



Module 10

Plugins and APIs

The Corda logo, featuring the word 'corda' in white lowercase letters on a red rectangular background.

corda



Learning outcomes

- Understand how nodes are customized using plugins
- Learn how to register an API
- Learn how to write your own node API



Node plugins

- Plugins extend the node to offer APIs and static web content
- The nodes registers any plugins it wishes to use in its [WebServerPluginRegistry](#)
- These plugins contain two things:
 - Web APIs hosted by the node
 - Any associated static web content

WebServerPluginRegistry

- Corda web plugins subclass **CordaPluginRegistry**:

```
interface WebServerPluginRegistry {
```

```
    val webApis get() = emptyList()
```

```
    val staticServeDirs get() = emptyMap()
```

```
}
```

Registering APIs

- The syntax to register an API in the plugin is as follows:

```
override val webApis: List<Function<CordaRPCOps, out Any>>
    = listOf(
        Function(::ExampleApi1),
        Function(::ExampleApi2)
    )
```

- The API itself is defined using Java's **JAX-RS**:

```
@Path("example")
class ExampleApi(val services: CordaRPCOps)

...
```

Registering static web content

- The syntax to register static web content in the plugin is as follows:

```
override val staticServeDirs: Map<String, String>
    = mapOf(
        "example" to javaClass
                                .classLoader
                                .getResource("exampleWeb")
                                .toExternalForm()
    )
```

- The static web content is placed in the resources directory, in a folder with the same name as the string passed to **getResource** above

Using plugins on a node

- Each CorDapp contains a *src/main/resources/META-INF/services* folder containing a file called [net.corda.webserver.services.WebServerPluginRegistry](#)
- The node will only load plugins if their fully-qualified class name is listed in this file:

```
# Register a ServiceLoader service extending from  
net.corda.webserver.services.WebServerPluginRegistry
```

```
com.example.plugin.ExamplePlugin
```



Plugins in summary

- Plugins extend the node to offer:
 - Web APIs
 - Static web content
- New plugins must be registered in the node's [WebServerPluginRegistry](#)



r3

Practical

Building the API

- Our API will have two endpoints to start with:
 - A **GET** endpoint listing the states in the node's vault
 - A **PUT** endpoint enabling us to issue IOUs

1. CorDapp Design

2. State

3. Contract

4. Flow

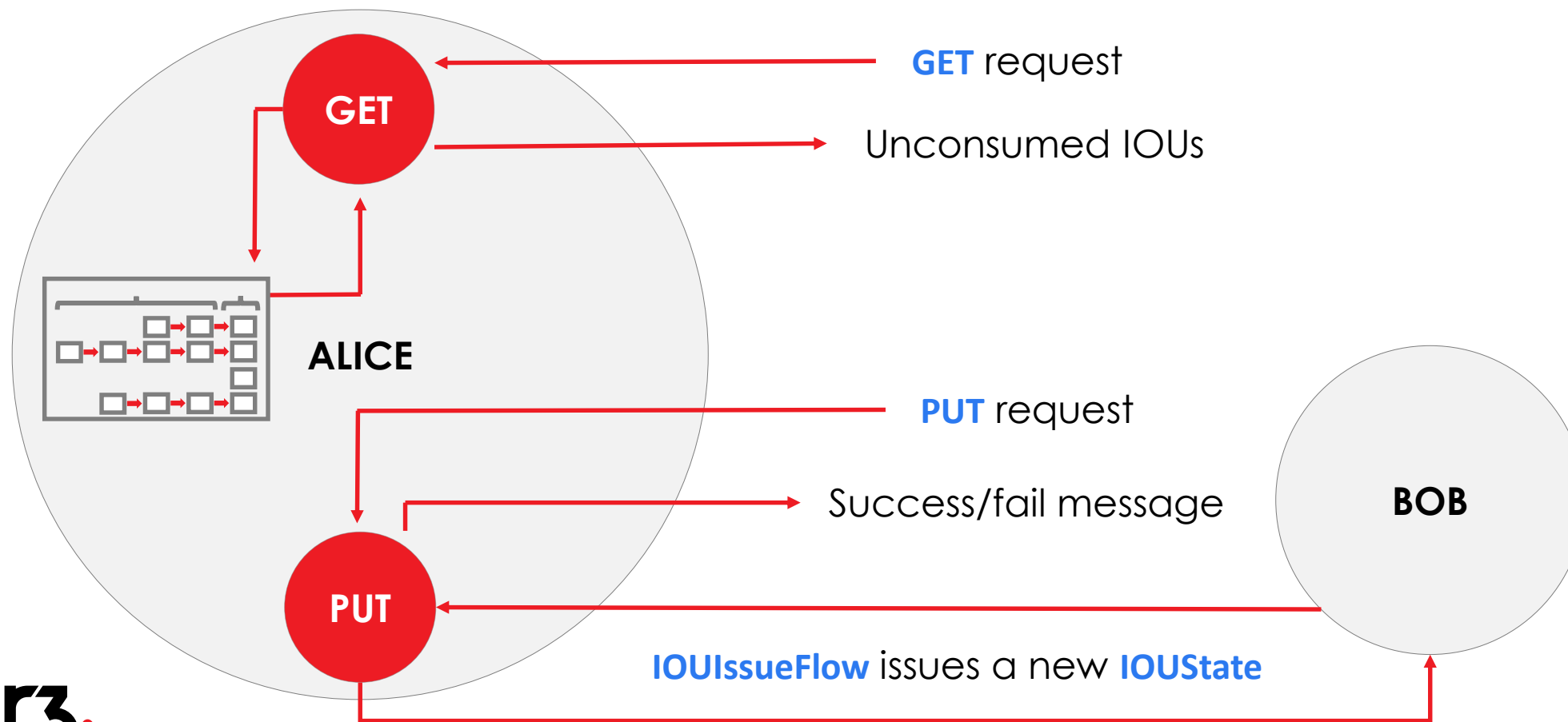
5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint



