~~Some~~ A couple of new technologies

Worth getting to know

Topics

- ØMQ – Lightweight messaging
- CMIS – A Protocol for interacting with ECM systems

ØMQ

- “No man is an island”
- Many options
  - MQSeries
  - RabbitMQ
  - ApacheMQ
  - OpenMQ
- Many attributes
  - %CPU
  - Footprint
  - Latency
  - Configuration (brokers etc.)

ØMQ

- ZeroMQ
- Library of APIs
- Modeled on standard TCP/IP semantics
- Not a message broker
  - But can be used to create one

ØMQ

```
static size_t __inline__getData(int sd, char *buffer, size_t size)
{
        size_t lMsg = 0;
        while (lMsg < size) {
          lMsg = recv(sd, (buffer+lMsg), (size – lMsg));
        }
        return(lMsg);
}
:
size_t msgLen;
char *msgData;

getData(sd, (char *) &msgLen, sizeof(msgLen));
msgData = malloc(msgLen);
getData(sd, msgData, msgLen);
zmq_msg_t request;
char *msgData;
int msgLen;

zmq_msg_init(&request);
zmq_recv(sd, &request, 0);
msgLen = zmq_msg_size(&request);
msgData = zmq_msg_data(&request);
```

ØMQ

```
zmq_msg_t request;
char *msgData;
int msgLen;

zmq_msg_init(&request);
zmq_recv(sd, &request, 0);
msgLen = zmq_msg_size(&request);
msgData = zmq_msg_data(&request);
```

ØMQ

- Multiple Carriers
  - **tcp://** is a plain old TCP socket with a host and port number.
  - **ipc://** uses UNIX inter-process communication such as domain sockets, MQ, or whatever is available.
  - **inproc://** is an in-process transport that passes messages via memory directly between threads sharing a single ØMQ context.
  - **pgm://** is reliable multicast messaging that uses raw IP layering and requires special privileges.
  - **epgm://** is an encapsulated version that uses regular User Datagram Protocol (UDP) to do reliable multicast messaging.

ØMQ

- N-to-N Dissemination
  - ØMQ sockets may be connected to multiple endpoints using zmq_connect(), while simultaneously accepting incoming connections from multiple endpoints bound to the socket using zmq_bind(). This allows many-to-many relationships
- Low Overhead and Fast Messaging
- Asynchronous I/O
- No need for mutexes, locks, or any other form of inter-thread communication

ØMQ

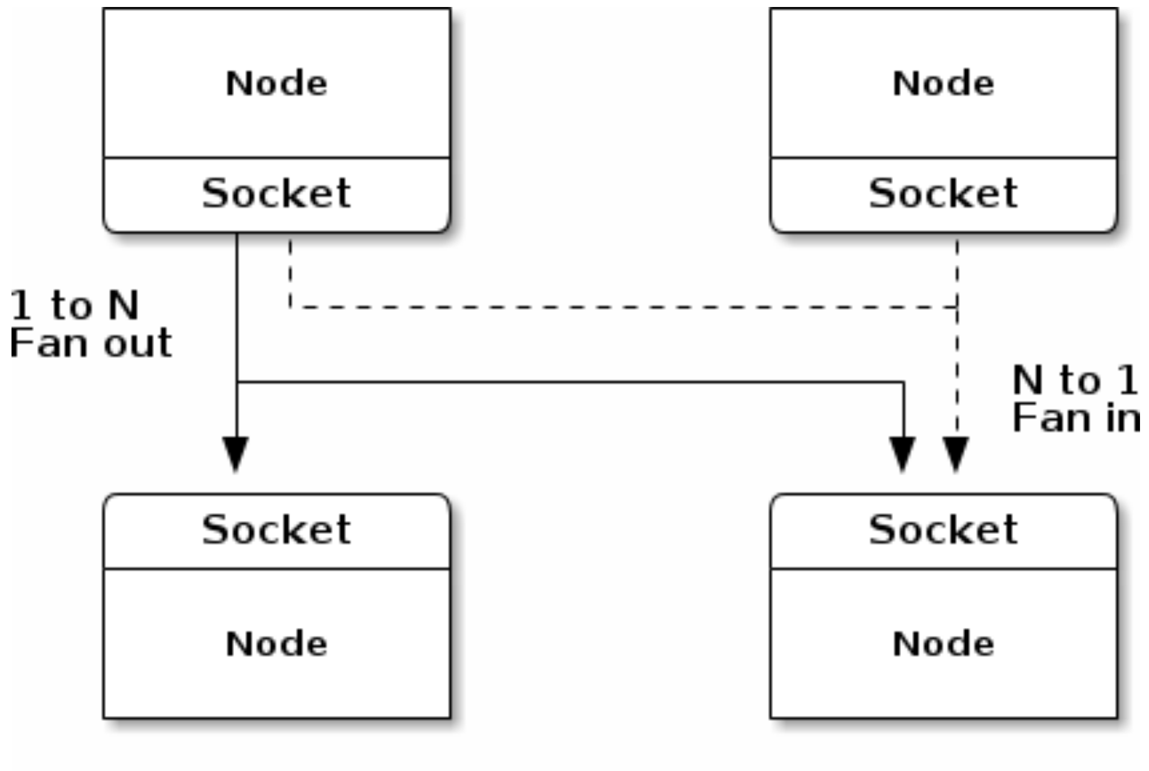


Figure 11 – ØMQ sockets are N to N

ØMQ

- Language bindings exist for:
  - Ada
  - Basic
  - C
  - Chicken Scheme
  - Common Lisp
  - C# (.NET & Mono)
  - C++
  - D
  - Erlang
  - Go
  - Haskell
  - Java
  - Lua
  - node.js
  - Objective-C
  - ooc
  - Perl
  - PHP
  - Python
  - Racket
  - Ruby
  - Tcl
- ØMQ is available on multiple platforms, including Linux, Windows, Solaris, and OpenVMS.

