

# Secure Asset Transfer Protocol (SATP)

Implementation in the Hyperledger Cacti Interoperability Framework

(draft-ietf-satp-core-02)

&

(draft-belchior-satp-gateway-recovery-00)

Zakwan Jaroucheh (Edinburgh Napier University)

Mentors: V. Ramakrishna (IBM), Rafael Belchior (Técnico Lisboa), Sandeep Nishad (IBM)

Prague, Czech Republic

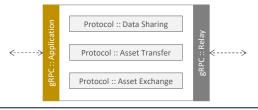
November 9, 2023



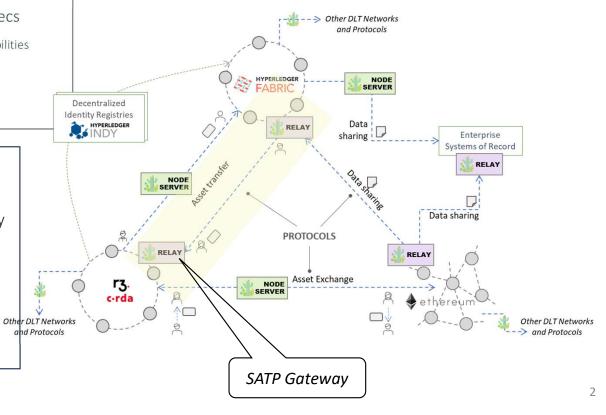
# Reference Implementation of SATP in Hyperledger Cacti



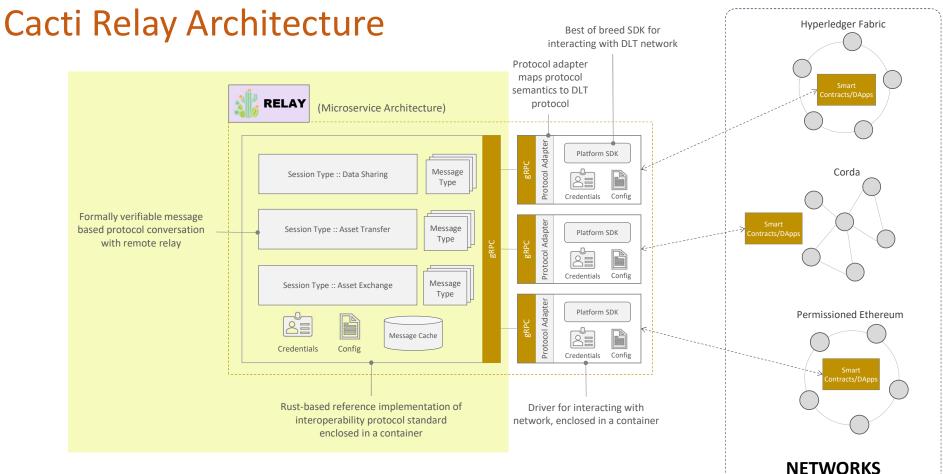
- https://wiki.hyperledger.org/display/INTERN/Cacti%3A+Implement+Standardized+Secure+Asset+Transfer+Protocol
- Augment Cacti "relay" according to SATP draft specs
  - SATP-standard endpoints and SATP message parsing capabilities
  - Error handling and crash recovery support
- ETA: end of November 2023
- Relay is a configurable module running gRPC services built on Rust
  - Not built for any specific DLT; compatible with any
  - Fits the specification for an SATP gateway