IOU API Diagram



r3.

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

A low-angle, black and white photograph of a modern skyscraper with a complex, geometric facade, viewed through a grid of small dots. The building's structure is composed of many sharp angles and lines, creating a sense of height and architectural complexity. The grid of dots is a light gray, semi-transparent overlay that covers the entire image, adding a technical or digital feel to the composition.

Step 1 – Testing Endpoints

Testing GET Endpoints with Postman

- We will be implementing a **GET** endpoint on the path “ious” to retrieve the node’s vault states
- We’ll use Postman to hit the endpoints
 - Download instructions: <https://www.getpostman.com/>
 - Installation instructions:
https://www.getpostman.com/docs/install_native
- Making **GET** requests with Postman:
 - Change the verb to “GET” in the top-left dropdown
 - Enter the URL for the first node’s endpoint,
<http://localhost:10005/api/iou/ious>
 - Press “Send”

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- **Testing endpoints**
- Get-IOUs endpoint
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

Testing Get-IOUs Endpoint - Instructions



r3.

Goal	Test the GET endpoint
Where?	The command line
Steps	<ol style="list-style-type: none">1. Deploy the nodes:<ul style="list-style-type: none">• Unix: ./gradlew deployNodes• Windows: gradlew deployNodes2. Run the nodes:<ul style="list-style-type: none">• Unix: sh build/nodes/runnodes• Windows: build\nodes\runnodes3. Use Postman to test the endpoint4. The endpoint will return a 404 “Not Found” error
Key Docs	https://www.getpostman.com/docs/requests

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- **Testing endpoints**
- Get-IOUs endpoint
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint



Step 2 – The Get-IOUs Endpoint

GET Endpoint Syntax

- The node's API is defined in JAX-RS, the Java API for RESTful web services
- We define a **GET** endpoint as follows:

```
@GET
@Path("exampleGetEndpoint")
@Produces(MediaType.APPLICATION_JSON)
fun exampleGETEndpoint() {
    return Response.accepted()
        .entity("Example GET endpoint.")
        .build();
}
```

- Where:
 - **@GET** specifies the endpoint's type
 - **@Path** specifies the endpoint's relative path
 - **@Produces** specifies the endpoint's return type



- Testing endpoints
- **Get-IOUs endpoint**
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

The Get IOUs Endpoint

- Our **GET** endpoint will list the states in the node's vault
- The API holds a **CordaRPCOps** object that allows us to perform actions such as retrieve transactions or start flows
- **CordaRPCOps.vaultTrackBy** returns a **DataFeed** of:
 - The states currently in the node's vault
 - An observable to monitor for future vault updates
- We are only interested in the existing states, not the future updates

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- **Get-IOUs endpoint**
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

The Get-IOUs Endpoint - Implementation



Goal	Create a GET endpoint for retrieving a node's active states
Where?	IOUAPI.Kt
Steps	1. Implement a GET endpoint on the path "ious" to retrieve a list of the node's vault states
Key Docs	https://docs.oracle.com/javaee/7/tutorial/jaxrs002.htm

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- **Get-IOUs endpoint**
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

The Get-IOUs Endpoint - Solution



Goal	Set up a GET endpoint returning the node's active states
Steps	<ul style="list-style-type: none">• Update the endpoint's path• Change the return type• Return only the first element of vaultAndUpdates
Code	<pre>@GET @Path("/ious") @Produces(MediaType.APPLICATION_JSON) fun getIOUs() = services.vaultQueryBy<IOUState>().states</pre>

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- **Get-IOUs endpoint**
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

Testing the Get-IOUs Endpoint - Instructions



r3.