ØMQ

Request-Reply	Used for sending requests from a *client* to one or more instances of a *service*, and receiving subsequent replies to each request sent.
ZMQ::REQ	Used by a client to send requests to and receive replies from a service. Each request sent is load-balanced among all services, and each reply received is matched with the last issued request
Compatible peer sockets	ZMQ::REP, ZMQ::XREP
Direction	Bidirectional
Send/receive pattern	Send, Receive, Send, Receive, ...
Outgoing routing strategy	Load-balanced
Incoming routing strategy	Last peer
HWM[1] action	Block
ZMQ::REP	Used by a service to receive requests from and send replies to a client
Compatible peer sockets	ZMQ::REQ, ZMQ::XREQ
Direction	Bidirectional
Send/receive pattern	Receive, Send, Receive, Send, ...
Incoming routing strategy	Fair-queued
Outgoing routing strategy	Last peer
HWM action	Drop

ØMQ

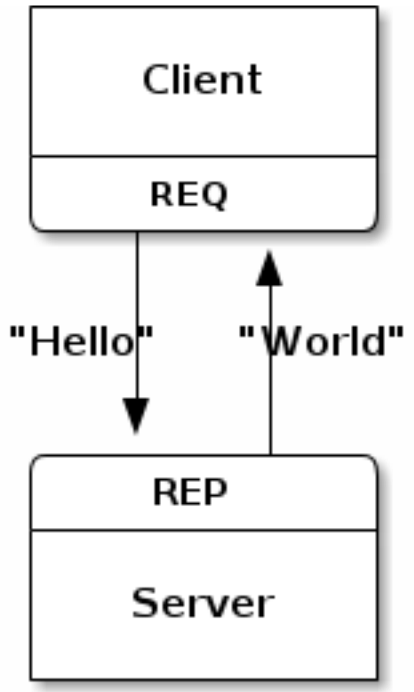


Figure 1 — Request-Reply

ØMQ

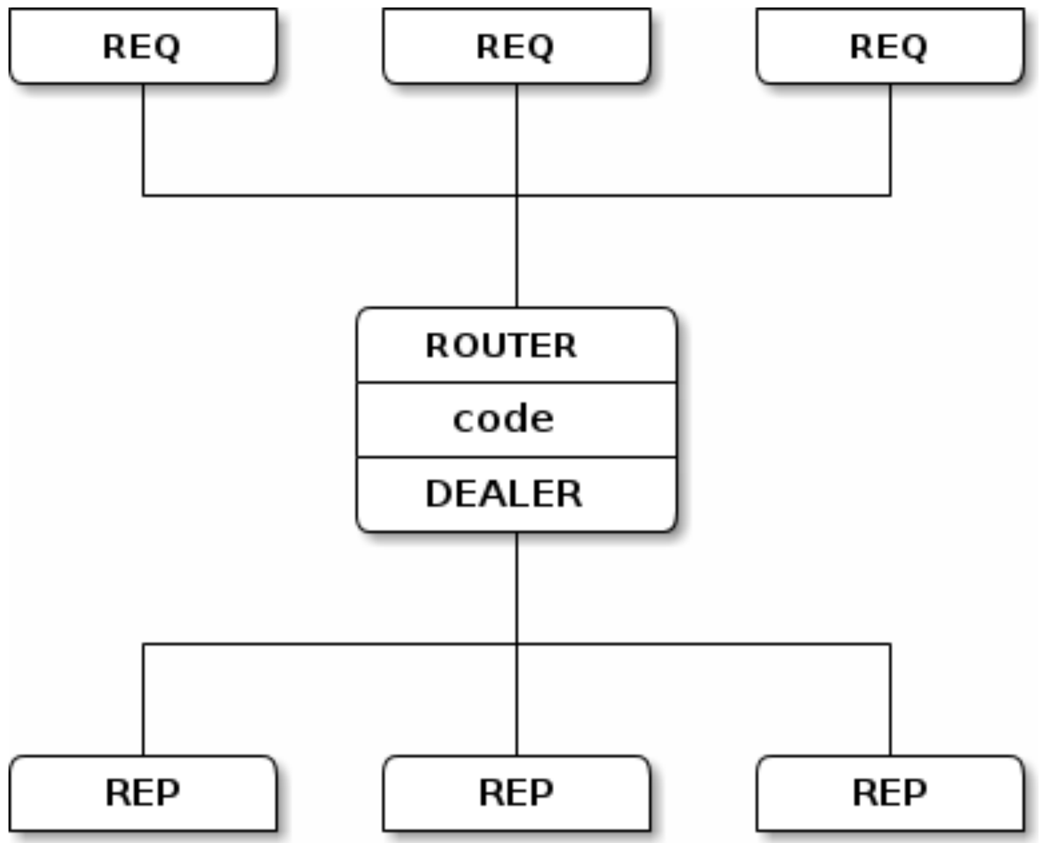


Figure 19 – Extended request-reply

ØMQ

Publish-Subscribe	Used for one-to-many distribution of data from a single publisher to multiple subscribers in a fan out fashion
ZMQ::PUB	Used by a publisher to distribute data. Messages sent are distributed in a fan-out fashion to all connected peers
Compatible peer sockets	ZMQ::SUB
Direction	Unidirectional
Send/receive pattern	Send only
Incoming routing strategy	N/A
Outgoing routing strategy	Fan-out
HWM action	Drop
ZMQ::SUB	Used by a *subscriber* to subscribe to data distributed by a *publisher*.
Compatible peer sockets	ZMQ::PUB
Direction	Unidirectional
Send/receive pattern	Receive only
Incoming routing strategy	Fair-queued
Outgoing routing strategy	N/A
ZMQ::HWM option action	N/A

