

TDI v7

Tivoli Directory Integrator v7

Workshop

Directory Integration

- Meta-directory products
- The integration task
 - Extract data
 - Reformat
 - Enrich
 - Synchronise
 - Report
- Not just 'directories'

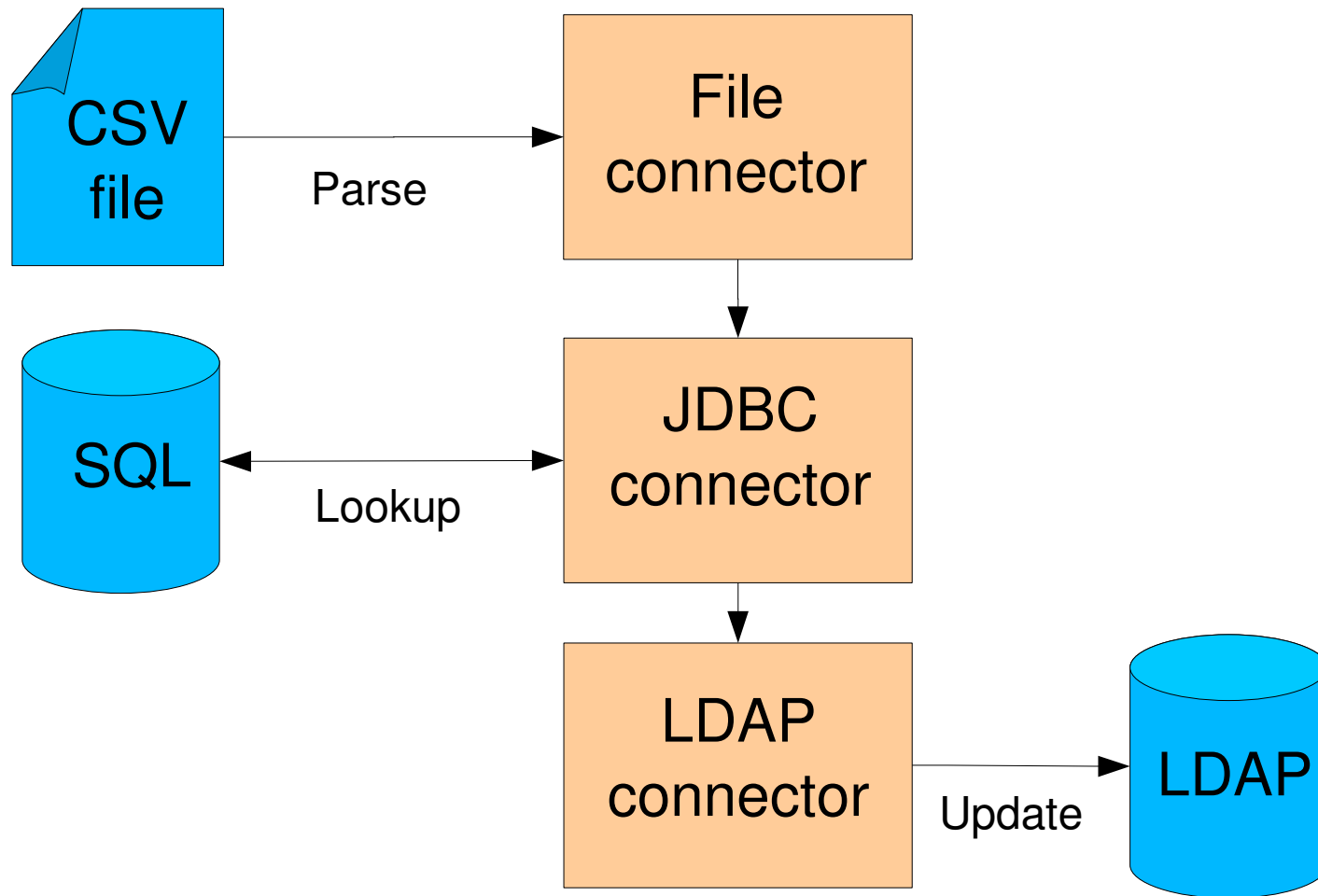
TDI Overview

- Toolkit – not 'solution' – choose your own architecture
- Data flows
 - Flow modelled by Assembly Line
 - Connectors for data stores
 - Maps and scripts for data conversion
- Multi platform (Java)
- Separate Config Editor and Runtime

TDI Data Model

- Process one entry at a time
- No fixed relationship between entries
- Entries are not named
- The Work Object – like an LDAP entry
 - Named attributes
 - Zero or more values per attribute
 - Values can be of any type

Assembly Line



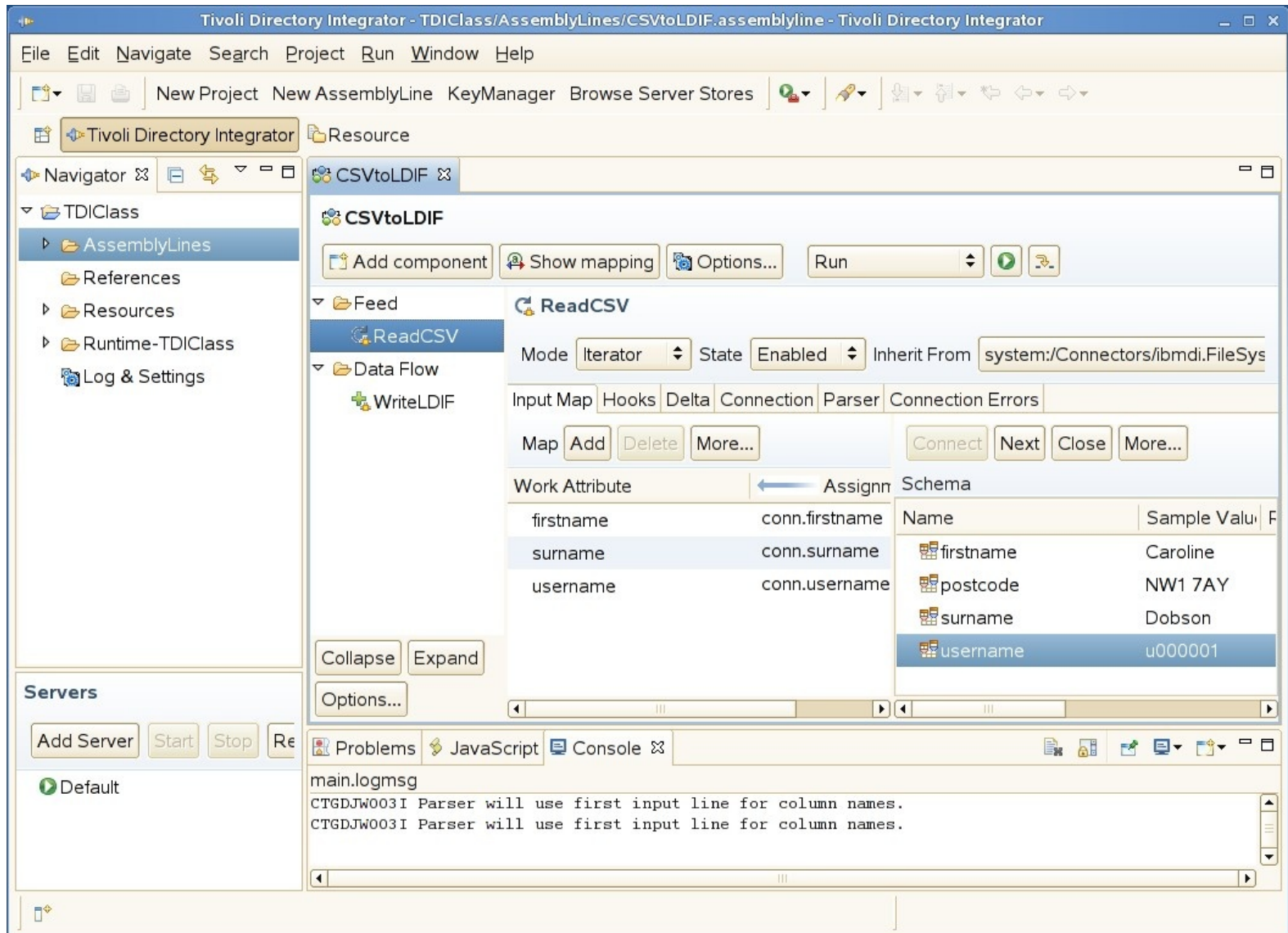
In the Toolbox

- Connectors
- Parsers
- Maps
- Scripts
- Loops
- Branches

TDI Architecture

- Configuration Editor
 - Eclipse GUI
 - Works with code repositories
 - Drag-n-drop components
 - Extendable
 - Writes XML file for runtime
- Runtime
 - Small, multi-platform, extendable

The Config Editor



Assembly Line View



- Can expand items
- Drag things up and down
- Add and delete

Show Mapping

CSVtoLDIF

Add component Show mapping Options... Run

Feed

- ReadCSV
- Data Flow
 - DBLookup
 - WriteLDIF

Map Add Delete Browse Data

Work Attribute	Assignment	Component Attribute
ReadCSV		
	←	[Source]
firstname	conn.firstname	firstname
surname	conn.surname	surname
username	conn.username	username
DBLookup		
	←	[Source]
phone	conn.phone	phone
WriteLDIF		
	→	[Target]
*	(Map all Attributes)	*

Collapse Expand Options...