Goal	Test the GET endpoint
Where?	The command line
Steps	<ol style="list-style-type: none">1. Make sure you've killed any nodes that are currently running2. Deploy the nodes:<ul style="list-style-type: none">• Unix: ./gradlew deployNodes• Windows: gradlew deployNodes2. Run the nodes:<ul style="list-style-type: none">• Unix: sh build/nodes/runnodes• Windows: build\nodes\runnodes3. Use Postman to test the endpoint4. The endpoint will return an empty list – there's nothing in the vault yet!
Key Docs	N/A

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- **Get-IOUs endpoint**
- Issue-IOUs endpoint
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint



Step 3 – The Issue-IOUs Endpoint

The Issue-IOUs Endpoint

- The ultimate goal of our new endpoint is to start the **IOUIssueFlow** and agree an IOU
- For now, the endpoint will just take the query-string params and return them as the body of an **Accepted Response** object
- The query-string params will be:
 - **amount (Int)**: the value of the IOU
 - **currency (String)**: the code of the IOU's currency
 - **party (String)**: the name of the party receiving the IOU

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- **Issue-IOUs endpoint**
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

The Issue-IOUs Endpoint - Instructions



Goal	Start building the endpoint to issue IOUs
Where?	IOUAPI.kt
Steps	<ol style="list-style-type: none">1. Implement a PUT endpoint on path "issue-iou" that:<ul style="list-style-type: none">• Takes an "amount", "Currency" and a "party" as querystring params• Returns an CREATED response to the user with these values as the response's body
Key Docs	N/A

r3.

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- **Issue-IOUs endpoint**
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

The Issue-IOUs Endpoint - Solution



r3.

Goal	Set up a dummy PUT endpoint for testing
Steps	<ul style="list-style-type: none">• Create a PUT endpoint taking three querystring params• Return the querystring params in a HTTP response
Code	<pre>@PUT @Path("issue-iou") fun issue-iou(@QueryParam(value = "amount") amount: Int, @QueryParam(value = "currency") currency: String, @QueryParam(value = "party") party: String): Response { return Response .status(Response.Status.CREATED) .entity("\$amount \$currency \$party") .build(); }</pre>

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- **Issue-IOUs endpoint**
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint

Testing the Issue-IOUs Endpoint - Instructions



r3.

Goal	Test the PUT endpoint
Where?	The command line
Steps	<ol style="list-style-type: none">1. Make sure you've killed any nodes that are currently running2. Re-deploy the nodes:<ul style="list-style-type: none">• Unix: ./gradlew deployNodes• Windows: gradlew deployNodes3. Re-run the nodes:<ul style="list-style-type: none">• Unix: sh build/nodes/runnodes• Windows: build\nodes\runnodes4. Postman to test the endpoint5. You should see your IOU value and counterparty<ul style="list-style-type: none">• e.g. 99NodeC
Key Docs	N/A

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- **Issue-IOUs endpoint**
- Issuing IOUs via API
- End-to-End Test
- ✓ Checkpoint



Step 4 – Issuing IOUs via the API

Wiring up the Issue-IOUs Endpoint

- We need to extend our Issue-IOUs endpoint to actually issue an **IOUState** onto the ledger
- For this, our endpoint must:
 1. **Gather the flow's arguments:**
 - Retrieve the identities of the node and its counterparty
 - Create the Amount object for the amount to be issued
 - Construct the **IOUState** to be issued onto the ledger
 2. **Run the flow:**
 - Start the flow
 - Return the flow's result
- We'll start with the first step

r3.

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- **Issuing IOUs via API**
- End-to-End Test
- ✓ Checkpoint

Flow Set-Up

- We retrieve the node's identity as follows:
 - `CordaRPCOps.nodeInfo().legalIdentities.first()`
- The counterparty's identity is retrieved using:
 - `CordaRPCOps.wellKnownPartyFromX500Name(CordaX500Name.parse(party))`
- The Amount object is created using:
 - `Amount(amount.toLong() * 100, Currency.getInstance(currency))`
- Then to create an **IOUState** instance with the desired attributes and send the state to the Issue Flow

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- **Issuing IOUs via API**
- End-to-End Test
- ✓ Checkpoint

Flow Set-Up - Implementation



Goal	Continue building the Issue-IOUs endpoint
Where?	IOUAPI.java
Steps	1. Write the code to retrieve the party identities and create the desired IOUState
Key Docs	N/A

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- **Issuing IOUs via API**
- End-to-End Test
- ✓ Checkpoint

Flow Set-Up - Solution



Goal	Issue the IOUState
Steps	<ul style="list-style-type: none">• Retrieve the Partys identities• Create the Amount object• Create the desired output IOUState
Code	<pre>val me = services.nodeIdentity().legalIdentity val lender = services.wellKnownPartyFromX500Name(CordaX500Name.parse(party)) val amountToIssue = Amount(amount.toLong() * 100, Currency.getInstance(currency)) val state = IOUState(amountToIssue,lender,me);</pre>