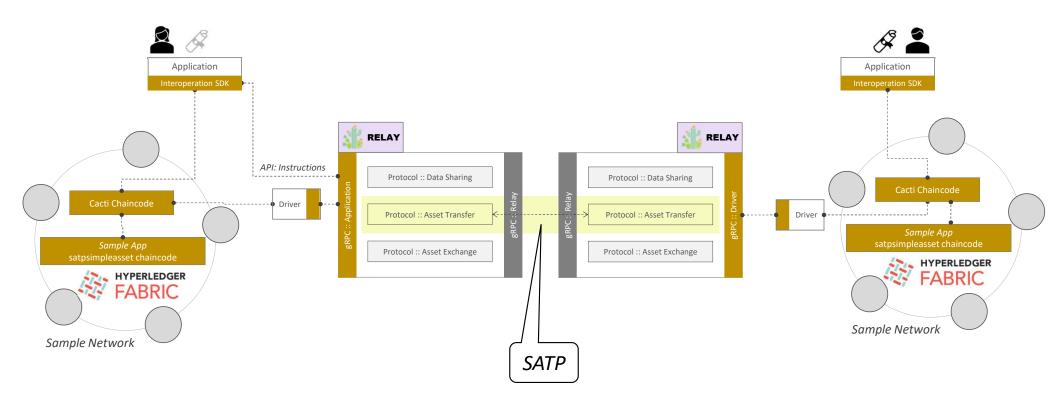


Spec: https://github.com/hyperledger/cacti/blob/main/weaver/rfcs/models/infrastructure/relays.md

Code: <a href="https://github.com/hyperledger/cacti/tree/main/weaver/core/relay">https://github.com/hyperledger/cacti/tree/main/weaver/core/relay</a>



# SATP Between Cacti-Augmented Networks



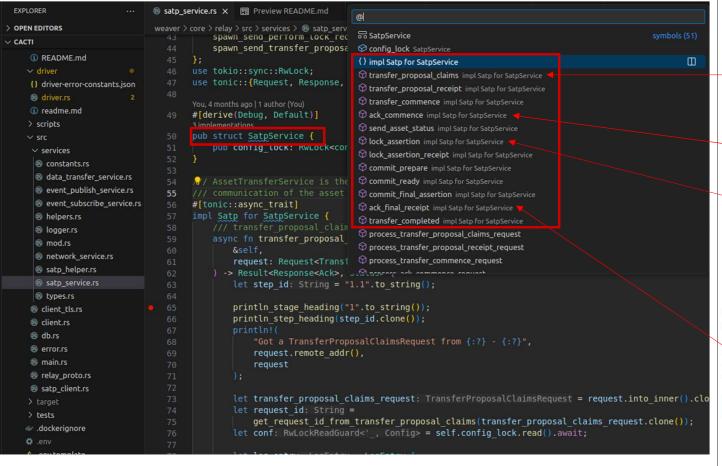


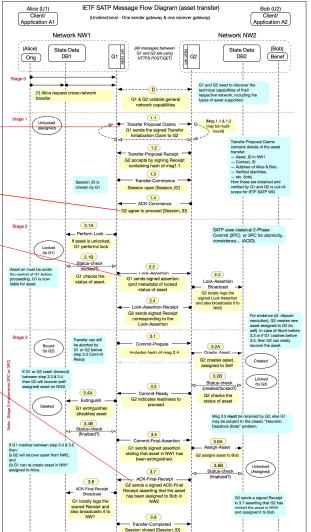
#### Augmenting Cacti Components for SATP

- Cacti Relay
  - Rust-based and runs gRPC services and clients
  - Added a SATP Service
    - Added the new service satp service.rs
    - Added a new client satp client.rs
    - Added/Changed the relevant library helper files
- SATP protobuf (service interface)
  - In github.com/hyperledger/cacti/weaver/common/protos, added the satp.proto
- Hyperledger Fabric App
  - In github.com/hyperledger/cacti/weaver/samples/fabric, added satpsimpleasset chaincode (satpsimpleasset.go), i.e., smart contract for Fabric
  - Augmented interoperation helper SDK for Fabric clients github.com/hyperledger/cacti/weaver/core/fabric-driver/server.ts to include the following functions: performLock, createAsset, extinguish, and assignAsset
- Reference: <a href="https://github.com/hyperledger/cacti/pull/2748">https://github.com/hyperledger/cacti/pull/2748</a>