Installing TDI

- Installable components
- Code and Data areas
- Solution Directory
- Workspace Directory
- Differences between OS platforms
- Extras in the Identity Edition

The workshop environment

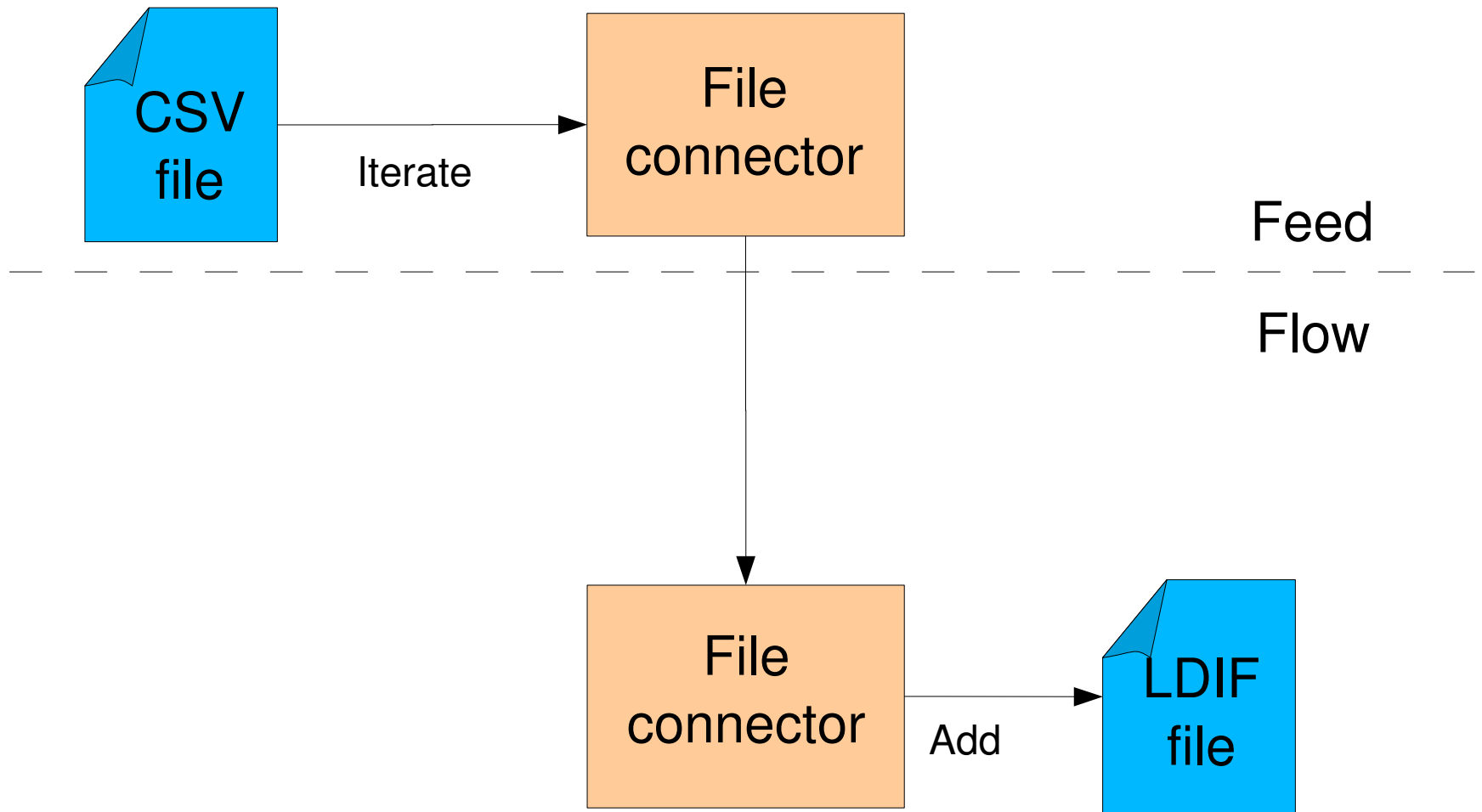
- VMWare
- SuSE Linux: SLES 10 eval
- *student* account
- TDI v7.1 Identity Edition eval
- Datafiles
- MySQL database
- LDAP servers

Exercises 1,2,3

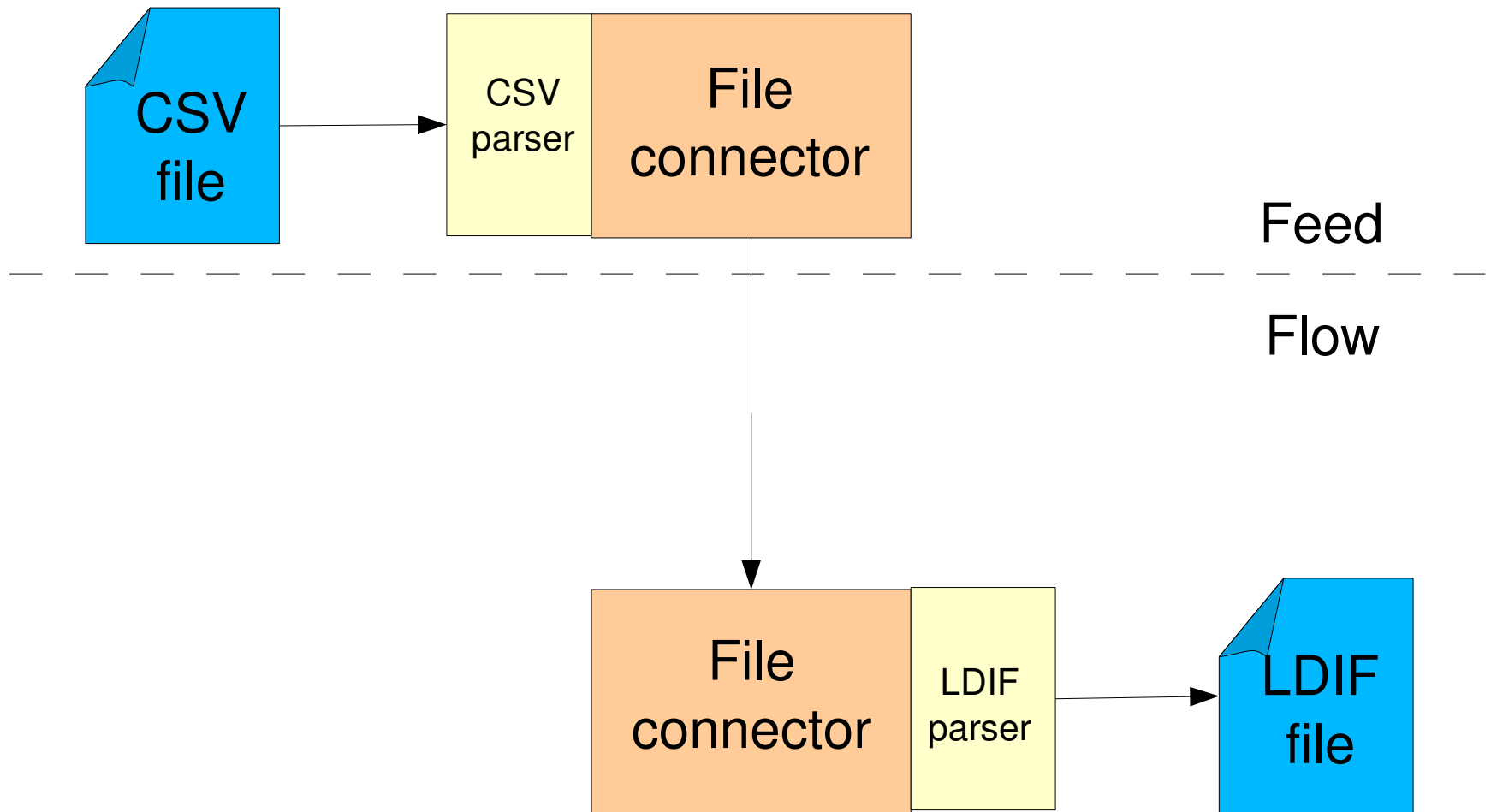


- Login and explore
- Install TDI
- Start the Config Editor

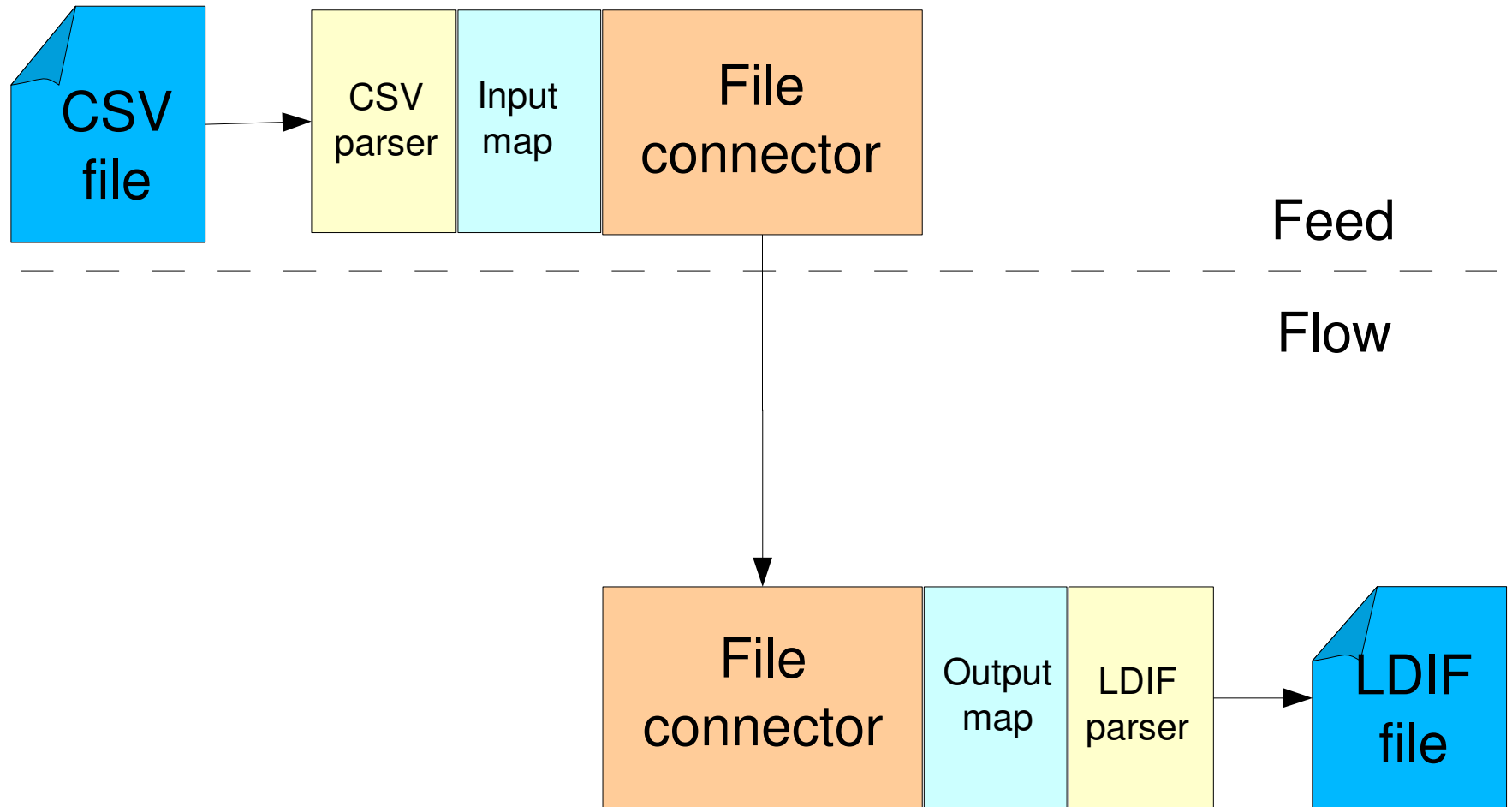
A Simple Assembly Line



Parsers



Input and Output Maps



Input Map and Schema Discovery

ReadCSV

Mode State Inherit From

Input Map

Map

Work Attribute	Assignment	Schema
firstname	conn.firstname	
surname	conn.surname	
username	conn.username	

