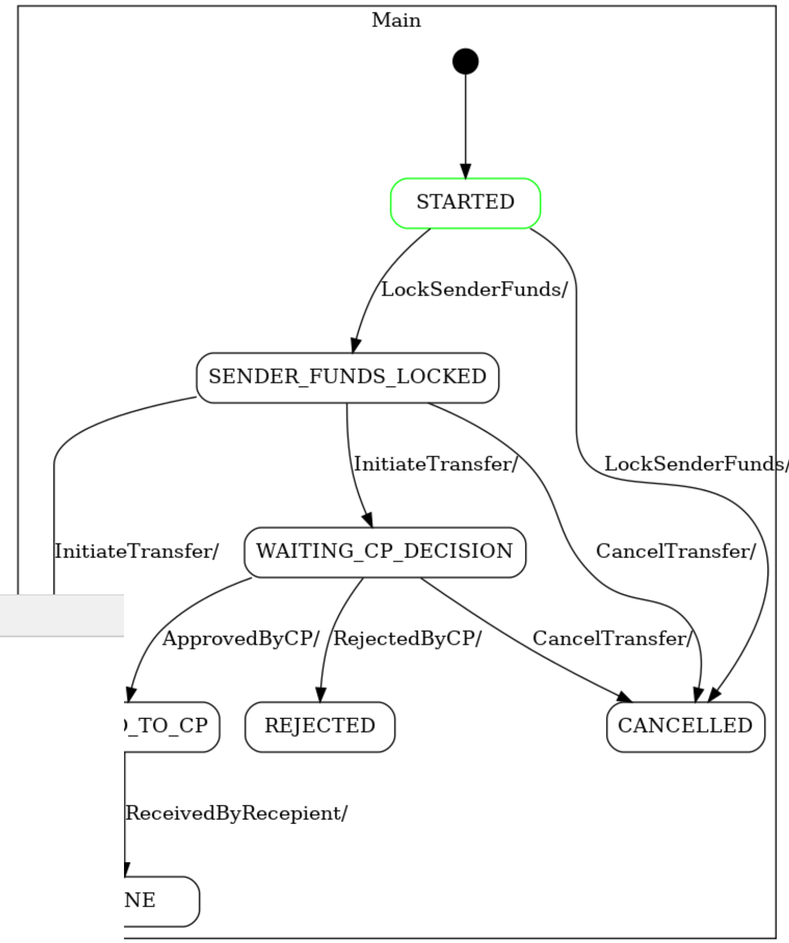


Start using Kotlin today!

FSM definitions with SMC

TransferAssets.sm

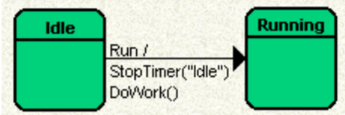
```
%map Main %%
    STARTED {
        LockSenderFunds [ ctxt.hasSufficientFunds()]
        SENDER_FUNDS_LOCKED {

        }
        LockSenderFunds [ ! ctxt.hasSufficientFunds()]
        CANCELLED {}
    }

    SENDER_FUNDS_LOCKED
      Entry {doSenderFundsLock();}
    {
        CancelTransfer
        CANCELLED {}

        InitiateTransfer [ ctxt.needsCPapproval()  ]
        WAITING_CP_DECISION {
        }

        InitiateTransfer [ !ctxt.needsCPapproval() ]
        PLACED_TO_CP {
        }
    }

    WAITING_CP_DECISION
       Entry {doRequestReceiverApproval();}
```



wf-description.dot

```
digraph TransferAssets.sm {

    node
        [shape=Mrecord width=1.5];

subgraph cluster_Main {

    label="Main";

    //
    // States (Nodes)
    //

    "Main::STARTED"
        [label="{STARTED}"];

    "Main::SENDER_FUNDS_LOCKED"
        [label="{SENDER_FUNDS_LOCKED}"];

    "Main::WAITING_CP_DECISION"
        [label="{WAITING_CP_DECISION}"];

    "Main::PLACED_TO_CP"
        [label="{PLACED_TO_CP}"];

    "Main::REJECTED"
        [label="{REJECTED}"];
```