# Relay Support for SATP Operations





Reference: draft-ietf-satp-core-02

IETF 118: SATP Working Group

6



#### SATP Protobuf (Service Interface for Relay)

```
service SATP {
rpc TransferProposalClaims(TransferProposalClaimsRequest) returns (common.ack.Ack) {};
rpc TransferProposalReceipt(TransferProposalReceiptRequest) returns (common.ack.Ack) {};
rpc TransferCommence(TransferCommenceRequest) returns (common.ack.Ack) {};
rpc AckCommence(AckCommenceRequest) returns (common.ack.Ack) {};
 rpc SendAssetStatus(SendAssetStatusRequest) returns (common.ack.Ack) {};
// declaring that the asset in question has been locked or escrowed by the sender gateway in
 rpc LockAssertion(LockAssertionRequest) returns (common.ack.Ack) {};
// The receiver gateway sends a LockAssertionReceipt request to the sender gateway to indicate acceptance
 rpc LockAssertionReceipt(LockAssertionReceiptRequest) returns (common.ack.Ack) {};
 rpc CommitPrepare(CommitPrepareRequest) returns (common.ack.Ack) {};
rpc CommitReady(CommitReadyRequest) returns (common.ack.Ack) {};
 rpc CommitFinalAssertion(CommitFinalAssertionRequest) returns (common.ack.Ack) {}:
 rpc AckFinalReceipt(AckFinalReceiptRequest) returns (common.ack.Ack) {};
 rpc TransferCompleted(TransferCompletedRequest) returns (common.ack.Ack) {};
```



### **SATP Service Sample Function**

```
Stage 1
Step 1.1
```

```
/// transf<mark>er proposal claims is run on the receiver gateway to allow the sender gateway to initiate an asset transfer.</mark>
async fn transfer proposal claims(
    request: Request<TransferProposalClaimsRequest>,
) -> Result<Response<Ack>, Status> {
    let transfer proposal claims request: TransferProposalClaimsRequest = request.into inner().clone();
    let request id: String =
        get request id from transfer proposal claims(transfer proposal claims request.clone());
    let conf: RwLockReadGuard<' , Config> = self.config lock.read().await;
    let log entry: LogEntry = LogEntry { --
    log::debug!("{}", log entry);
    match process transfer proposal claims request(
        transfer proposal claims request.clone(),
        conf.clone(),
            let reply: Result<Response<Ack>, Status> = Ok(Response::new(ack));
            let log entry: LogEntry = LogEntry { --
            log::debug!("{}", log_entry);
            reply
        Err(e: Error) => {
            let error message: String = "Transfer proposal claims failed.".to string();
            let reply: Result<Response<Ack>, Status> = create ack error message(request id.clone(), error message.clone(), e);
            let log entry: LogEntry = LogEntry {--
            log::error!("{}", log_entry);
            reply
```



#### SATP Gateway Calls Fabric Driver

```
> scripts
                                             /// process ack commence request is invoked by the receiver gateway to ack the transfer commence request
                                             /// requested ed by the sender gateway
  ∨ src
                                             pub fn process ack commence request(
    services
                                                  ack commence request: AckCommenceRequest,
    ® constants.rs
    ® data_transfer_service.rs
                                              ) -> Result<Ack, Error> {
    ® event_publish_service.rs
                                                  let request id: String = ack commence request.session id.to string();
    @ event_subscribe_service.rs
                                                  let is valid request: bool = is valid ack commence request(ack commence request.clone());
    ® helpers.rs
                                                  // TODO some processing
    ® logger.rs
                                                  if is valid request {
    ® mod.rs
    ® network_service.rs
                                                      let perform lock request: PerformLockRequest =
    ® satp_helper.rs
                                                          create perform lock request(ack commence request);
    ® satp_service.rs
    ® types.rs
                                                      let log entry: LogEntry = LogEntry { -

    client tls.rs

                                                      log::debug!("{}", log_entry);
   ® client.rs
   ® db.rs
                                                      match send perform lock request(perform lock request.clone(), conf) {
   ® error.rs
                                                          Ok(ack: ACK) => {
   ® main.rs
                                                               println!("Ack ack commence request.");
   relay_proto.rs
                                                               let reply: Result<Ack, Error> = Ok(ack);
                                                               println!("Sending back Ack: {:?}\n", reply);
   satp_client.rs
  > tests
  .dockerignore
  .env
                                                  } else {
  $ .env.template
  $ .env.template.2
                                                      return Ok(Ack {
OUTLINE
                                                          request id: request id.to string(),
TIMELINE
                                                          message: "Error: The ack commence request is invalid".to string(),
SAVED COMMANDS
 env cacti
```