Name	Sample Value	Reqd	Java Class	N
 firstname	Clive		java.lang.String	
 postcode			java.lang.String	
 surname	Rhodes		java.lang.String	
 username	u000002		java.lang.String	

Exercise 4



- Create an Assembly Line
- Read CSV file
- Write LDIF file

Data Stepper

AssemblyLine Outline AssemblyLine components, attributes and values

Next > Continue Stop Show/Hide Debugger

Feed
 ReadCSV
Data Flow
 IF: IF
 LDAPattrs
 DBLookup

AssemblyLine Work Bucket

Attribute	Value
\$dn	uid=u000005
cn	Nicholas Banks
firstname	Nicholas
givenname	Nicholas
objectClass	inetOrgPerson
postalcode	SW1A 1AA
postcode	SW1A 1AA
sn	Banks
surname	Banks

ReadCSV (Iterator)

Source Attribute	Source Value
firstname	Nicholas
postcode	SW1A 1AA
surname	Banks
username	u000005

DBLookup (Lookup)

Source Attribute	Source Value
firstname	Christopher
lastname	Bolouri
phone	+44 987 000005
pid	6
username	u000005

WriteLDIF (AddOnly)

Target Attribute	Target Value
\$dn	uid=u000005,dc=peop
cn	Nicholas Banks
objectClass	inetOrgPerson
postalcode	SW1A 1AA
sn	Banks
telephoneNumber	+44 987 000005
uid	u000005

// Enter javascript expression and hit return

```
13:29:31,750 INFO - CTGDIS295I AssemblyLine AssemblyLines/CSVtoLDIF is started.  
13:32:00,893 INFO - [ReadCSV] CTGDJW003I Parser will use first input line for column names.  
13:32:01,182 INFO - CTGDIS087I Iterating.
```

TDI Debugger

The screenshot displays the TDI Debugger interface for a process named CSVtoLDIF. The top toolbar includes icons for running, stepping, and other debugging actions, along with a text input field for JavaScript expressions. The main area is divided into two panes: AssemblyLine Components and Watch List.

AssemblyLine Components

- ☐ Hooks ☐ Attributes
- ☒ Feed
 - ☐ ReadCSV
- ☒ Data Flow
 - ☒ WriteLDIF

Watch List

Edit Watch List

Name/Expression	Value	Java Class
work	{ "surname": "Dobson", "firstname": "Caroline", "username": "u000001" }	com.ibm.di.entry.Entry
conn		Null
Global variables		
Watch List		

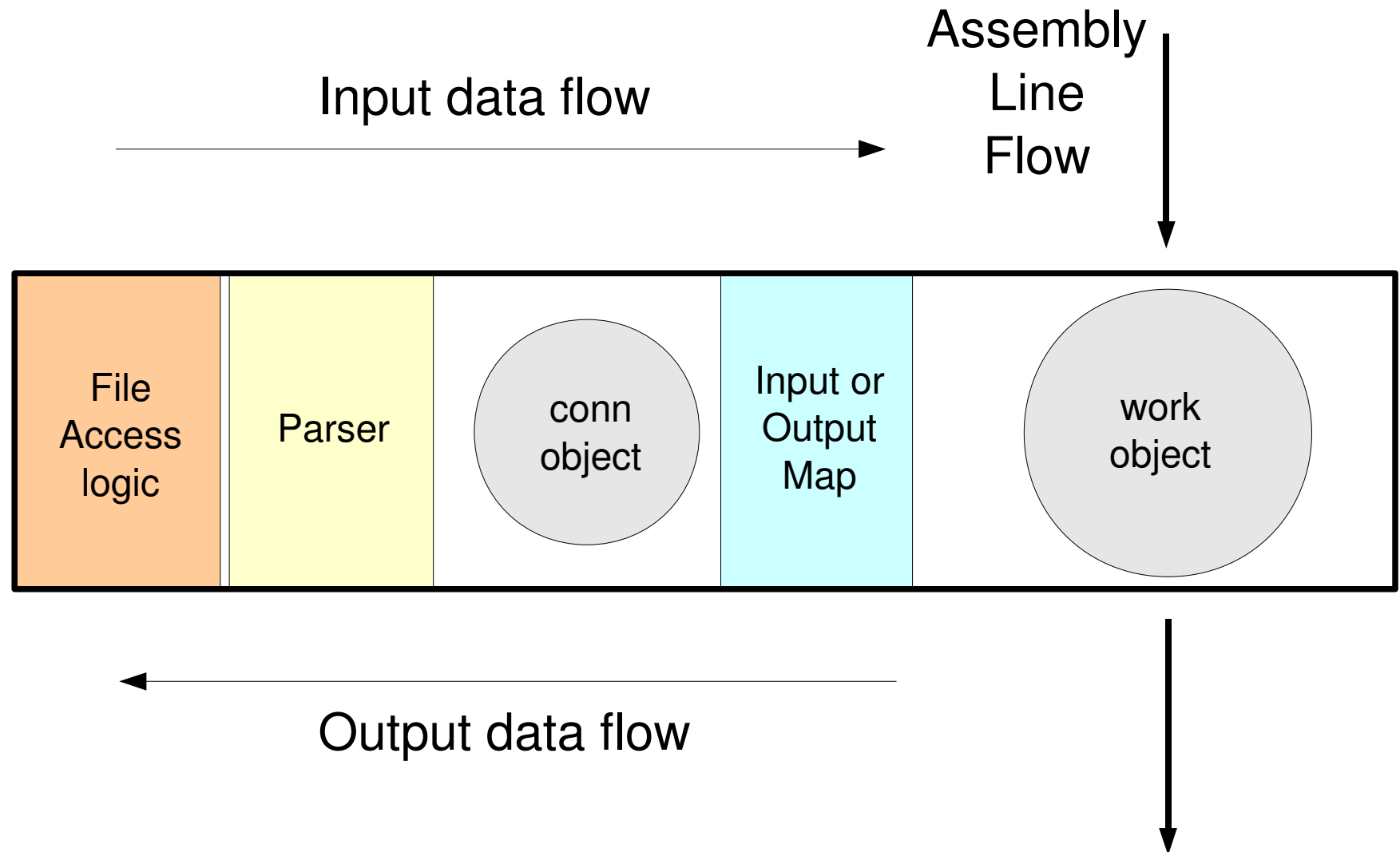
Log: 18:46:49,336 INFO - CTGDIS087I Iterating.

