

Progress and Sonic

Developing a
Sonic Messaging
based solution

What are you doing with Progress Software products?

PROGRESS
SOFTWARE

Progress
OpenEdge

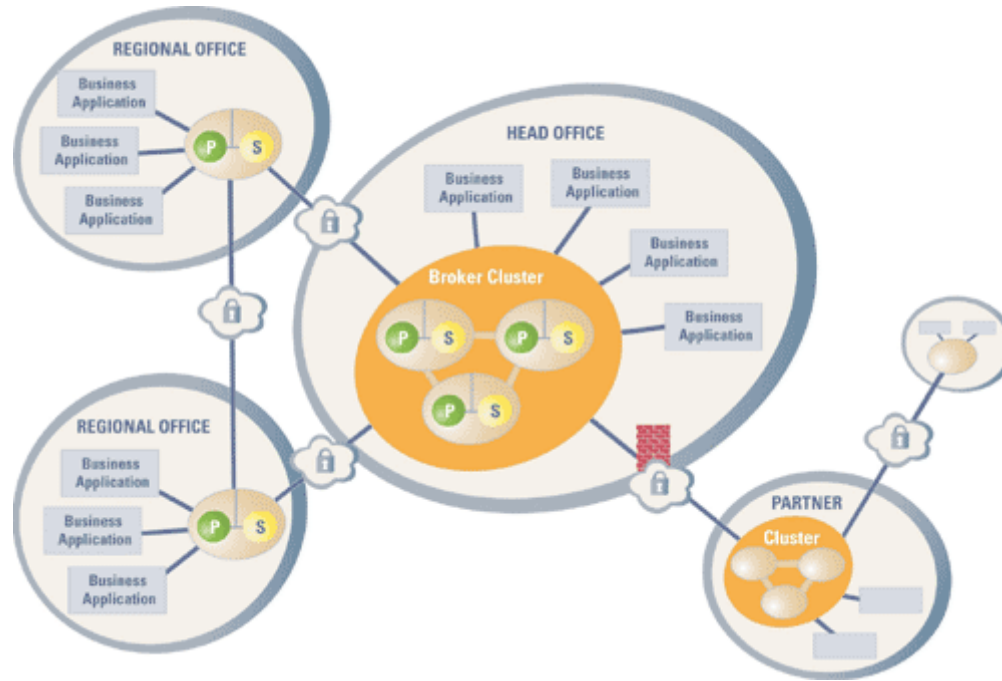
Progress
Sonic

Progress
Apama

Progress
EasyAsk

Progress
ObjectStore

Progress
DataXtend

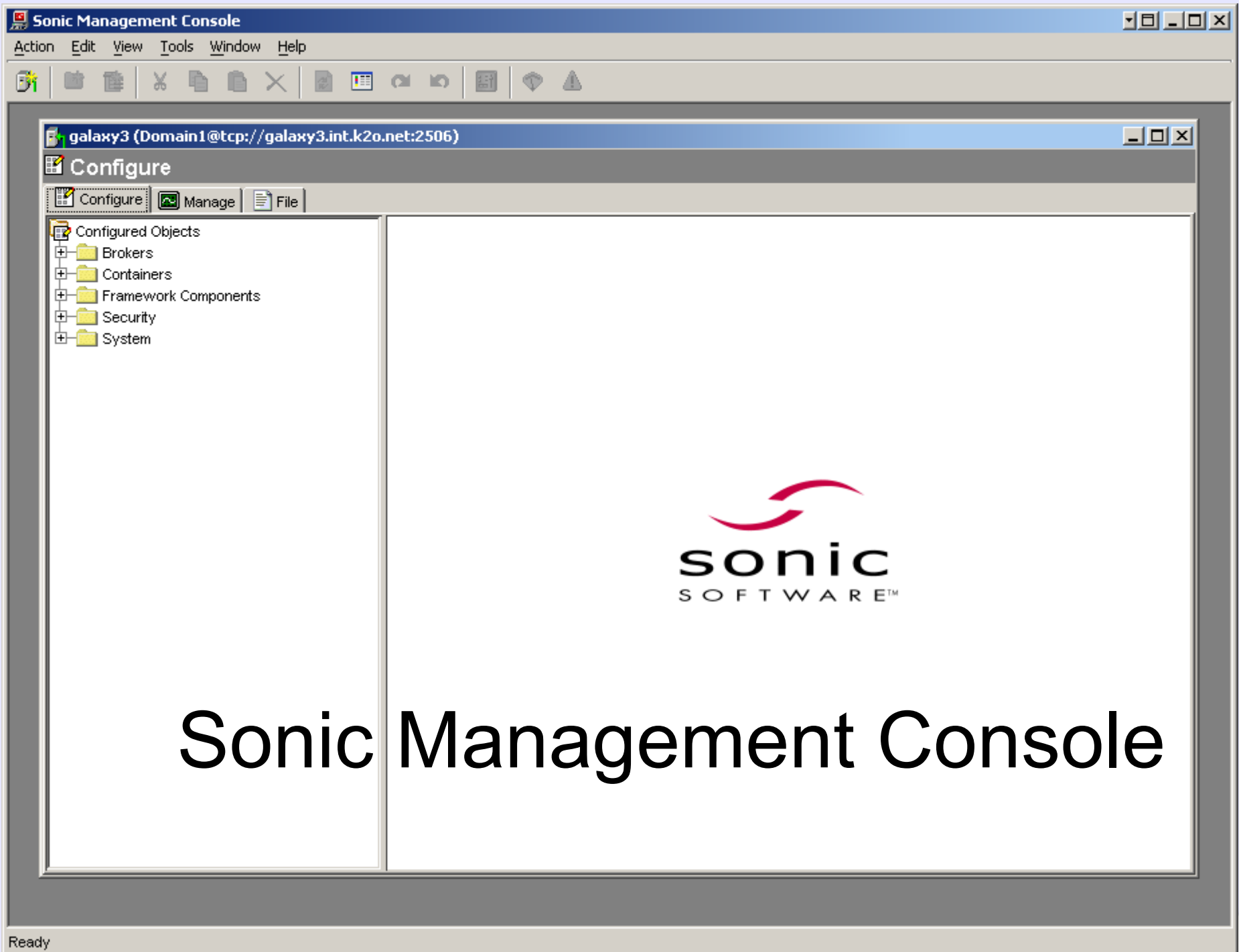


Some Sonic Messaging Uses

- Event Logging
- Interprocess communication
- Publication of data/events (Pub/Sub)
- Data Sync with Software systems
- Routing of data to remote systems
- Async gateway to an ESB
- Provide data stream for CEP(Complex Event Processing) products like Apama

Setup Sonic Messaging

- Create a Container
- Create a Broker
- A Broker needs an Acceptor
- Queues need to be defined
- Topics are created dynamically
- Define Routing to other brokers
- Create an Activation Daemon to start the Broker
- Create Users
- Create Policies (ACLs) to allow routing
- Warning: Everything is CASE sensitive!



Sonic Management Console



Sonic Management Console

Action Edit View Tools Window Help

galaxy3 (Domain1@tcp://galaxy3.int.k2o.net:2506)

Configure - /

Configure Manage File

Configured Objects

- Brokers
 - BR_ManuvL1
 - Acceptors
 - Queues
 - Replication Connections
 - Routing
 - Definitions
 - Global Subscriptions
 - Broker1
- Containers
 - Container1
 - ManuvL1
- Framework Components
 - AD_ManuvL1
 - AGENT MANAGER
 - DIRECTORY SERVICE
- Security
 - Default Authentication
 - Groups
 - Users
 - Default Policies
 - ACLs
 - QOPs
- System
 - SMC

Name	Type
Brokers	Folder
Containers	Folder
Framework Components	Folder
Security	Folder
System	Folder

Ready

Component View



Sonic Management Console

galaxy3 (Domain1@tcp://galaxy3.int.k2o.net:2506)