Step 1.1 → Step 2.1A



#### SATP Gateway Calls Fabric Driver

Stage 2
Step 2.1A

```
@ data_transfer_service.rs
                                           pub fn spawn send perform lock request(
  ® event_publish_service.rs
                                               driver info: Driver,
  ® event_subscribe_service.rs
                                               perform lock request: PerformLockRequest,
  ® helpers.rs
                                               tokio::spawn(future:async move {
  logger.rs
                                                    let request_id: String = perform_lock_request.session_id.to_string();
  @ mod.rs
  network_service.rs
                                                        "Locking the asset of the lock assertion request id: {:?}",
  satp_helper.rs
                                                        request id
  satp_service.rs
  ® types.rs
 ® client_tls.rs
                                                    let result: Result<(), Error> = call_perform_lock(driver_info, perform_lock_request).await;
 ® client.rs
                                                    match result {
 ® db.rs
 ® error.rs
                                                            println!("Perform lock request sent to driver\n")
 ® main.rs
 R relay_proto.rs
                                                            println!("Error sending perform lock request to driver: {:?}\n", e);
 ® satp_client.rs
                                                             // TODO: what to do in this case?
> tests
.dockerignore
.env
$ .env.template
```



### SATP Gateway Calls Fabric Driver

Stage 2
Step 2.1A

```
    data_transfer_service.rs

                                          async fn call perform lock(
 @ event_publish_service.rs
                                               driver info: Driver,
 @ event_subscribe_service.rs
                                               perform lock request: PerformLockRequest,
 ® helpers.rs
                                               let client: DriverCommunicationClient<Channel> = get_driver_client(driver_info).await?;
 ® logger.rs
                                               println!("Sending request to driver to lock the asset
 ® mod.rs
                                               let ack: Ack = client DriverCommunicationClient<Channel>
 ® network service.rs
                                                   .clone() DriverCommunicationClient<Channel>
 satp_helper.rs
                                                  .perform lock(perform lock request) im | Future<Output = Result<Response<Ack>, Status>>
 ® satp_service.rs
                                                   .into inner();
 B types.rs
                                               println!("Response ACK from driver to perform lock {:?}\n", ack);
® client tls.rs
                                               let status: Status = ack::Status::from i32(ack.status) Option<Status>
® client.rs
                                                    .ok or(err: Error::Simple("Status from Driver error".to string()))?;
® db.rs
                                              match status {
® error.rs
® main.rs
® relay_proto.rs
                                                       return Ok(());

  satp_client.rs

                                                   ack::Status::Error => Err(Error::Simple(format!("Error from driver: {}", ack.message))),
> target
> tests
.dockerignore
```



12

#### Fabric Driver Invokes satpsimpleasset Chaincode

A sample/reference implementation for networks built on Hyperledger Fabric that use SATP gateways