Exercise 5

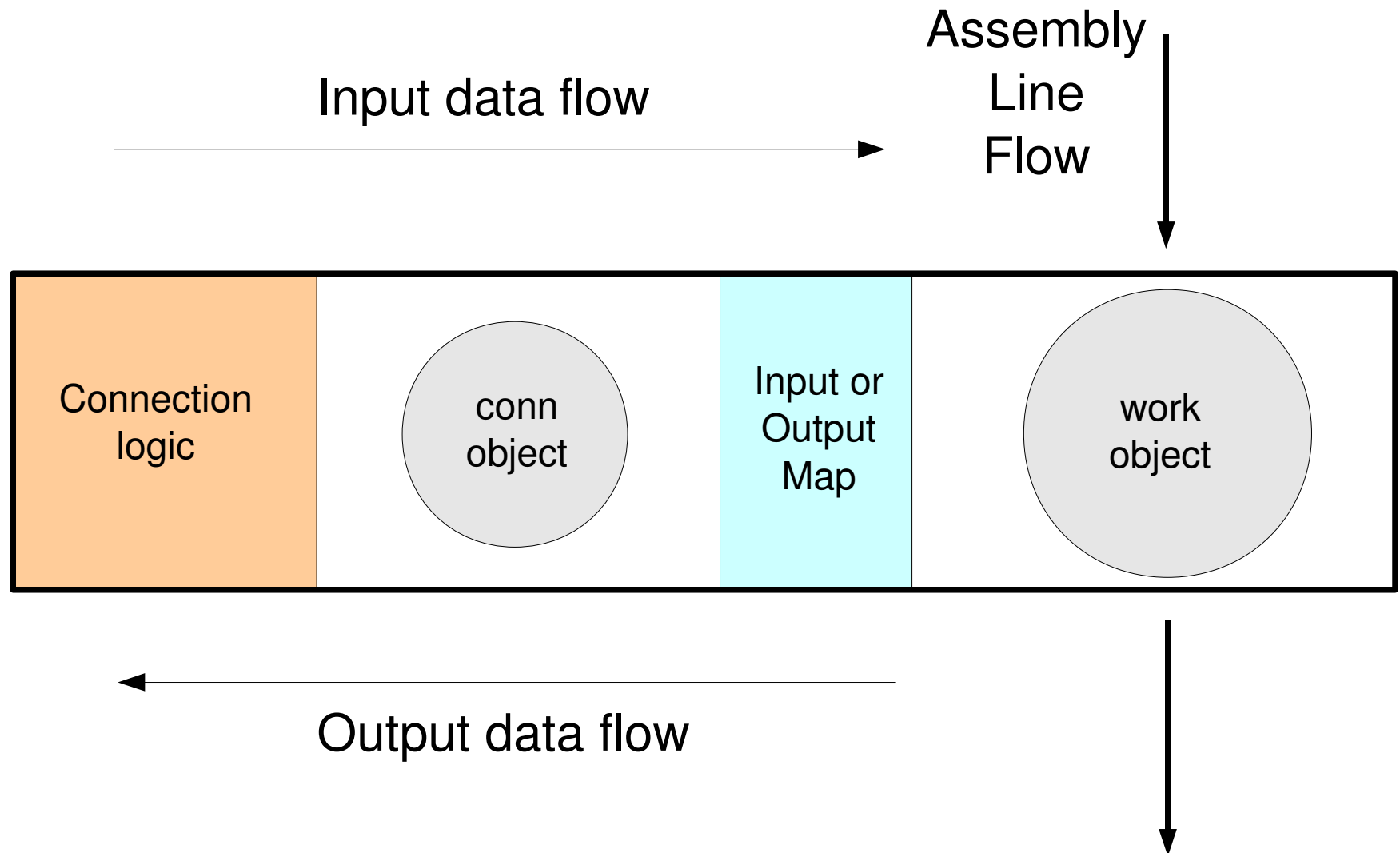


- TDI Debugger

Inside a Filesystem Connector



Inside a Database Connector



Scripting in Maps

- Scripts can build new values
`work.firstname + " " + work.surname`
- Scripts can be:
 - Expressions (as above)
 - JavaScript code
 - Text with substitution

Script Editor

- Ctrl-space gives list of valid objects
- 'objname.' and wait – list of methods
- Syntax sensitive

Java + Script != JavaScript

- Beware of object types
- Java objects behave differently
- Mapping can be ambiguous
- Returned objects not always strings
- See Users Guide *Scripting* section
- Keep data as all-Java or all-JavaScript if possible

Null Behaviour

- Data fields are not always filled in
- They may not exist at all in some records
- The Null Behaviour setting defines what happens
- Settable at each level
 - Assembly Line
 - Map
 - Attribute

Exercise 6



- Attribute Maps
- Improve your LDIF output

Flow Control: IF-THEN-ELSE

The screenshot shows the CSVtoLDIF application window. The left sidebar displays a tree view of components: Feed (ReadCSV), Data Flow (IF: IFHasNames, WriteLDIF, ELSE: ELSE, DumpWorkEnti). The main panel is titled 'IF branch' and contains a table for defining conditions.

Buttons at the top: Add component, Show mapping, Options..., Run, and a green play button.

Buttons in the IF branch panel: IF branch (dropdown), Close, Add, Remove, Move Up, Move Down, Script, Match all (checked).

	Attribute	NOT	Operator	Value	Case Sensitive
	firstname	☒	equals		☒
+	surname	☒	equals		☒

Buttons at the bottom: Collapse, Expand, Options...

Flow Control: Switch/Case

The screenshot displays the CSVtoLDIF application window. The top toolbar includes buttons for 'Add component', 'Show mapping', 'Options...', and a 'Run' button with a dropdown arrow. The left sidebar shows a tree view of the workflow: 'Feed' (containing 'ReadCSV') and 'Data Flow' (containing 'IF: IFHasNames', 'ELSE: ELSE', and 'SWITCH: {work.firstname}'. The 'SWITCH' component is selected and expanded, showing three cases: 'CASE: "Joe"' (containing 'DumpWorkEntry_1'), 'CASE: "Sally"', and 'CASE: "Eddie"'. The right pane is titled 'Switch' and contains buttons for 'Add Case components ...', 'Add Default Case', and 'Close'. It features a radio button selection for 'Work Attribute' (selected), 'AssemblyLine Operations', 'Work entry operations (delta entry)', and 'User Defined - Specify value/expression on each line'. A dropdown menu shows 'firstname'. Below this, a text area contains the expression '{work.firstname}'. At the bottom of the window are 'Collapse', 'Expand', and 'Options...' buttons.

Exercise 7



- Use IF and ELSE
- Report entries with missing names

Database Lookups

- Connect to many different 'databases'
- SQL
- LDAP
- SOAP
- REST
- IMAP
- SNMP
- FTP...

JDBC Connector

DBLookup

Mode State Inherit From

Input Map | Hooks | Link Criteria | **Connection** | Connection Errors

JDBC URL * ?

JDBC Driver * ?

Username ?

Password ?

Schema ?

Table Name * ?

Comment ?

Detailed Log ☐ ?

► **Advanced**

Link Criteria

DBLookup

Mode Lookup State Enabled Inherit From system:/Connectors/ibmdi.JDBC

Input Map Hooks **Link Criteria** Connection Connection Errors

☐ Build criteria with custom script

Add ☐ Match Any

username	▼	equa	▼	\$username	▼	ⓘ	Delete
-----------------------	----------------	-------------------	----------------	-------------------------	----------------	----------------	---------------------

Exercise 8

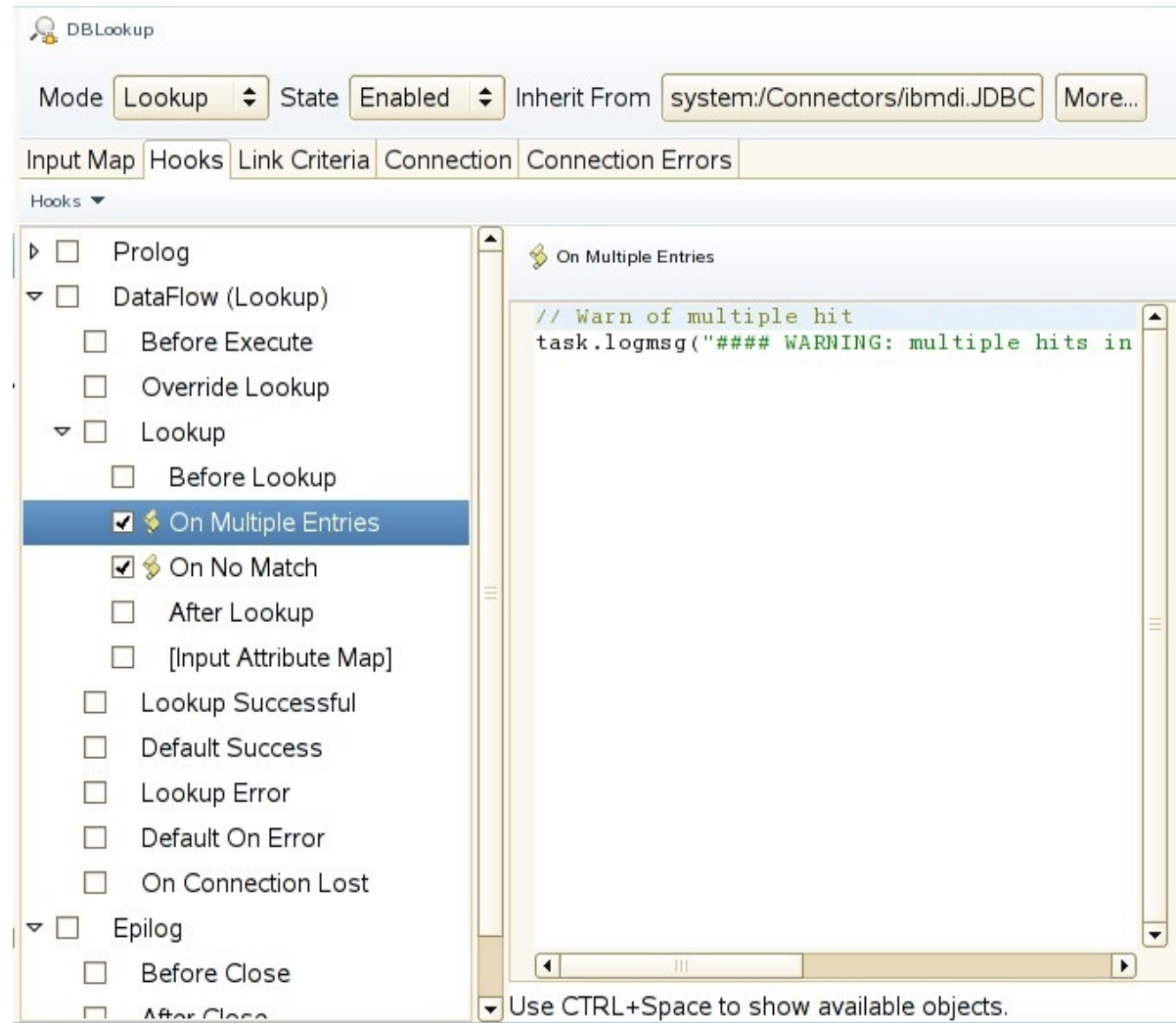


- Configure JDBC for MySQL
- Lookup phone numbers in SQL database

Hooks and Scripts

- Connector Hooks
- Assembly-Line hooks
- Using hooks for error handling
- Scripts to use in hooks
- Writing to the error log

Connector Hooks



Assembly Line Hooks

The screenshot displays the HRtoLDIF application interface. The 'Options...' button in the top toolbar is circled in red. A dropdown menu is open from this button, with 'AssemblyLine Hooks' selected and also circled in red. The 'DBLookup' component is selected in the left-hand component tree. The 'Hooks' tab is active, showing a list of hooks with checkboxes. The 'On Multiple Entries' hook is checked and highlighted. The 'Log Settings' tab is also visible, showing a log message: `// Warn of multiple task.logmsg("#### W`.