A Transition with Actions

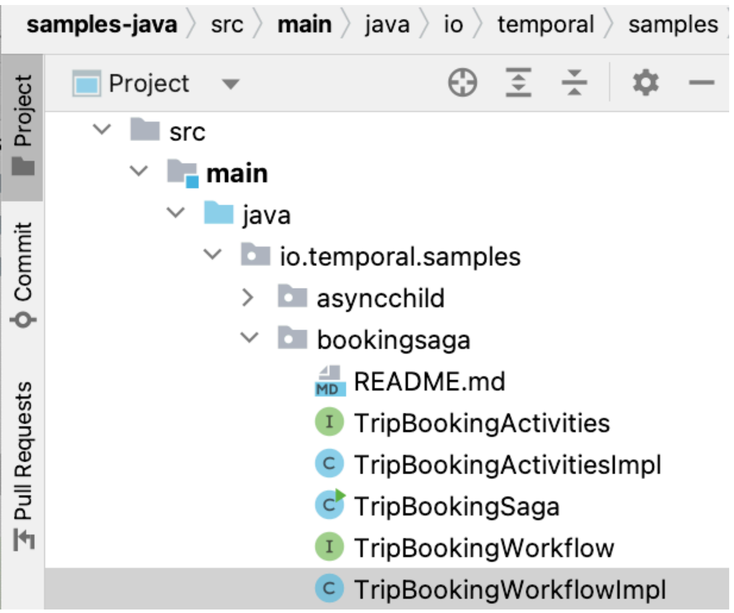



```
// State
Idle
{
    // Trans
    Run
        // Next State
        Running
        // Actions
        {
            StopTimer("Idle");
            DoWork();
        }
}
```

https://smc.sourceforge.net/SmcManual.htm

Yes, Temporal.io has it! (What was the question btw.?)

Saga pattern support - executing compensation actions on failure



```java
@WorkflowInterface
public interface TripBookingWorkflow {
  6 usages  1 implementation  Maxim Fateev +1
  @WorkflowMethod
  void bookTrip(String name);
}
```

```java
public void bookTrip(String name) {
  // Configure SAGA to run compensation activities in parallel
  Saga.Options sagaOptions = new Saga.Options.Builder().setParallelCompensation(true).build();
  Saga saga = new Saga(sagaOptions);
  try {
    String carReservationID = activities.reserveCar(name);
    saga.addCompensation(activities::cancelCar, carReservationID, name);

    String hotelReservationID = activities.bookHotel(name);
    saga.addCompensation(activities::cancelHotel, hotelReservationID, name);

    String flightReservationID = activities.bookFlight(name);
    saga.addCompensation(activities::cancelFlight, flightReservationID, name);
  } catch (ActivityFailure e) {
    saga.compensate();
    throw e;
  }
}
```

Encryption at rest

Forever Workflow (continue as new)

```java
// Request that the new child workflow run is invoked
PollingChildWorkflow continueAsNew = Workflow.newContinueAsNewStub(PollingChildWorkflow.class);
continueAsNew.exec(pollingIntervalInSeconds);
// unreachable
return null;
}
```

Timer Support

```java
public class DynamicSleepWorkflowImpl implements DynamicSleepWorkflow {

  3 usages
  private UpdatableTimer timer = new UpdatableTimer();

  Maxim Fateev
  @Override
  public void execute(long wakeUpTime) { timer.sleepUntil(wakeUpTime); }
```

WF Versioning

https://docs.temporal.io/go/versioning

https://github.com/kgignatyev/wf-simulators-4-presentation
https://github.com/kgignatyev/wf-subscription-management
https://github.com/kgignatyev/wf-assets-transfer
https://github.com/kgignatyev/fsm-with-smc
https://github.com/kgignatyev/wf-assets-transfer
https://github.com/kgignatyev/wf-subscription-management
https://github.com/kgignatyev/wf-user-management

Activity Timeouts

Most Important
+
Generally Useful
Timeouts

ScheduleToClose

Max overall execution time
INCLUDING retries

StartToClose

Max time of single invocation
ALWAYS set this!

For Long Running
Activities

Heartbeat

Max time between Heartbeats
Retry faster if worker crashes

For Queue Timeouts
& Task Routing

ScheduleToStart

Max time to stay in queue
Rarely needed! Not retryable!
Monitor this + set alerts

Temporal has four timeouts — two that are commonly used, and two that are useful only in specific cases:

- Schedule-To-Close: Limits the maximum execution time including retries.
- Start-To-Close: Limits the maximum execution time of a single execution. **We recommend ALWAYS setting this!**
- Heartbeat: Limits the maximum time between Heartbeats. *For long running Activities*, enables a quicker response when a Heartbeat fails to be recorded.
- Schedule-To-Start: Limits the maximum time that an Activity Task can sit in a Task Queue. Mainly to identify whether a Worker is down or for Task routing. **This is rarely needed!**

https://temporal.io/blog/activity-timeouts

Unit testing workflows with multi day timeouts

WFSubscriptionManagementImpl

```
29  public void subscribeForService(SubscriptionRequest sr) {
30      subscriptionStatus = SubscriptionStatus.TRIAL;
31      nextPaymentDue = new Date(Workflow.currentTimeMillis() + daysAsMillis(10));
32      Workflow.await(Duration.ofMillis(daysAsMillis(11)), () -> cancellationRequested || (paymentInfo != null));
33      if (!cancellationRequested && paymentInfo == null) {
34          userNotificationActivity.notifyUser(new UserNotification()
35                  .setUserId(sr.userId)
36                  .setMessage("Trial expired, please pay"));
37      }
38  }
```

! Always use workflow time !