ØMQ

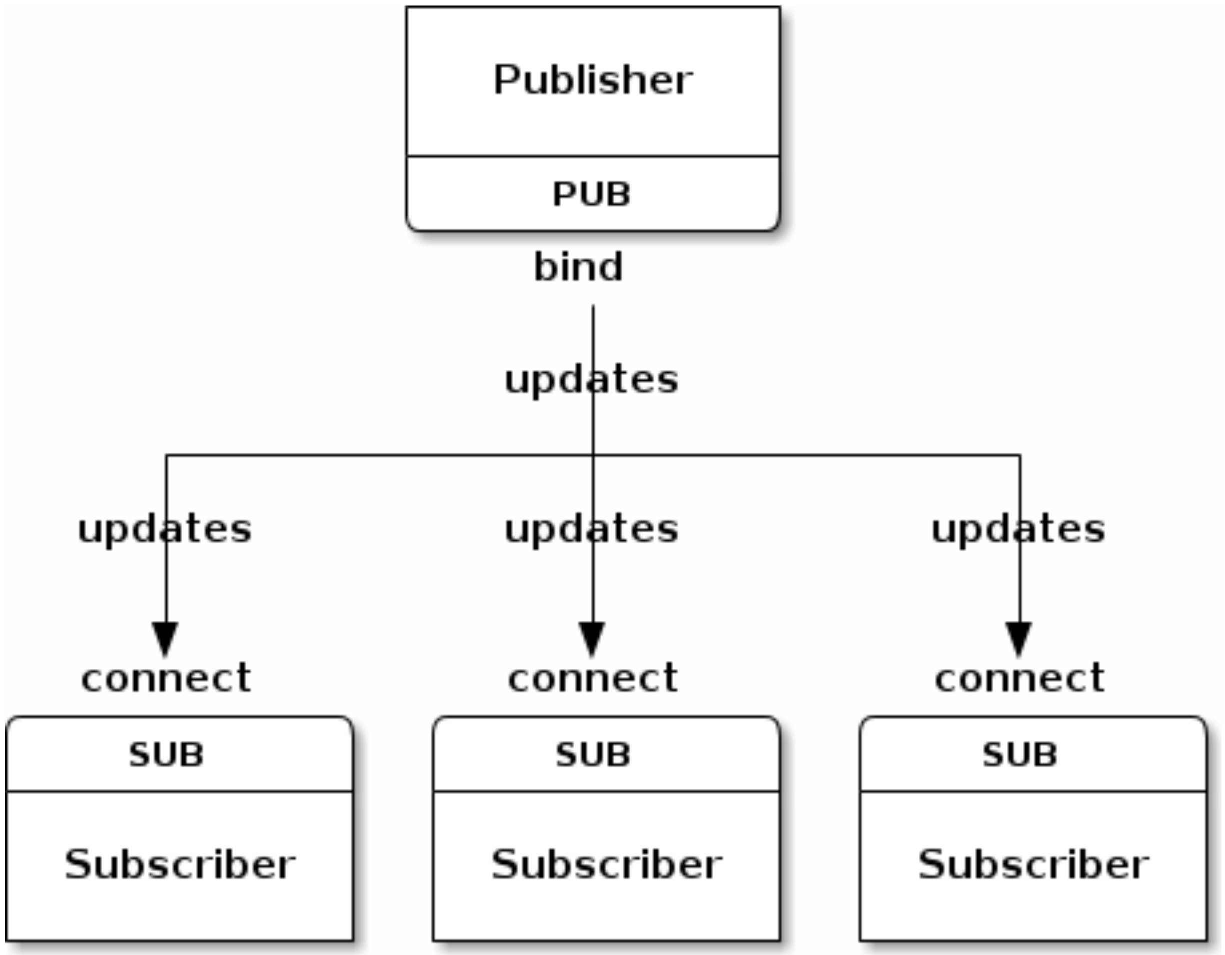


Figure 4 – Publish-Subscribe

ØMQ

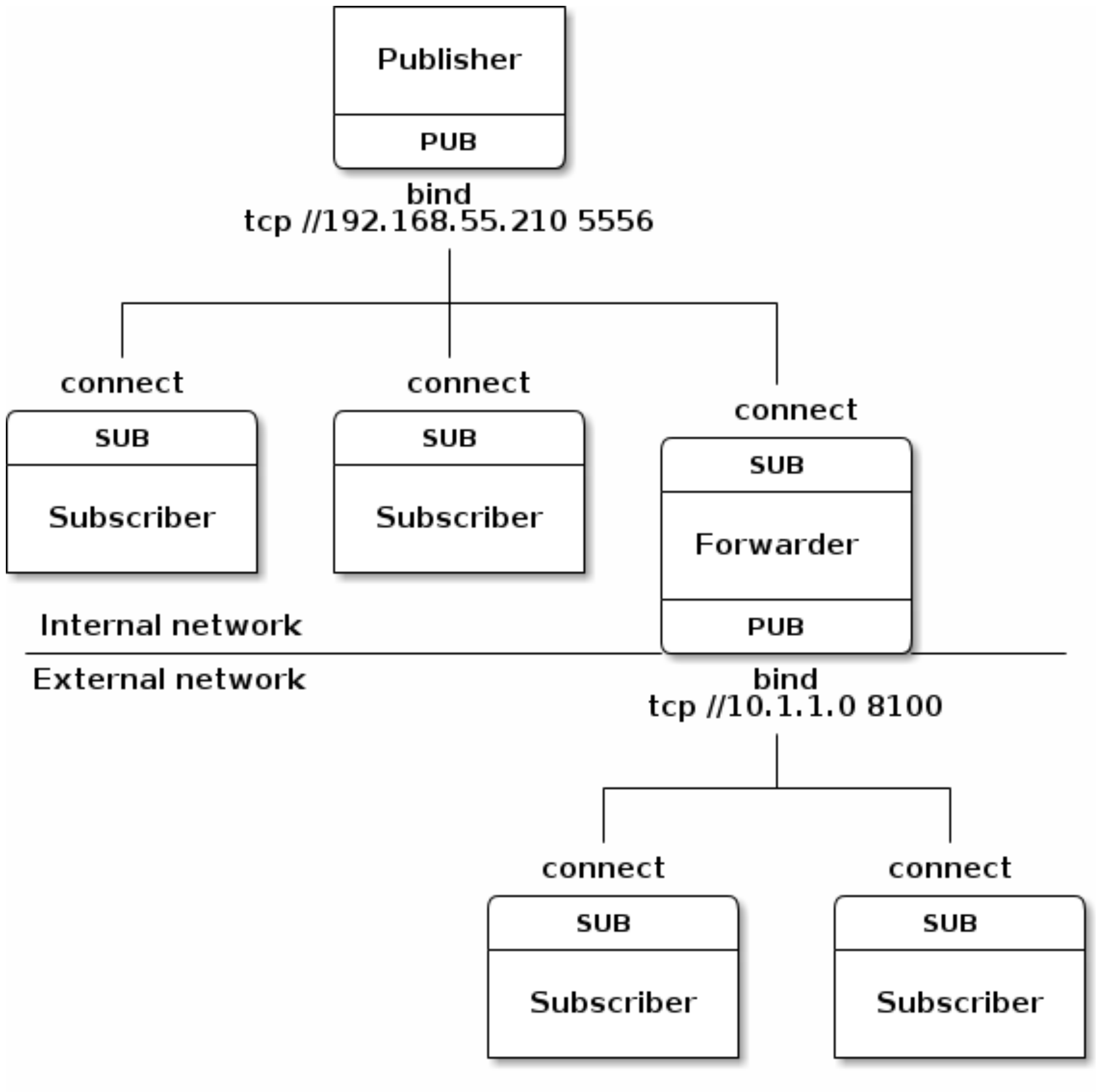


Figure 17 – Forwarder proxy device

ØMQ

Pipeline	Used for distributing data to *nodes* arranged in a pipeline. Data always flows down the pipeline, and each stage of the pipeline is connected to at least one *node*. When a pipeline stage is connected to multiple *nodes* data is load-balanced among all connected *nodes*.
ZMQ::PUSH	Used by a pipeline node to send messages to downstream pipeline nodes. Messages are load-balanced to all connected downstream nodes.
Compatible peer sockets	ZMQ::PULL
Direction	Unidirectional
Send/receive pattern	Send only
Incoming routing strategy	N/A
Outgoing routing strategy	Load-balanced
HWM action	Block