Function to process asset received from another network

Step 3.6A

```
EXPLORER
                                                        ∞ assetmomt.go 9+ X ≡ Settings

■ Spell Checker Preferences

                                                                                                                       Preview README.md
                                                                                                                                                Dummy Relay.

    satp service.rs

OPEN EDITORS
                                     weaver > samples > fabric > satpsimpleasset > ∞ assetmgmt.go > ♦ (*SmartContract).AssignAsset
CACT
   relay-docker.md
                                                } else {
  VERSION
                                                     return false, logThenErrorf("claim on bond asset type %s with asset id %s failed", assetAgreement
 ① README.md
 > docs
 > resources
                                            func (s *SmartContract AssignAsset ctx contractapi.TransactionContextInterface, assetAgreementSerialized
> rfcs
                                                 assetAgreement, er .- s.amc.validateAndExtractAssetAgreement(assetAgreementSerializedProto64)
                                                if err != nil {
                                                     return false, err
  fabric
                                                // Change asset ownership to claimant You. 3 months ago * Squashed commit: first implementation
   > fabric-cli
                                                 recipientECertBase64, err := getECertOfTxCreatorBase64(ctx)
   > go-cli

    satpsimpleasset

                                                     return false, logThenErrorf(err.Error())
   gitignore
                                                 asset, err := getBondAsset(ctx, assetAgreement.AssetType, assetAgreement.Id)
   assetmgmt_test.go
                                                 if err != nil {
                                                     return false, logThenErrorf(err.Error())
   bondasset test.go
   co bondasset.go
                                                 asset.Owner = string(recipientECertBase64)
                                                 assetJSON, err := json.Marshal(asset)
    ≣ go.sum
                                                 if err != nil {
                                                     return false, logThenErrorf(err.Error())
   co helper.go
   oo main.go
                                                err = ctx.GetStub().PutState(getBondAssetKey(assetAgreement.AssetType, assetAgreement.Id), assetJSON
   M Makefile
                                                if err != nil {

≡ satpsimpleasset

                                                     return false, logThenErrorf(err.Error())
   co tokenasset test.go
   co tokenasset.go
```



#### Support for Other Kinds of Networks

- Relay is DLT-agnostic, so the SATP augmentation will work for any DLT (not just Fabric)
- Hyperledger Cacti supports connectivity to various kinds of DLTs:
   Hyperledger Fabric, Hyperledger Besu (permissioned Ethereum), Quorum,
   R3 Corda, Hyperledger Sawtooth, and others; more in the pipeline
  - Offers a connector/driver for each
    - Need to augment these to talk to the relay, just as we did for the Fabric driver in this project
  - Offers a client library/SDK for each
    - Need to add functions just like we did for the Fabric SDK in this project
  - Offers sample apps (smart contracts or DApps) for each
    - Need to add asset transfer endpoint functions (to lock, burn, mint, redeem, assets)



# Logging (Crash Recovery)

```
let log entry: LogEntry = LogEntry {
#[derive(Serialize, Deserialize)]
                                                    request id: request id.clone(),
                                                    request: serde json::to string(&transfer proposal claims request.clone()).unwrap(),
pub struct LogEntry {
                                                    step id: step id.clone(),
    pub request id: String,
                                                    operation: Operation::Init,
    pub request: String,
                                                   network id: "todo network id".to string(),
    pub step id: String,
                                                    gateway id: transfer proposal claims request TransferProposalClaimsRequest
    pub operation: Operation,
                                                        .clone()
    pub network id: String,
                                                        .sender gateway network id,
    pub gateway id: String,
                                                    received: true,
    pub received: bool,
                                                    details: None,
    pub details: Option<String>,
                                                log::debug!("{}", log entry);
```

Types: Init, Exec, Done

*Reference*: draft-belchior-satp-gateway-recovery-00



## Logging (Crash Recovery)

```
fn log(&self, record: &Record) {
   if self.enabled(record.metadata()) {
      let conn: MutexGuard<' , Connection> = self.conn.lock().unwrap();
      if let Ok(log_entry: LogEntry) = serde_json::from_str::<LogEntry>(&record.args().to_string()) {
          let mut details: String = "".to string();
          if let Some(d: &String) = log entry.details.as ref() {
               details = d.to_string();
          conn.execute(
              sql: "INSERT INTO log entries (debug_level, timestamp, request id, request, step id, operation, gateway id, received, details) VALUES
              params: &[
                  &record.level().to_string(),
                  &log_entry.request_id,
                  &log_entry.request,
                  &log entry.step id,
                  &log entry.operation.to string(),
                  &log_entry.gateway_id,
                  &log entry.received.to string(),
                  &details,
           .expect(msg: "Failed to insert log entry");
```