HRtoLDIF

Add component Show mapping Options... Run

Feed

- ReadCSV

Data Flow

- IF: IF
 - LDAPAttrs
 - DBLookup**
 - FOR-EACH: LookupCard
 - WriteLDIF
 - CorpLDAP
 - ELSE: ELSE

Mode Lookup

Input Map Hooks Li

Hooks

- ☐ Prolog
- ☐ DataFlow (Lookup)
 - ☐ Before Execute
 - ☐ Override Lookup
 - ☐ Lookup
 - ☐ Before Lookup
 - ☒ On Multiple Entries
 - ☒ On No Match
 - ☐ After Lookup
 - ☐ [Input Attribute Map]

AssemblyLine settings

Log Settings

AssemblyLine Hooks

Operations

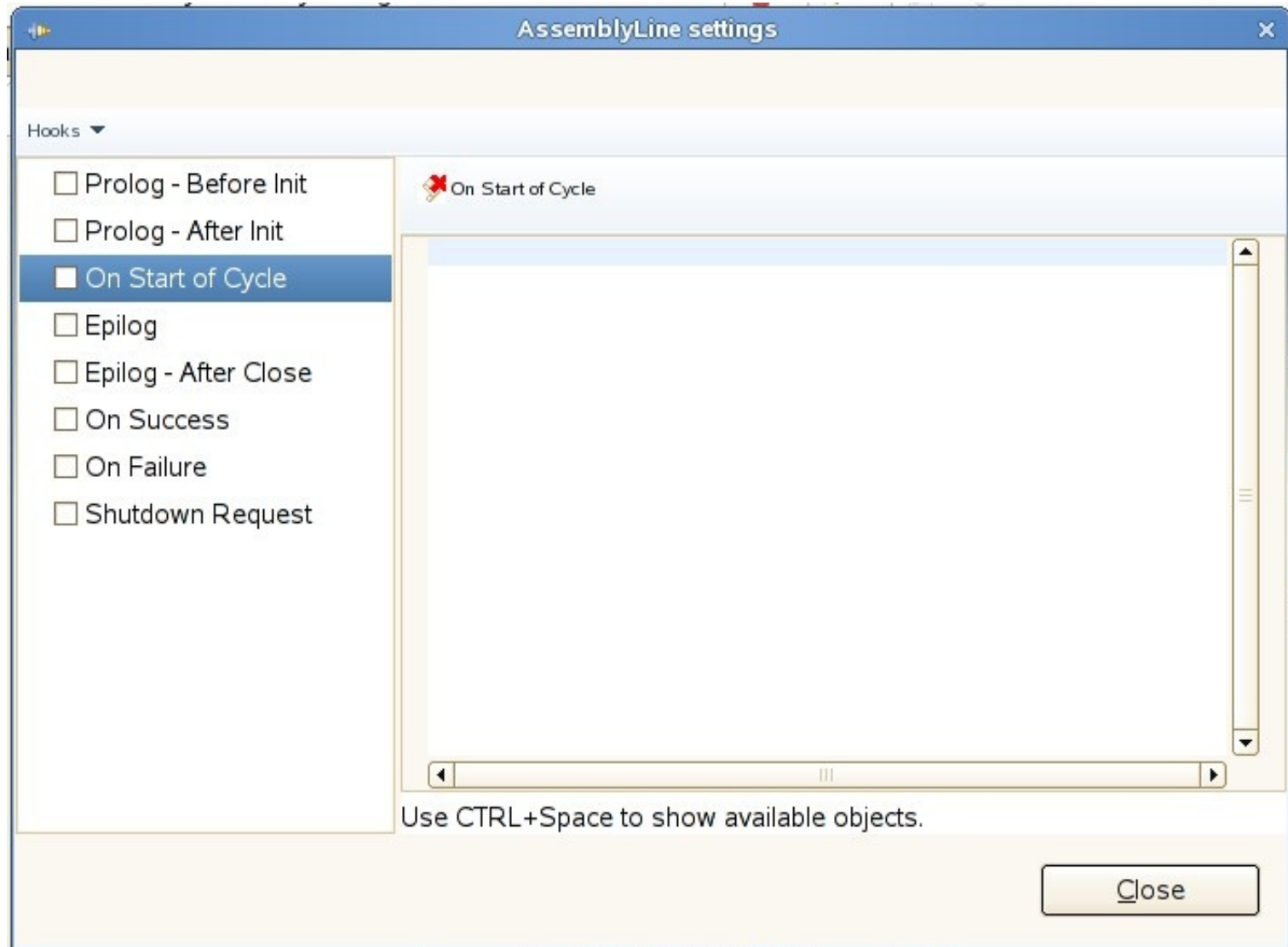
Simulation Settings

Sandbox Settings

On Multiple Entries

```
// Warn of multiple
task.logmsg("#### W
```

Assembly Line Hooks



Which hook?

- Flowcharts
- Debugger

Available objects

- Ctrl-space in script editor
- Typically:
 - work
 - conn
 - error
 - system
 - task
 - All named components in AL

Exercise 9



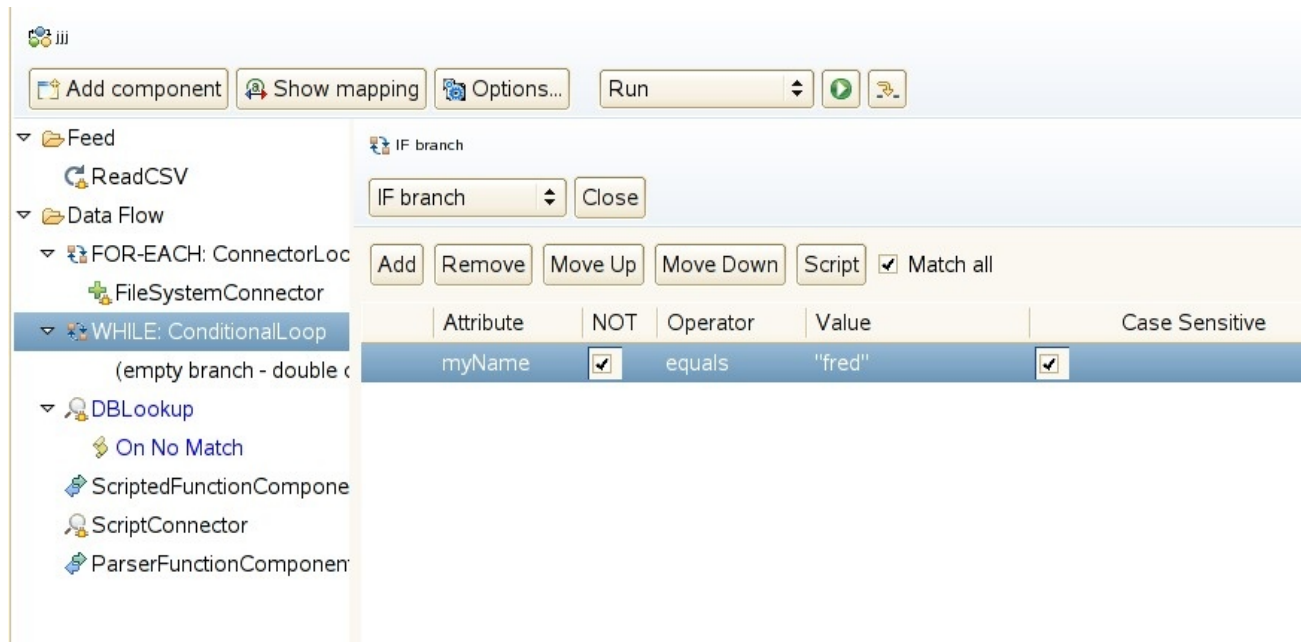
- Hooks and Scripts
- Basic error handling
- Missing Data
- Multiple hits

Loops

- Conditional loops
 - While <condition> do...
- Attribute-value loops
 - Handle each value separately
- Connector loops
 - Handle each entry of a multi-hit result

Conditional loops

- Loop continues while condition is true
- Condition set just like IF-THEN



Attribute-Value loops

- Iterate through values of one attribute
 - e.g. Members of an LDAP group

The screenshot shows the 'ReadGroups' tool interface. On the left, a tree view shows the workflow: 'Feed' (containing 'LDAPIterator') and 'Data Flow' (containing 'FOR-EACH Attribute' and 'ReportPhone'). The 'FOR-EACH Attribute' component is selected. On the right, the 'Attribute Value Loop' configuration panel is open. It contains a 'Close' button and a text instruction: 'Specify the name of the work attribute that contains the values to be looped over. For each value in this attribute a new attribute (the loop attribute) is set to the value for each iteration'. Below this, there are two input fields: 'Work Attribute Name:' with the value 'telephoneNumber' and a dropdown arrow, and 'Loop Attribute Name:' with the value 'thisPhone'.