ZMQ::PULL	Used by a pipeline *node* to receive messages from upstream pipeline *nodes*. Messages are fair-queued from among all connected upstream nodes.
Compatible peer sockets	ZMQ::PUSH
Direction	Unidirectional
Send/receive pattern	Receive only
Incoming routing strategy	Fair-queued
Outgoing routing strategy	N/A
HWM action	N/A

ØMQ

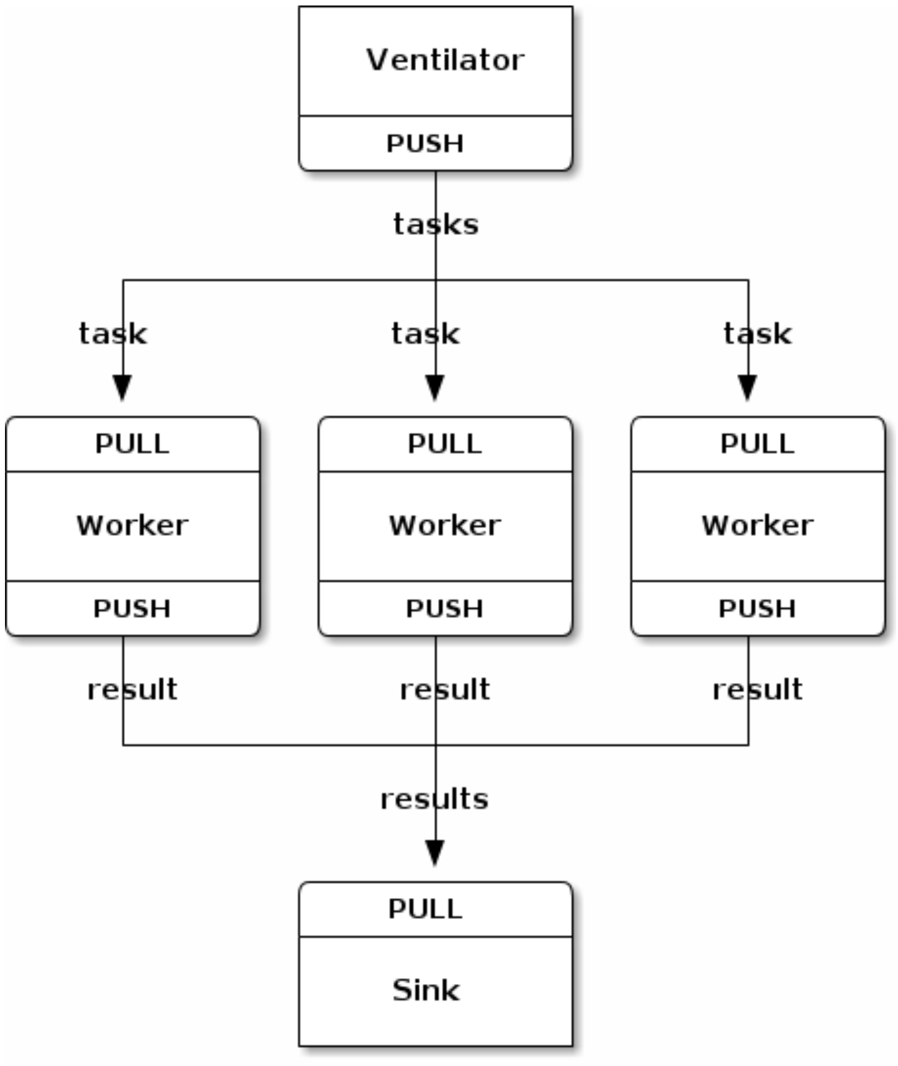


Figure 5 – Parallel Pipeline

ØMQ

```
# pub.py - Publish weather data for multiple zipcodes
import zmq
import random

context = zmq.Context()
socket = context.socket(zmq.PUB)
socket.bind("tcp://*:5556")

while True:
    zipcode = random.randrange(10000,11000)
    temperature = random.randrange(1,215) - 80
    relhumidity = random.randrange(1,50) + 10

    socket.send("%d %d %d" % (zipcode, temperature, relhumidity))
```