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- **Issuing IOUs via API**
- End-to-End Test
- ✓ Checkpoint

Kicking Off the Flow

- To actually start the flow, we call:

```
CordaRPCOps
    .startTrackedFlowDynamic(IOUIssueFlow::class.java, stateAndContract)
    .returnValue
    .get()
```

- This will:
 1. Return a **FlowHandle**
 2. Convert the **FlowHandle** into a **ListenableFuture**
 3. Convert the **ListenableFuture** into the on-ledger **SignedTransaction**

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- **Issuing IOUs via API**
- End-to-End Test
- ✓ Checkpoint



Returning a Response

- Finally, we return a **Created Response** object:

```
return Response
    .status(Response.Status.CREATED)
    .entity("Transaction id ${result.id} sent to counterparty.")
    .build()
```

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- **Issuing IOUs via API**
- End-to-End Test
- ✓ Checkpoint



Kicking Off the Flow - Implementation



Goal	Finalize the Issue-IOUs endpoint
Where?	IOUAPI.java
Steps	1. Wire up the endpoint to kick off the IOUIssueFlow and return a response
Key Docs	Web logs accessible at tail -f build/nodes/ParticipantA/logs/web/node-<machine_name>.local.log

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- **Issuing IOUs via API**
- End-to-End Test
- ✓ Checkpoint

Kicking Off the Flow - Solution



Goal	Run the IOUFlow and return a HTTP response to the user
Steps	<ul style="list-style-type: none">• Retrieve IOUFlow's output• Return a HTTP response with the transaction's ID
Code	<pre>val result = rpcOps.startTrackedFlow(IOUIssueFlow::, state) .returnValue .get() return Response .status(Response.Status.CREATED) .entity("Transaction id \${result.id} sent to counterparty.") .build()</pre>



Step 5 – An End-to-End Test

An End-to-End Test - Instructions



r3.

Goal	Test the finalized Issue-IOUs endpoint
Where?	The command line
Steps	<ol style="list-style-type: none">1. Make sure you've killed any nodes that are currently running2. Deploy the nodes:<ul style="list-style-type: none">• Unix: <code>./gradlew deployNodes</code>• Windows: <code>gradlew deployNodes</code>3. Run the nodes:<ul style="list-style-type: none">• Unix: <code>sh build/nodes/runnodes</code>• Windows: <code>build\nodes\runnodes</code>4. Hit the <code>Issue-IOUs</code> endpoint5. Use the <code>Get-IOUs</code> endpoint to check that the transaction has been recorded by the node
Key Docs	Postman collection with prebuilt endpoints is the template repository

1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- Issuing IOUs via API
- **End-to-End Test**
- ✓ Checkpoint

An End-toEnd Test - Output

The endpoint should return the created IOU in JSON form!

```
{
  "state": {
    "data": {
      "amount": "99.00 GBP",
      "lender": "C=US,L=New York,O=ParticipantB",
      "borrower": "C=GB,L=London,O=ParticipantA",
      "paid": "0.00 GBP",
      "linearId": {
        "externalId": null,
        "id": "2f1a3ec4-9e08-4dc3-a4b5-addd6f62bee3"
      },
      "participants": [
        "C=US,L=New York,O=ParticipantB",
        "C=GB,L=London,O=ParticipantA"
      ],
      "contract": "net.corda.training.contract.IOUContract",
      "notary": "C=GB,L=London,O=Network Map Service,CN=corda.notary.validating",
      "encumbrance": null,
      "constraint": {"attachmentId": "48D97A620C18B4B2ED9FB7B89E05837FDF3BB5F6F1C5"}
    },
    "ref": {
      "txhash": "1877F31EF01EF342CC46118105B1C23ADDAE5DA07D74DE4F5260C242CBF8C8DE",
      "index": 0
    }
  }
}
```



1. CorDapp Design

2. State

3. Contract

4. Flow

5. Network

6. API

- Testing endpoints
- Get-IOUs endpoint
- Issue-IOUs endpoint
- Issuing IOUs via API
- **End-to-End Test**
- ✓ Checkpoint

Conclusion



Conclusion

- The IOU CorDapp is now complete
- We can apply the same process to build any CorDapp:

1. State

Write the states representing your shared facts

2. Contract Design

Design the constraints on the evolution of these facts

3. Contract Code

Use LedgerDSL to develop the corresponding contract

4. Flow Design

Define the process of agreeing the shared facts

5. Flow Code

Use the Flow Testbed to develop the corresponding flow

6. User Interface

Define how you'll interact with your node:

- Via an API
- Via RPC