ReadGroups

Add component Show mapping Options... Run

Feed

- LDAPIterator

Data Flow

- FOR-EACH Attribute
- ReportPhone

Attribute Value Loop

Close

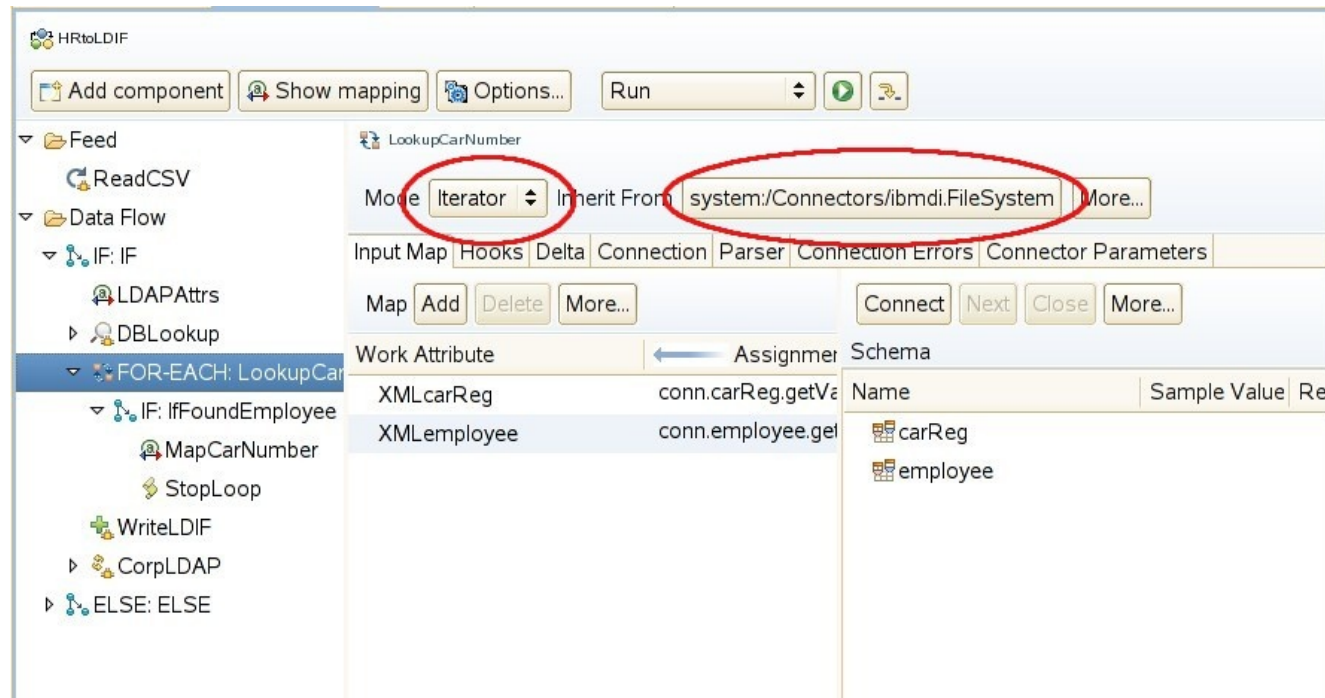
Specify the name of the work attribute that contains the values to be looped over. For each value in this attribute a new attribute (the loop attribute) is set to the value for each iteration

Work Attribute Name: telephoneNumber

Loop Attribute Name: thisPhone

Connector Loops

- Embedded connector in Lookup or Iterator mode
- Loops once for each entry returned



Exercise 10



- Connector loops
- Do lookup in XML file

Connector Modes

- Iterator
- Lookup
- AddOnly
- Delete
- Update
- Delta
- CallReply
- Server

Iterator Mode

- Read every entry in a data source
- Commonly used to process files
- Some (e.g. LDAP, JDBC) can select a subset of entries by configuration
- Normally used in the Feed section

Iterators with Delta Engine

- Detect changes in data sources
- Stores copy in TDI system store
- Issues tagged attributes in work object
 - Add
 - Delete
 - Modify
- Some connectors can use these tags (Delta Mode)
 - Efficient update with no pre-read

Lookup Mode

- Use attributes from the *work* object to search a data source: Link Criteria
- Import values from the data source into *work* attributes: Input Map
- Source is usually a database but can be almost anything

Add-Only Mode

- Adds one entry to a data source
- Often used to write files
- Output map only

Delete Mode

- Delete a single entry from a database
- Use Link Criteria to choose the entry
- Has an input map
 - Used to return the contents of the deleted entry

Update Mode

- Link criteria define entry to update
- If no entry found, adds one
- Separate control over attributes:
 - Updating
 - Adding
- Can override add or modify in hooks
- Compute changes option

Update Mode

CorpLDAP

Mode **Update** State **Enabled** Inherit From **system:/Connectors/ibmdi.LDAP** **Less...**

Lookup Limit **10** ☒ Compute Changes ☐ Skip Lookup Initialize at startup

Output Map Hooks Link Criteria Connection Connection Errors

Map **Add** **Delete** **More...** **Connect** **Next** **Close**

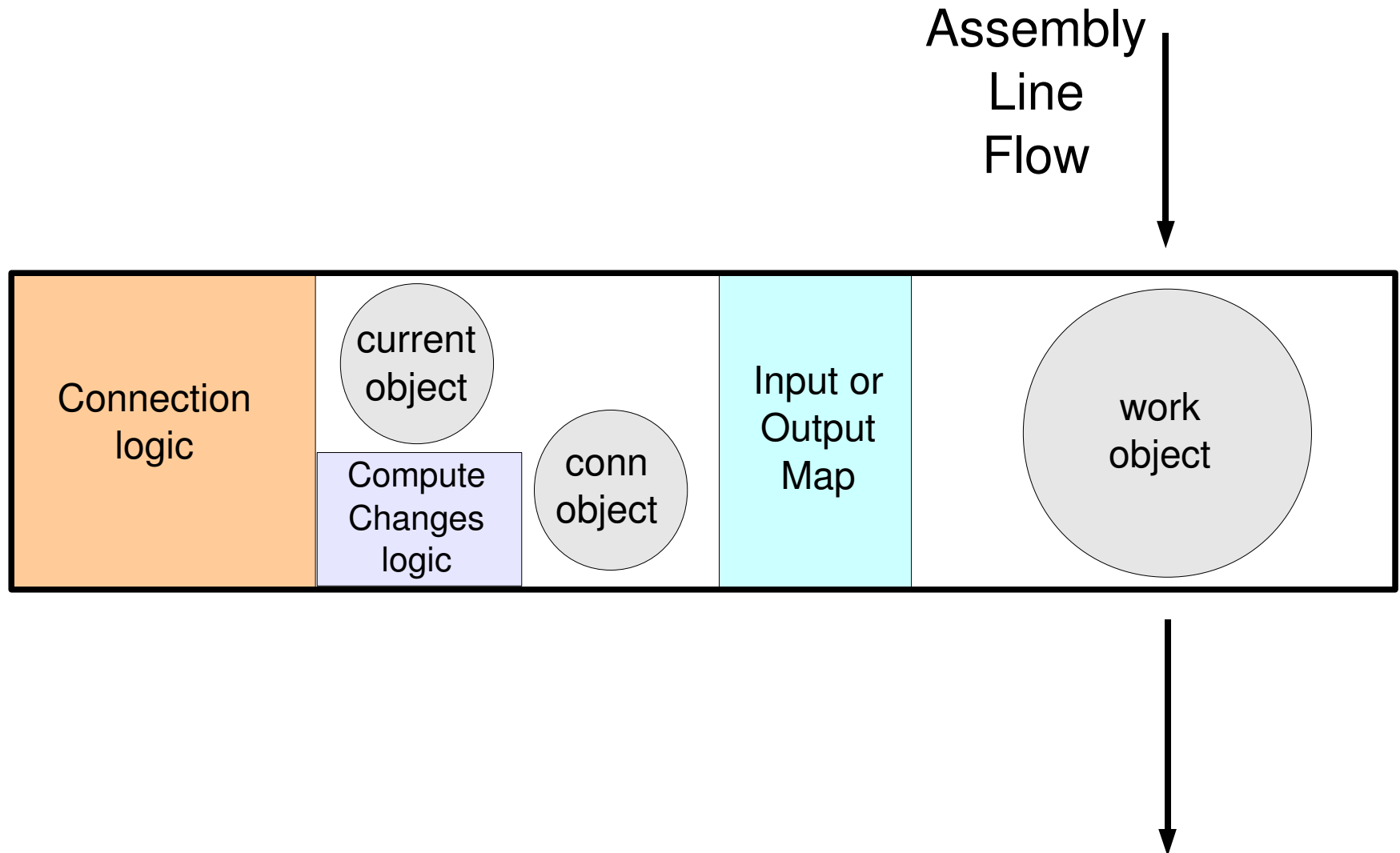
Assignment	Add	Mod	Component Attrib
work["\$dn"]	true	false	\$dn
work.carLicense	true	true	carLicense
work.cn	true	true	cn
work.givenName	true	true	givenName
work.objectClass	true	false	objectClass
work.postalCode	false	true	postalCode
work.sn	true	true	sn
work.telephoneNumber	true	true	telephoneNumber
work.uid	true	false	uid

More...

Name

- \$dn
- audio
- businessCategory
- carLicense
- cn
- departmentNumber
- description
- displayName
- employeeNumber
- employeeType

Database Connector in Update



Delta Mode

- Like Update mode but uses Delta tags
- Can also process deletes
- Very efficient synchronisation when used with the Delta Engine
 - Take care! This only works if it is the *only* process updating the target data.

Call-Reply Mode

- Send one or more attributes to a remote service
- Receive one or more attributes back
- Similar to Lookup mode
- Has both input and output maps

Server Mode

- Server connectors listen for events
 - Incoming message
 - TCP connection
 - Clock tick
- Parse incoming data
- Build work object
- Create network response at end of AL

Server connectors available

- Axis2 web service server
- DSMLv2 SOAP server
- HTTP server
- LDAP server
- SNMP server
- TCP server
- Web service receiver server
- ... other server-like things

Exercise 11



- LDAP server
- Apache Directory Studio
- LDAP connector

TDI system store

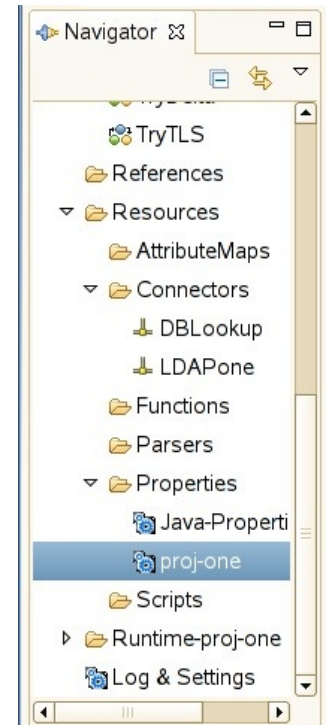
- Derby / Cloudscape database
- Embedded or Server mode
 - Embedded only usable by single process
 - Server mode now preferred
- Supports Delta Engine and other features
- Can be used from scripts

Queues

- General-purpose mechanism
- Memory queue
- System queue
- JMS
 - MQe etc

Property Stores

- Remove connection details from config
- Name-Value stores
- Multiple stores possible
- CE can upload to remote stores
- Properties can be protected (encrypted in file)



Edit Properties

*LDAPserver *HRtoLDIF *jji ReadGroups *proj-one

CTGDIC037I Update completed succesfully.