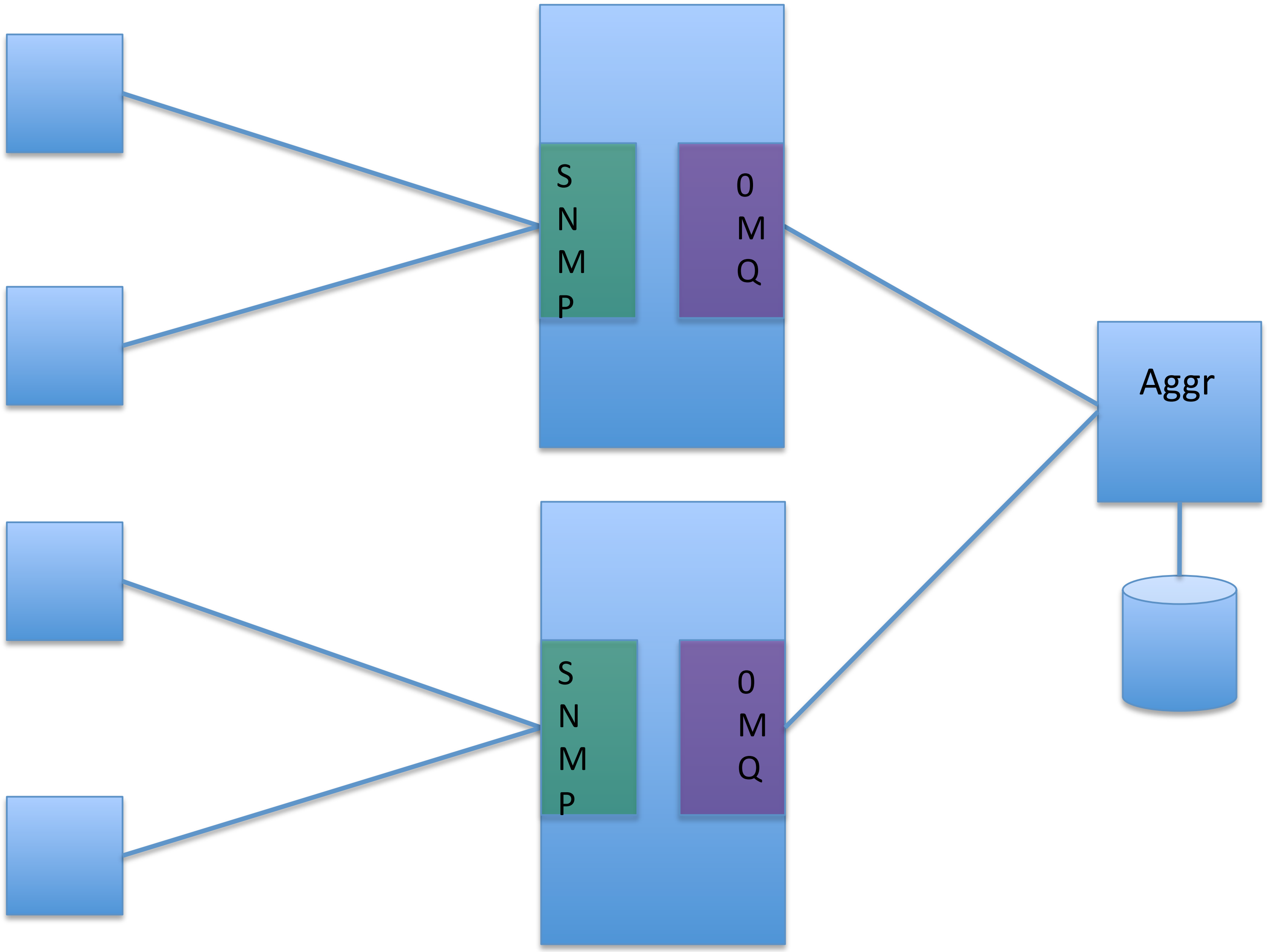
ØMQ

```
# sub.py - Subscribe to weather data for a given zipcode
import sys
import zmq

context = zmq.Context()
socket = context.socket(zmq.SUB)

print "Collecting updates from weather server..."
socket.connect ("tcp://localhost:5556")

# Subscribe to zipcode, default is NYC, 10001
filter = sys.argv[1] if len(sys.argv) > 1 else "10001"
socket.setsockopt(zmq.SUBSCRIBE, filter)

# Process 5 updates
total_temp = 0
for update_nbr in range (5):
    string = socket.recv()
    zipcode, temperature, relhumidity = string.split()
    total_temp += int(temperature)

print "Average temperature for zipcode '%s' was %dF" % (
      filter, total_temp / update_nbr)
```

ØMQ

```
[neale@fedora ~]$ python pub.py &
[neale@fedora ~]$ python sub.py 10200 &
[neale@fedora ~]$ python sub.py 10300 &
[neale@fedora ~]$ python sub.py 10400 &
[neale@fedora ~]$ python sub.py 10500 &
Collecting updates from weather server...
Collecting updates from weather server...
Collecting updates from weather server...
Collecting updates from weather server...
Average temperature for zipcode '10400' was 16F
Average temperature for zipcode '10500' was 18F
Average temperature for zipcode '10200' was 30F
Average temperature for zipcode '10300' was 15F
```