Storage Implementation: SQLite database



16

# Sample Snapshot of Log Entries

	123 id 💌	asc debu 🔻	nec timestamp	RBC request id	ABC request	anc step ▼	asc operation	▼ noc gateway id	▼ asc receiv
28		DEBUG	2023-10-31 09:32:04.813467831 +00:00	session id1	{"message type":"message type1","session id":"sessi	1.4	Done	todo_gateway_id	true
29	129	DEBUG	2023-10-31 09:32:04.819427511 +00:00	session id1			Done		false
30	130	DEBUG	2023-10-31 09:44:38.261779408 +00:00	hard coded transfer proposal	{"message type":"message type1","asset asset id":"a	1.1	Init	sender gateway r	et true
31	131	DEBUG	2023-10-31 09:44:38.301532806 +00:00		{"message_type":"message_type1","asset_asset_id":"a		Exec	sender gateway r	
32	132	DEBUG	2023-10-31 09:44:38.307321484 +00:00		{"message_type":"message_type1","asset_asset_id":"a		Done	sender gateway r	et true
33	133	DEBUG	2023-10-31 09:44:38.314183193 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_r	et false
34	134	DEBUG	2023-10-31 09:44:38.326025051 +00:00	hard coded transfer proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender gateway r	et true
35	135	DEBUG	2023-10-31 09:44:38.333864947 +00:00		{"message type":"message type1","session id":"to be		Exec	todo gateway id	true
36	137	DEBUG	2023-10-31 09:44:38.364966704 +00:00	hard coded transfer proposal	{"message type":"message type1","asset asset id":"a	1.2	Done	sender gateway r	et true
37	139	DEBUG	2023-10-31 09:44:38.382240429 +00:00	to be calculated session id	{"message type":"message type1", "session id": "to be	1.3	Init	todo gateway id	true
38	140	DEBUG	2023-10-31 09:44:38.394858645 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"sessi	1.3	Exec	todo_gateway_id	true
39	141	DEBUG	2023-10-31 09:44:38.413857783 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_be	1.3	Done	todo_gateway_id	true
40	142	DEBUG	2023-10-31 09:44:38.428883513 +00:00	session id1	{"message type":"message type1","session id":"sessi		Init	todo gateway id	true
11	144	DEBUG	2023-10-31 09:44:38.440126841 +00:00	session id1	{"session id":"session id1"}	1.4	Exec	todo gateway id	true
12	145	DEBUG	2023-10-31 09:44:38.445342687 +00:00	session_id1	{"message_type":"message_type1","session_id":"sessi	1.4	Done	todo gateway id	true
13	147	DEBUG	2023-11-02 12:56:08.816127280 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Init	sender gateway r	et true
14	148	DEBUG	2023-11-02 12:56:08.832983294 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Exec	sender_gateway_r	et true
15	149	DEBUG	2023-11-02 12:56:08.838496872 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Done	sender_gateway_r	et true
16	150	DEBUG	2023-11-02 12:56:08.848544677 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_r	et false
17	151	DEBUG	2023-11-02 12:56:08.865015914 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_r	et true
48	152	DEBUG	2023-11-02 12:56:08.878016932 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","session_id":"to_be	1.2	Exec	todo gateway id	true
49	153	DEBUG	2023-11-02 12:56:08.881571873 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Done	sender_gateway_r	et true
50	154	DEBUG	2023-11-02 12:56:08.889036179 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_be	1.3	Init	todo_gateway_id	true
51	155	DEBUG	2023-11-02 12:56:08.894634330 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"sessi	1.3	Exec	todo_gateway_id	true
52	156	DEBUG	2023-11-02 12:56:08.898092023 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_be	1.3	Done	todo_gateway_id	true
3	157	DEBUG	2023-11-02 12:56:08.906376850 +00:00	session_id1	{"message_type":"message_type1","session_id":"sessi	1.4	Init	todo_gateway_id	true
54	158	DEBUG	2023-11-02 12:56:08.909971998 +00:00	session_id1	{"session_id":"session_id1"}	1.4	Exec	todo_gateway_id	true
55	159	DEBUG	2023-11-02 12:56:08.913557005 +00:00	session_id1	{"message_type":"message_type1","session_id":"sessi	1.4	Done	todo_gateway_id	true
6	160	DEBUG	2023-11-03 09:05:18.728351562 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Init	sender_gateway_r	et true
7	161	DEBUG	2023-11-03 09:05:18.760018150 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Exec	sender_gateway_r	et true
8	162	DEBUG	2023-11-03 09:05:18.777034518 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.1	Done	sender_gateway_r	et true
9	163	DEBUG	2023-11-03 09:05:18.824966745 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_r	et false
0	164	DEBUG	2023-11-03 09:05:18.846549066 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Init	sender_gateway_r	et true
51	165	DEBUG	2023-11-03 09:05:18.854730148 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","session_id":"to_be	1.2	Exec	todo_gateway_id	true
52	166	DEBUG	2023-11-03 09:05:18.869134800 +00:00	hard_coded_transfer_proposal	{"message_type":"message_type1","asset_asset_id":"a	1.2	Done	sender_gateway_r	et true
53	167	DEBUG	2023-11-03 09:05:18.904405135 +00:00	to_be_calculated_session_id	{"message_type":"message_type1","session_id":"to_be	1.3	Init	todo_gateway_id	true
54	168	DEBUG	2023-11-03 09:05:18.913498099 +00:00	to be calculated session id	{"message type":"message type1","session id":"sessi	1.3	Exec	todo gateway id	true