Properties

Add property Download Upload

Editor Connector

Search: ☐ Hide non-local properties

Name	Protected	Local Value	Server Value
LDAP_MGR_DN	false	cn=root,dc=example,dc=org	cn=root,dc=example,dc=org
LDAP_MGR_PW	true	*****	*****
LDAP_URL	false	ldap://:1389/	ldap://:1389/

Exercise 12



- Build a property store
- Use properties in LDAP configuration

Handling variable-size results

- Zero results may not be an error
- Multiple results may not be an error
 - OK to have multiple phone numbers
- Handle these cases with loops:
 - Connector loops when searching
 - Attribute-value loops when updating
- Scripts can add new values to attributes

Building Attributes

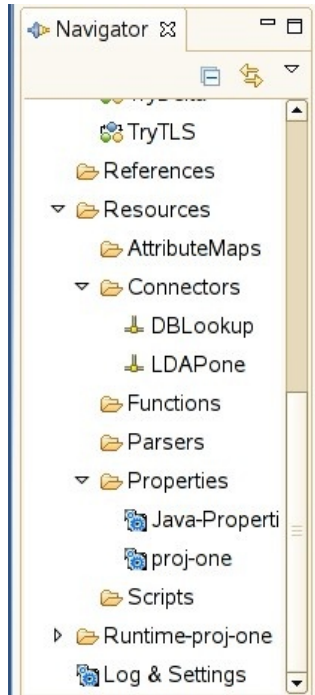
- `work.newAttribute("telephoneNumber")`
- `work.telephoneNumber.addValue(...)`
- `work.getString("phone")`

Exercises 13 and 14



- Build multi-value attributes
- Build a web-server that answers white-pages queries

Building a Library



- Resources
 - Attribute Maps
 - Connectors
 - Functions
 - Parsers
 - Properties
 - Scripts
- Inheritance

Connector Re-Use

- Re-use within an AL
- Single connection to data source
- Lower resource usage
- May not mix operation types
- Inherits config
 - Items marked in blue

Connector Pooling

- Start with a connector in the library
- Enable Pooling (*Pool* tab)
- Set pool size
- TDI runtime maintains a pool of initialised connectors

Function Components

- Attribute Map
- Parser
- Assembly Line
- SOAP support
- Castor XML support
- Memory queue
- Send e-mail
- etc...

Scripted Components

- Write your own
 - Connector
 - Function
 - parser
- *Connection* tab has *Edit Script* button
 - First use provides template script

Exercise 15



- Delta mode

SSL and TLS

- Encryption
- Public-Key system
- Authentication

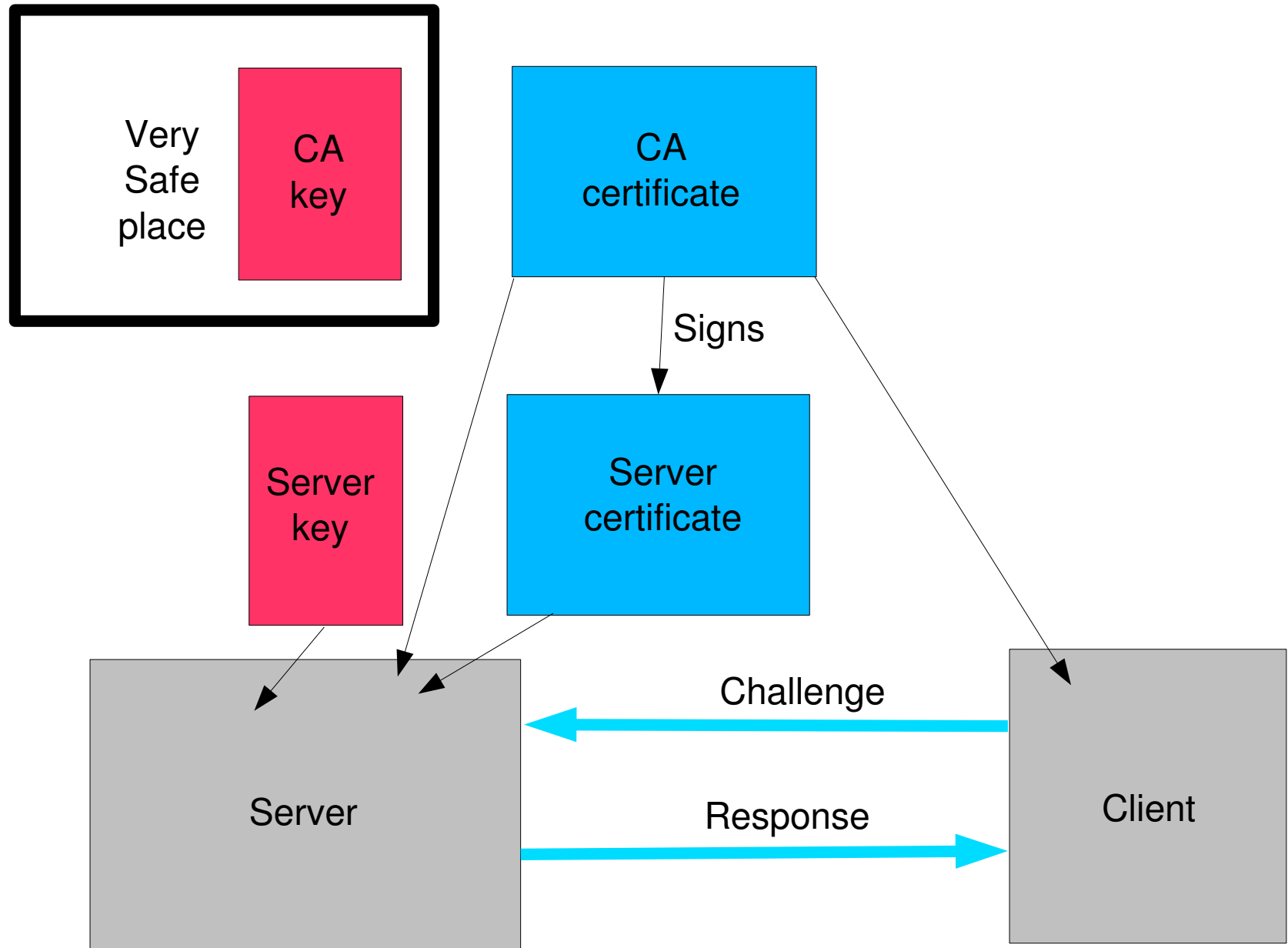
Public-key cryptosystems

- RSA, Diffie-Hellman etc
- One key to encrypt
- One key to decrypt
- Cannot derive one from the other
- Uses:
 - Key distribution
 - Zero-knowledge proof => authentication
 - Session-key generation

Certificates

- Use public-key system to sign data
 - Make a tamper-evident container
 - Bind a name to a key-pair => authentication
 - Use one certificate to sign others => Public Key Infrastructure
 - Certificate contains the public key => key distribution

Certificates for SSL/TLS



SSL or TLS

- SSL sets up encryption first
- TLS layers encryption into running protocol
- SSL needs new port number and also new IP address for each service
 - Deprecated
- Use TLS if possible
 - More hardened protocol

Exercise 16



- Use TLS with LDAP

Password Synchronisation

- Passwords not usually stored in clear
- Hash forms not usually cross-platform
- Password hashes often unobtainable
- Must *capture* plain-text passwords
 - At the point of use
 - When password is changed

Password Capture Plugins

- TDI supplies plugins for
 - Standalone Windows (XP and later)
 - Active Directory domains
 - Sun Directory Server
 - IBM Tivoli Directory Server
 - Lotus Domino
 - Unix/Linux PAM systems

Example Plugins

- Active Directory
 - DLL
 - Acts as a password-quality checker
 - Must be installed on all domain controllers
- Unix/Linux PAM
 - Shared Library
 - Configure into the normal PAM stack

Alternatives to the plugins

- Ordinary LDAP searches
 - If passwords stored in clear
 - If password hashes are usable
- Other capture mechanisms
 - Microsoft Services For Unix (SFU)
 - Password Hook:
passwdhk.sourceforge.net

Password Storage

- Hold captured passwords securely
- LDAP
- JMS (MQe)
- Both can use SSL
- Passwords can be separately encrypted for transport (reversible)
 - AL must have access to the keys
 - Script needed for decryption