ØMQ



CMIS

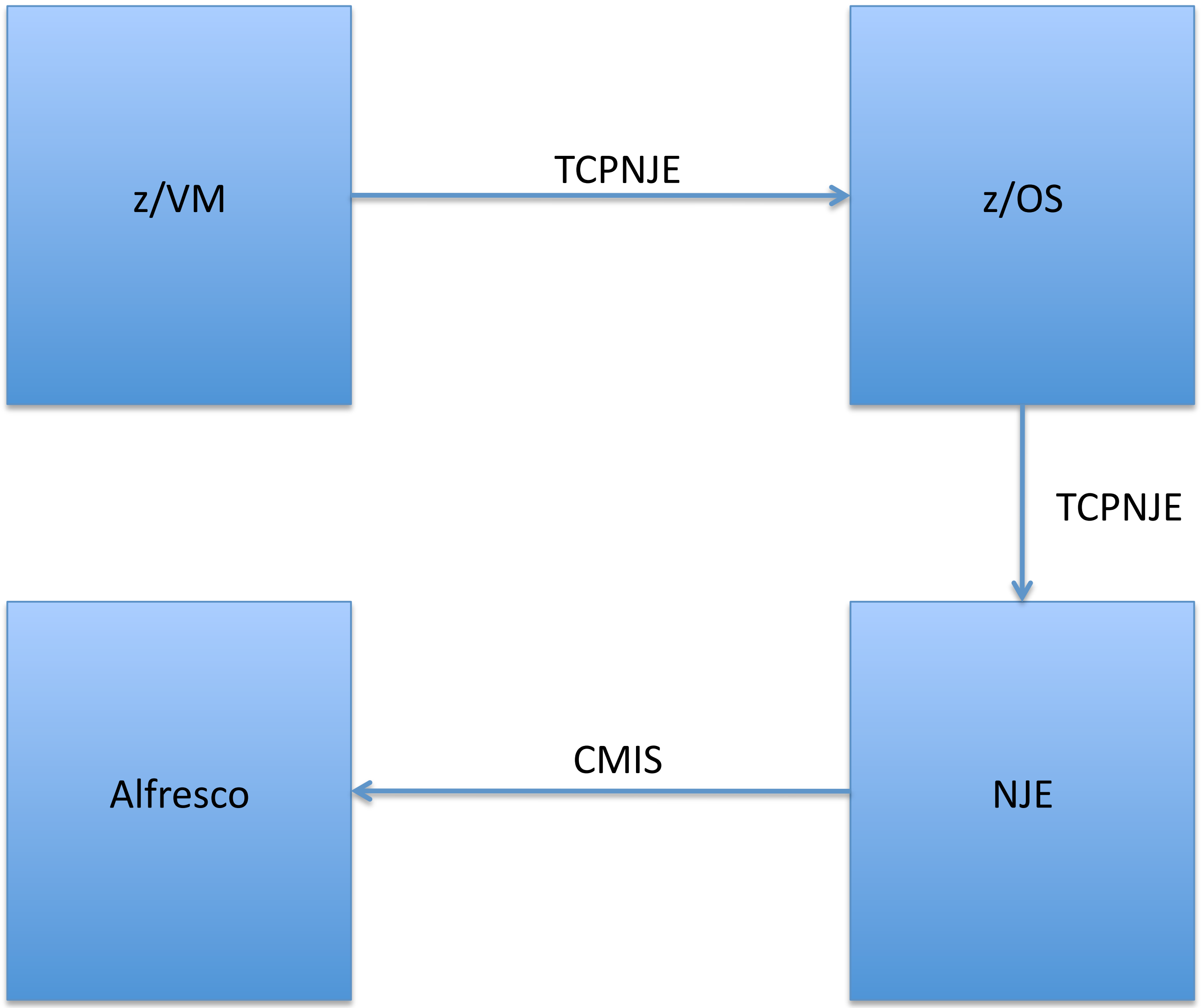
A Protocol for Accessing and manipulating ECM Systems

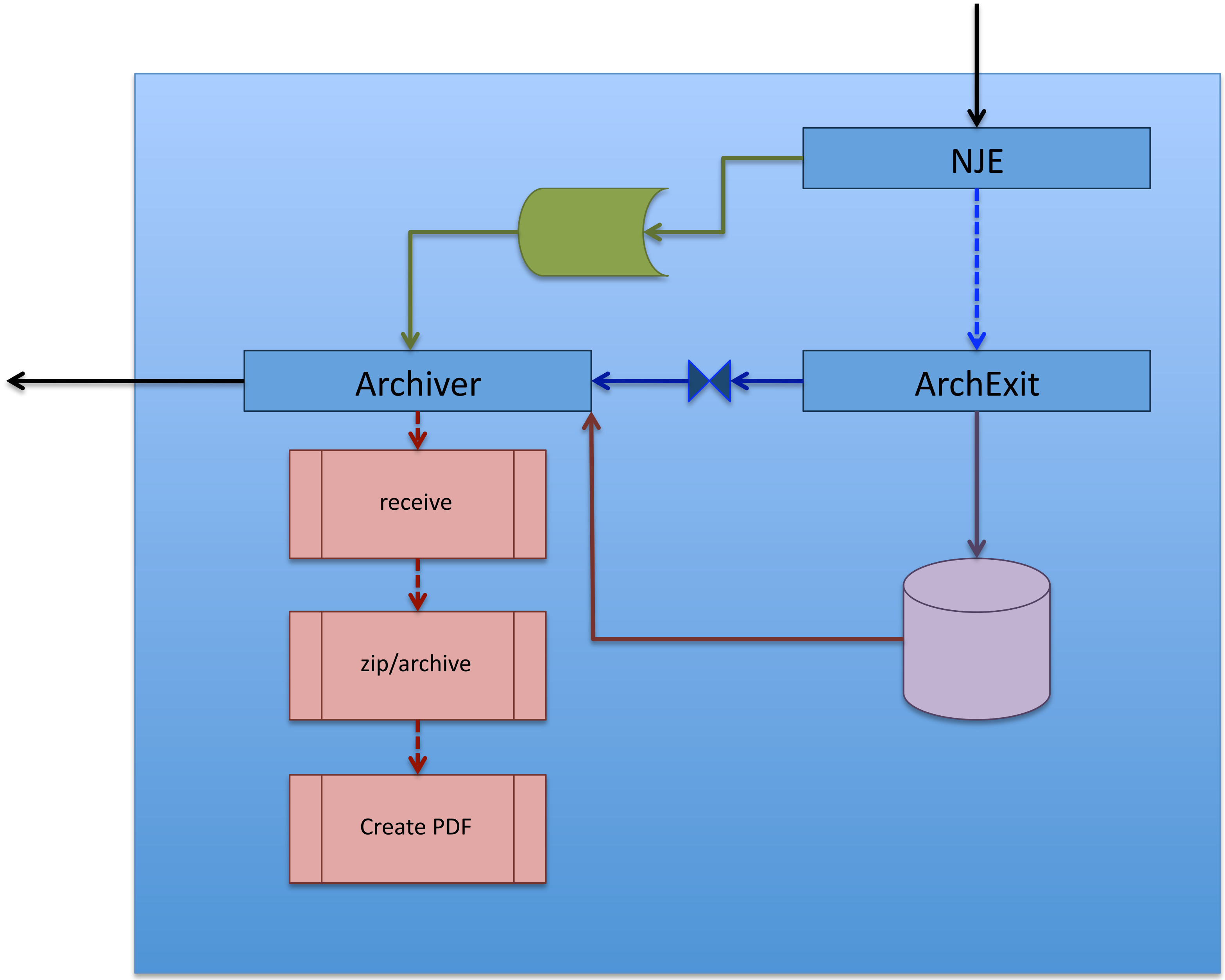
CMIS

- A specification for improving interoperability between Enterprise Content Management systems
- OASIS specification
- Participants include Liferay, Alfresco, eXo, Day Software, EMC, FatWire, IBM, Microsoft, Open Text, Oracle and SAP