WFSubscriptionManagementTest

```
54      @Before
55      public void setUp() {
56          testEnv = TestWorkflowEnvironment.newInstance();
57          wfWorker = testEnv.newWorker(WFSubscriptionManagement.WF_TASKS_QUEUE);
58          wfWorker.registerWorkflowImplementationTypes(WFSubscriptionManagementImpl.class);
59          notificationWorker = testEnv.newWorker( taskQueue: "USER_NOTIFICATION_ACT");
60          notificationWorker.registerActivitiesImplementations(notificationActivityTest);
61          testEnv.start();
```

```
87  testEnv.sleep(Duration.ofDays(6));
88  ListClosedWorkflowExecutionsResponse closedWorkflowsList = svc.listClosedWorkflowExecutions( ListClosedWorkflowExecutionsRequest.newBuilder()
89          .setNamespace( wfNamespace)
90          .setExecutionFilter( WorkflowExecutionFilter.newBuilder().setWorkflowId( wfID).build())
91          .build());
92  assertEquals( message: "we should not have finished workflows yet", expected: 0,closedWorkflowsList.getExecutionsCount());
93  testEnv.sleep(Duration.ofDays(6));
94
95  Awaitility.await( alias: "workflow "+wfID+" finished").atMost(Duration.ofSeconds(10)).pollInterval(Duration.ofSeconds(1)).until(() -> {
96      try {
97          ListClosedWorkflowExecutionsResponse closedWorkflows = svc.listClosedWorkflowExecutions( ListClosedWorkflowExecutionsRequest.newBuilder()
98                  .setNamespace( wfNamespace)
99                  .setExecutionFilter( WorkflowExecutionFilter.newBuilder().setWorkflowId( wfID).build())
100                 .build());
101         System.out.println("# closed workflows = " + closedWorkflows.getExecutionsCount());
102         return closedWorkflows.getExecutionsCount()==1 ;
103     } catch (Exception e) {
104         e.printStackTrace();
105     }
106     return false;
107 });
108 assertEquals( expected: 1, notificationActivityTest.receivedNotifications.size());
```

Workflow continuity (WF history cannot grow indefinitely)

Status	Workflow ID
Completed	1583375e-e54c-4ede-8e74-43b7513d0a0b
Cont'd as New	1583375e-e54c-4ede-8e74-43b7513d0a0b
Cont'd as New	1583375e-e54c-4ede-8e74-43b7513d0a0b

WFSubscriptionManagementImpl.

```
31      @Override
32      public void subscribeForService(SubscriptionRequest sr, PaymentInfo pi) {
49          int paymentsCounter = 0;
50          while( !cancellationRequested ){
51              System.out.println("paymentsCounter = " + paymentsCounter);
52              serviceManagementActivity.chargeServiceFee( paymentInfo );
53              //let's charge every 5 seconds to demonstrate continuity
54              Workflow.await(Duration.ofMillis(5000),()->false );
55              paymentsCounter++;
56              if( paymentsCounter == 5 ){
57                  Workflow.continueAsNew(
58                          options: null,//important! means we do not override current settings
59                          sr, paymentInfo );
60              }
61          }
62          if( cancellationRequested ){
63              serviceManagementActivity.cancelSubscription( sr );
64          }
```

ContinuityDemoRunner.java

```
29          //let's start workflow that uses continuation
30          WFSubscriptionManagement wfStub = client.newWorkflowStub(WFSubscriptionManagement.class,
31                  WorkflowOptions.newBuilder()
32                          .setTaskQueue(QUEUE_WS_TASKS)
33                          .build()
34          );
35          WorkflowExecution wfExecution = WorkflowClient.start(wfStub::subscribeForService,
36                  new SubscriptionRequest().setProduct("A").setUserId("user1"),
37                  new PaymentInfo().setAmountCents(1).setCreditCardNumber("123456"));
38          System.out.println("started wfExecution = " + wfExecution.getWorkflowId());
39
40          Thread.sleep( millis: 55000); //we should see 3 instances of the workflow with different run IDs
41          wfStub.cancel();
42          Thread.sleep( millis: 5000);
```