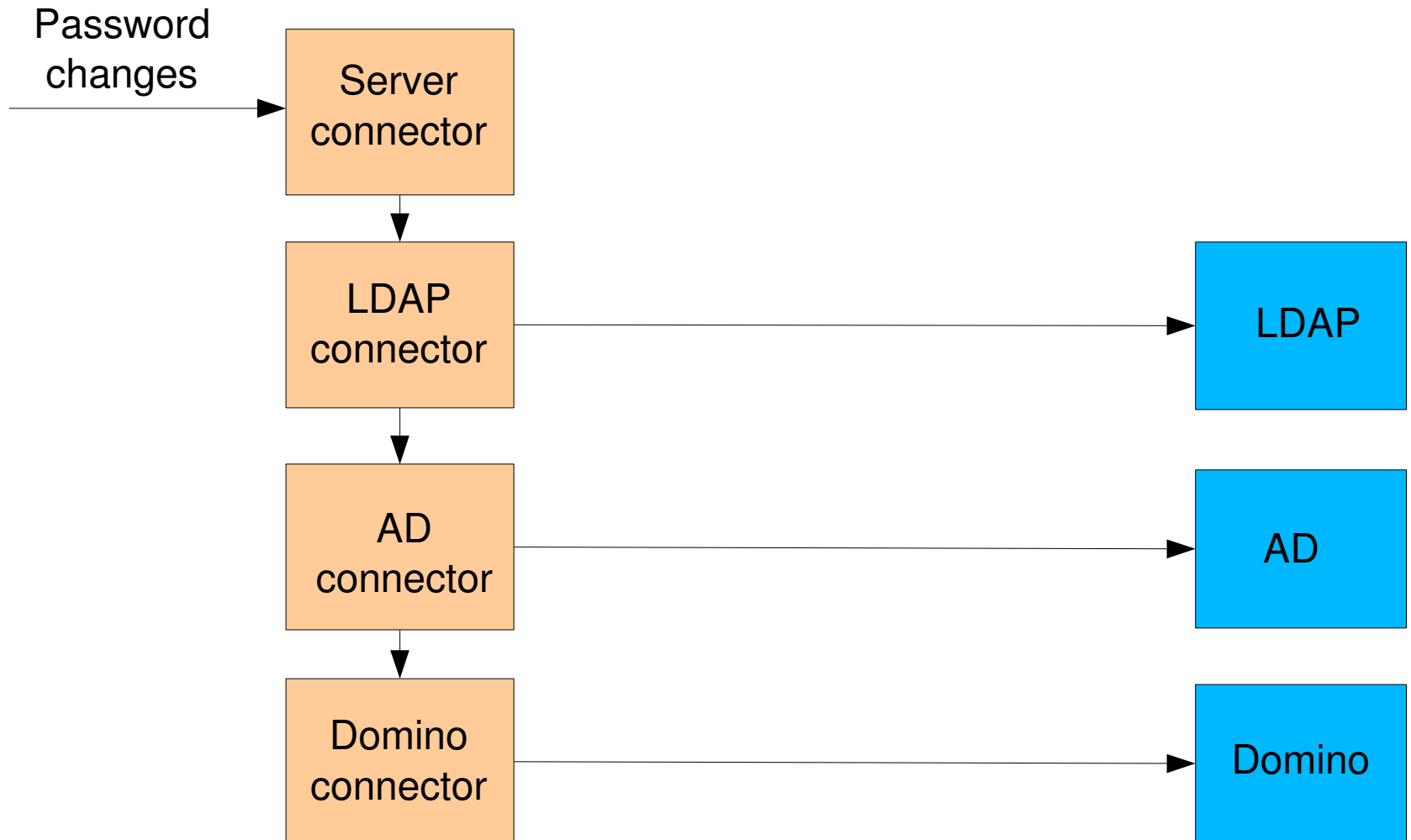
Robust Systems

- Good error-handling
- Not losing updates on error or crash
- Recover from connectivity problems
- Testable components
- Good architecture
 - Simple single-task ALs
 - Each data flow independent of others
 - Well-defined, replaceable ALs

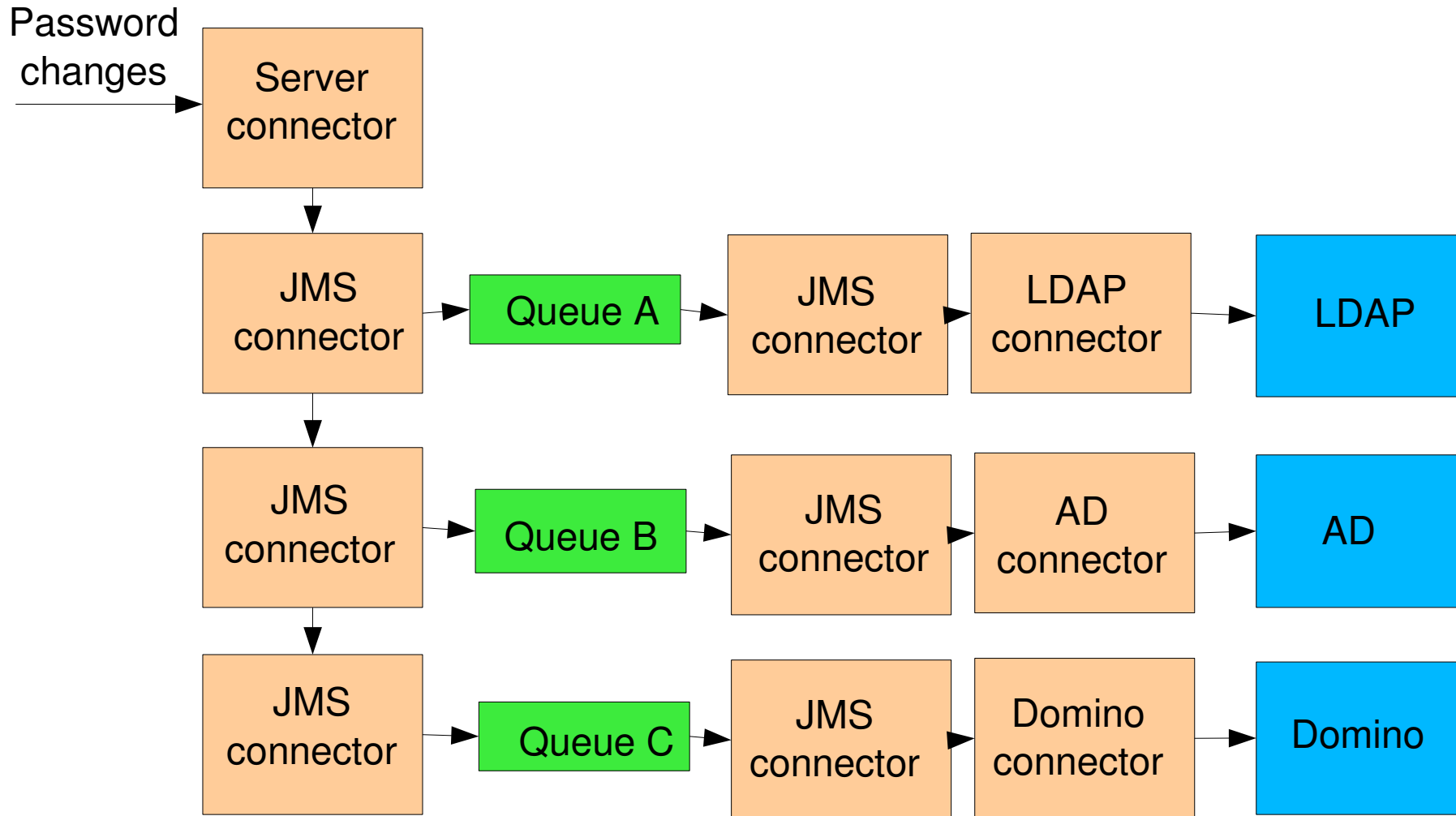
Error Handling

- Don't just crash
 - At least log a useful message
 - Try to recover
- Don't lose updates
 - Hold work-in-progress in persistent store
 - Write Idempotent assembly lines
 - Failure of one target must not prevent update of others
- Avoid massive damage
 - Set limits on critical changes

A risky assembly line



A safer approach



Reconnecting

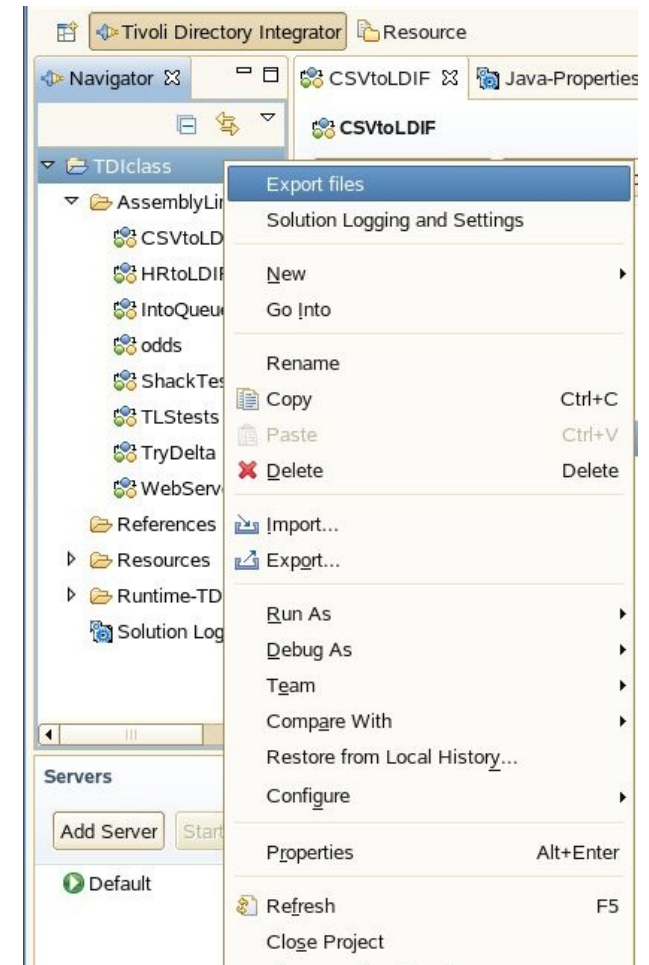
- *Connection errors* tab in connector
- Retry on:
 - Initial connection failure
 - Connection loss in operation
- Configure timeout and retry limit
- Beware: takes time, blocks error hooks

Going Into Production

- Export config files for runtime
- Property files
- Boot-time startup
- Scheduled tasks
- Logging
- Monitoring
 - AMC
 - Action Manager

Exporting the runtime config

- Right-click on the project
- Select *Export Files*
- Choose a directory
- Choose ALs to run



Starting the runtime server

- Installer adds lines to /etc/inittab on Unix-like systems: server runs as *root*
- Installer creates a system service on Windows: server runs as *System*
- Consider security!
 - Does the task *need* that much power?
- Consider command-line startup
 - ibmdisrv
 - tdisrvctl

Scheduling ALs

- Timer Connector
 - Feed a work object into an AL at certain times of day
 - Scheduling AL could call others
- Crontab
 - Invoke ALs from command-line

Monitoring TDI

- Build an AL to monitor the others
 - Report to enterprise console etc.
- AMC: Admin and Monitoring Console
 - Bundled with TDI
 - Can invoke “Health AL” and display result
- Action Manager
 - Automatic actions based on rules
- SNMP

Exercises 17 - 23



- Prepare for production
- AMC
- Report differences between data sources
- Account provisioning

Summary

- TDI structure
 - CE and Runtime
 - Assembly Lines, Connectors, etc
 - Scripts and hooks
 - Debugger
- Solution design

What have we missed?

- Many advanced topics
 - Creating re-usable TDI components
 - Access-control for runtime API
- Many types of connector
- Many types of parser
- Action Manager

Learning more

- TDI Users website:
www.tdi-users.org
- Google Group:
ibm.software.network.directory-integrator
- PDF manuals from IBM website