CMIS

- Is language-agnostic (REST and SOAP are implemented in many languages)
- Decouples web service and content: CMIS can be used to access to an historic document repository

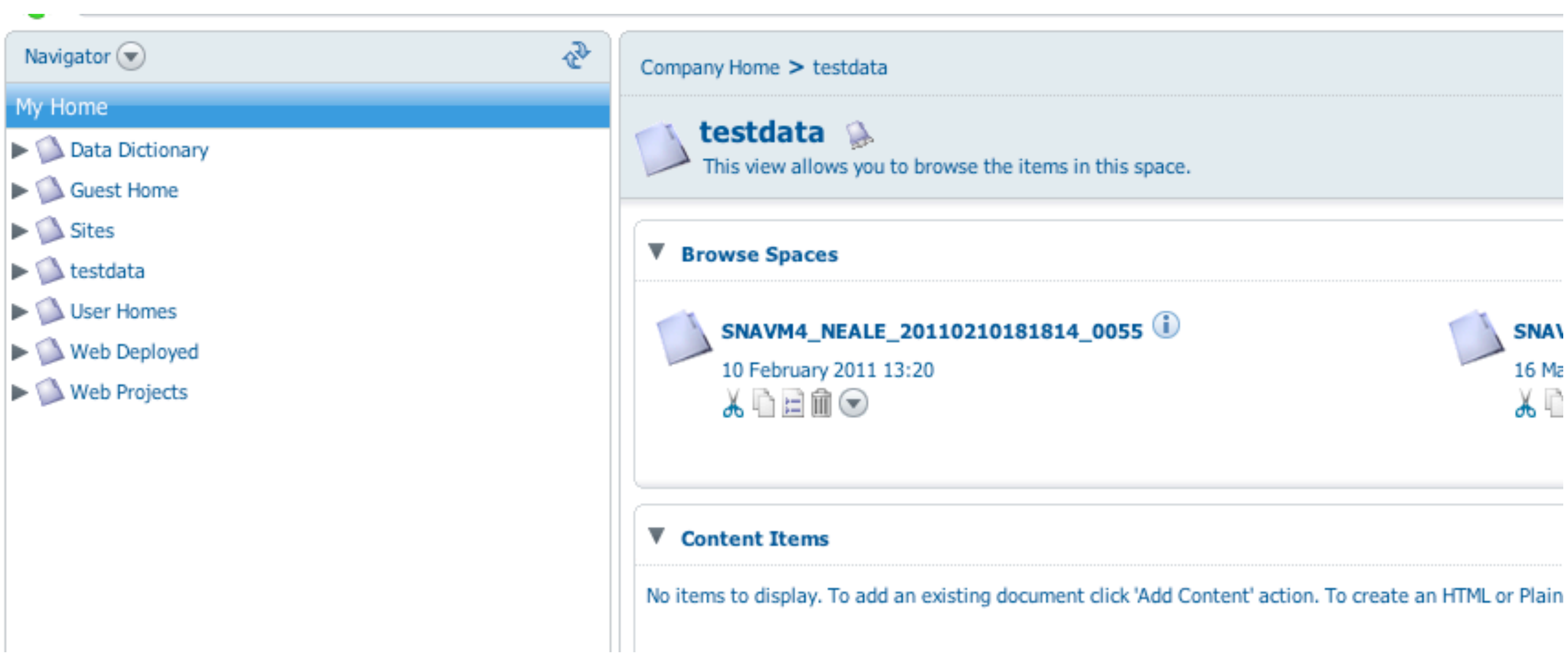

z/VM
TCPNJE
z/OS
TCPNJE
NJE
CMIS
Alfresco


NJE
Archiver
ArchExit
receive
zip/archive
Create PDF

```
#--------------------------------------------------------------#
# initialize the CMIS client object                            #
#--------------------------------------------------------------#
client = CmisClient(UrlCmisService, user_id, password)
repo = client.defaultRepository
```

```
def CreateCmisFolderIfItDoesNotExist(targetFolderObject, newFolderName):
    #----------------------------------------------------------#
    # first lets find out if a folder already exists by this   #
    # name (newFolderName)                                     #
    #----------------------------------------------------------#
    children = targetFolderObject.getChildren()
    for child in children:
        if (child.name == newFolderName):
            return child
    logger.debug("Creating folder " + newFolderName)
    return targetFolderObject.createFolder(newFolderName)
```

```
props = createPropertyBag(propBag, targetClass)
f = open(docLocalPath, 'rb')
newDoc = folder.createDocument(docName, props, contentFile=f)
logger.debug("Cmislib create returned id=" + newDoc.id)
f.close()
```



Navigator

My Home

Data Dictionary

Guest Home

Sites

testdata

User Homes

Web Deployed

Web Projects

Company Home > testdata

testdata

This view allows you to browse the items in this space.