Storage Implementation: SQLite database



#### **Project Status**

#### SATP Service

- All endpoints for Stage 1 to Stage 3 have been implemented
- An endpoint has been added to allow the driver to update the status of an asset for the steps: 2.1B, 3.2B, 3.4B and 3.6B.
- Placeholders have been added to validate each incoming requests
- Placeholders have been added to derive the corresponding request object (that needs to be sent to the other gateway) based on the incoming requests
- Demo how an asset can be transferred from one Fabric network to another Farbic network using SATP protocol implementation

#### TODO

- All endpoints related to Stage 0
- Fill the above placeholders according to the SATP logic
- Remove some hardcoded values used for demoing creating an asset, assign it and destroy it
- Add a looping mechanism that enables the gateway to repeatedly inquire about the asset's status from the driver.

17



#### Feedback from Implementer

- It would be beneficial to have the format of these messages documented in a Git repository to serve as the definitive source of truth.
- Interface between gateway and driver (executor) is not clear (*Note*: out of scope)
- Unclear how to get a unique ID for each request
- How do gateways negotiate compatible signature algorithms? (*Note*: Stage 0)
- Ambiguity about asset state inference in the face of failure
  - If a gateway recovers after a crash and wishes to resume SATP but discovers that an asset is locked, how does it know that the asset was locked by the in-progress SATP instance and not by an unrelated process?
  - *Note*: do we need a generic interface for networks to expose the states of digital assets to third parties?
- Managing contention and avoid overhead: read asset state first before attempting to lock (atomic operation)

ΤŞ



#### Thank you and Q&A

Zakwan Jaroucheh: <a href="mailto:zakwanj@gmail.com">zakwanj@gmail.com</a>

Rafael Belchior: <a href="mailto:rafael.belchior@tecnico.ulisboa.pt">rafael.belchior@tecnico.ulisboa.pt</a>

V. Ramakrishna: <u>vramakr2@in.ibm.com</u>

Sandeep Nishad: <a href="mailto:sandeep.nishad1@ibm.com">sandeep.nishad1@ibm.com</a>

Zakwan's demo with voiceover: <a href="https://we.tl/t-dWH2vFeNt4">https://we.tl/t-dWH2vFeNt4</a>