Browse Spaces

SNAVM4_NEALE_20110210181814_0055

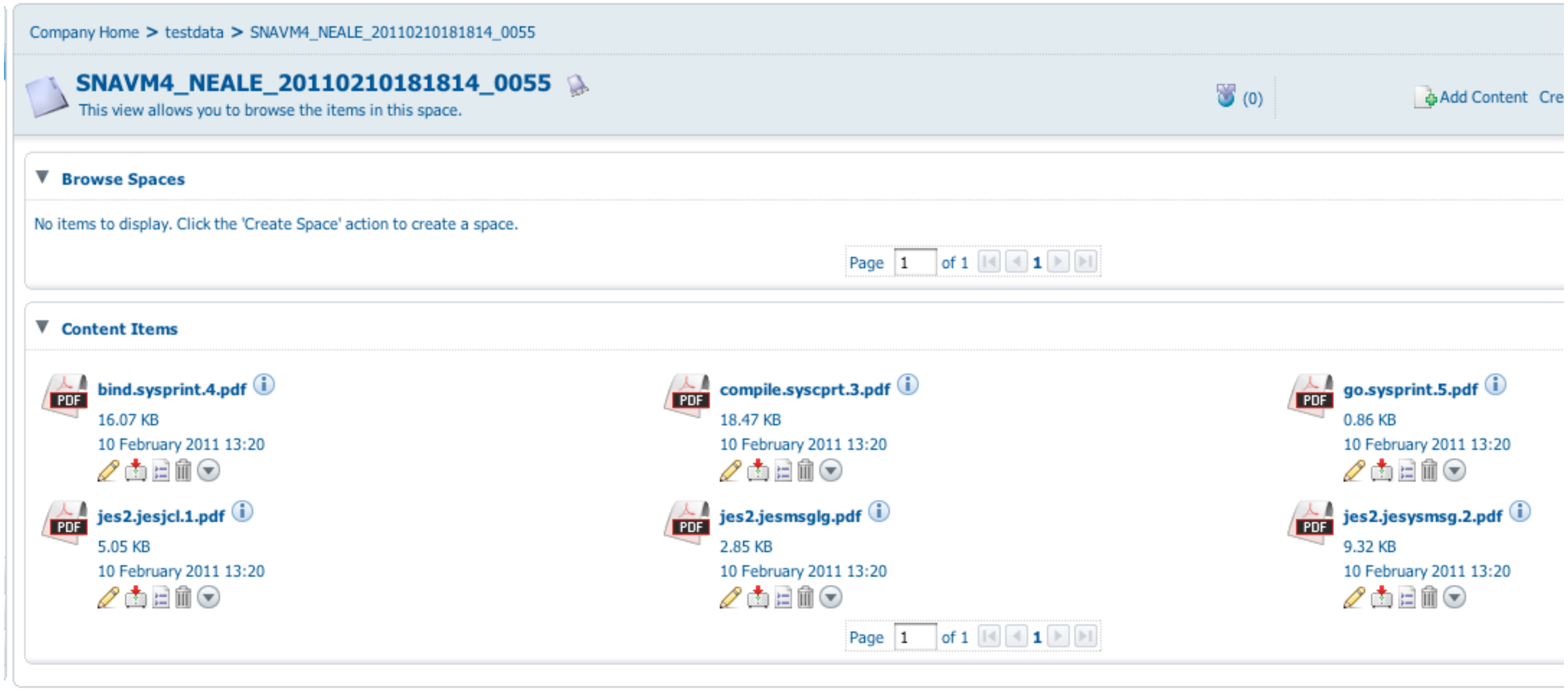
10 February 2011 13:20

SNA

16 Ma

Content Items

No items to display. To add an existing document click 'Add Content' action. To create an HTML or Plain


Company Home > testdata > SNAVM4_NEALE_20110210181814_0055

SNAVM4_NEALE_20110210181814_0055

This view allows you to browse the items in this space.

(0)

Add Content Cre

▼ Browse Spaces

No items to display. Click the 'Create Space' action to create a space.

Page 1 of 1 1

▼ Content Items

bind.sysprint.4.pdf 16.07 KB 10 February 2011 13:20	compile.syscprt.3.pdf 18.47 KB 10 February 2011 13:20	go.sysprint.5.pdf 0.86 KB 10 February 2011 13:20
jes2.jesjcl.1.pdf 5.05 KB 10 February 2011 13:20	jes2.jesmsglg.pdf 2.85 KB 10 February 2011 13:20	jes2.jesysmsg.2.pdf 9.32 KB 10 February 2011 13:20

Page 1 of 1 1

```
z/OS V1 R9 BINDER     12:18:02 THURSDAY FEBRUARY 10, 2011
BATCH EMULATOR  JOB(CC64BLD ) STEP(BIND    ) PGM= IEWL
IEW2278I B352 INVOCATION PARAMETERS - MAP,RENT,DYNAM=DLL,CASE=MIXED,LIST=NOIMP

IEW2322I 1220  1   ******************************************************************/
IEW2322I 1220  2   **                                                              */
IEW2322I 1220  3   ** CELQS003                                                     */
IEW2322I 1220  4   **                                                              */
IEW2322I 1220  5   ** LICENSED MATERIALS - PROPERTY OF IBM                         */
IEW2322I 1220  6   **                                                              */
IEW2322I 1220  7   ** 5694-A01 5688-198                                            */
IEW2322I 1220  8   **                                                              */
IEW2322I 1220  9   ** (C) COPYRIGHT IBM CORP. 2004, 2007                           */
IEW2322I 1220  10   **                                                              */
IEW2322I 1220  11   ** US GOVERNMENT USERS RESTRICTED RIGHTS - USE,                 */
IEW2322I 1220  12   ** DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP              */
IEW2322I 1220  13   ** SCHEDULE CONTRACT WITH IBM CORP                              */
IEW2322I 1220  14   **                                                              */
IEW2322I 1220  15   ** STATUS = HLE7740                                             */
IEW2322I 1220  16   **                                                              */
IEW2322I 1220  17   ******************************************************************/
IEW2322I 1220  18   *
IEW2322I 1220  19   * These statements allow the application to share
IEW2322I 1220  20   * external functions defined by/within the C library.
IEW2322I 1220  21   *
IEW2322I 1220  5623   *
IEW2322I 1220  5624   * These statements allow the application to share the
IEW2322I 1220  5625   * external storage class (global) variables.
IEW2322I 1220  5626   * These variables are defined by/within the C library.
IEW2322I 1220  5627   *
IEW2322I 1220  5646     NAME TESTC(R)
IEW2617I 4C43 DEFINITION SIDE FILE IS EMPTY. THERE ARE NO SYMBOLS TO BE EXPORTED.
```