Configure - /Containers/ManuvisL1

Configure Manage File

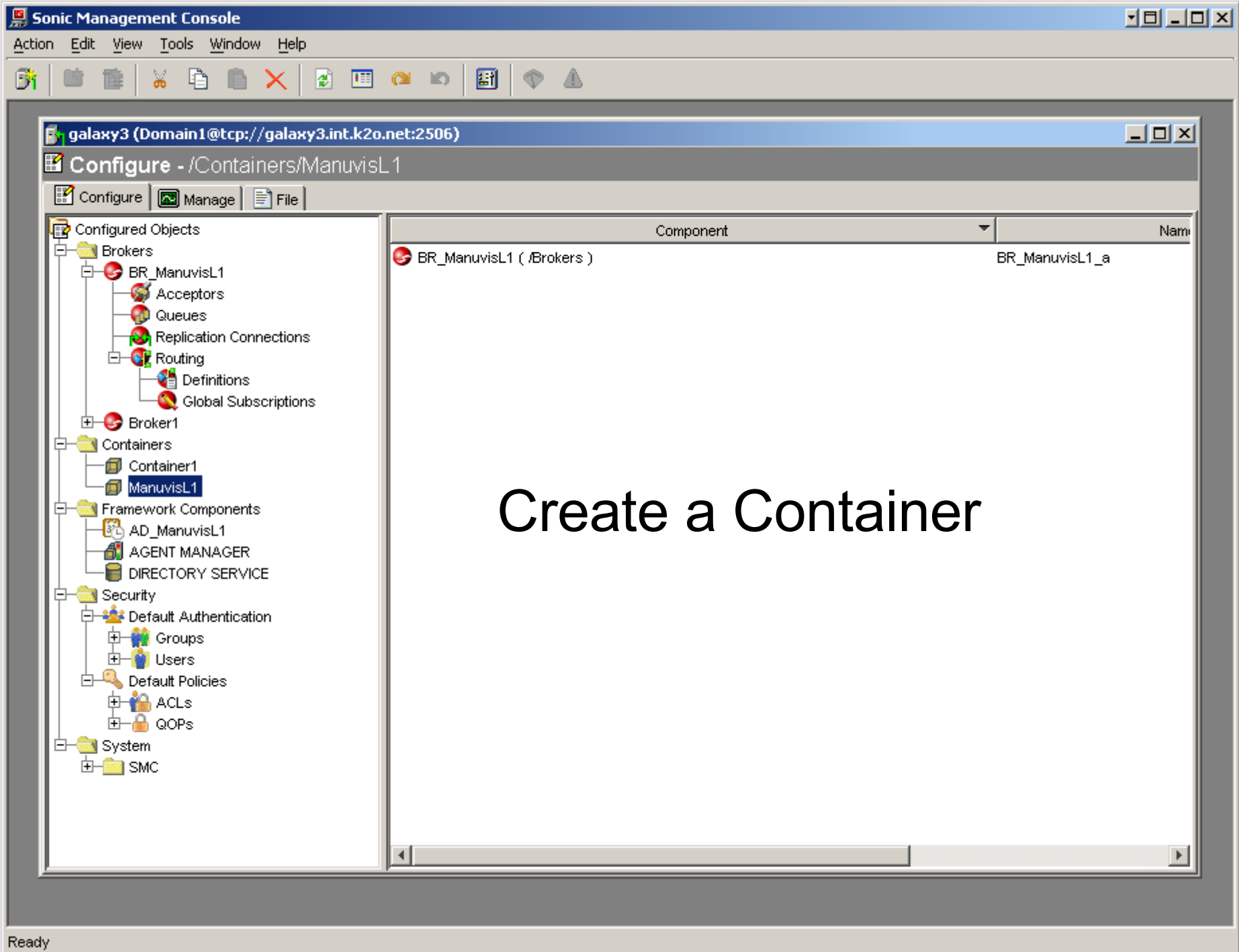
Configured Objects

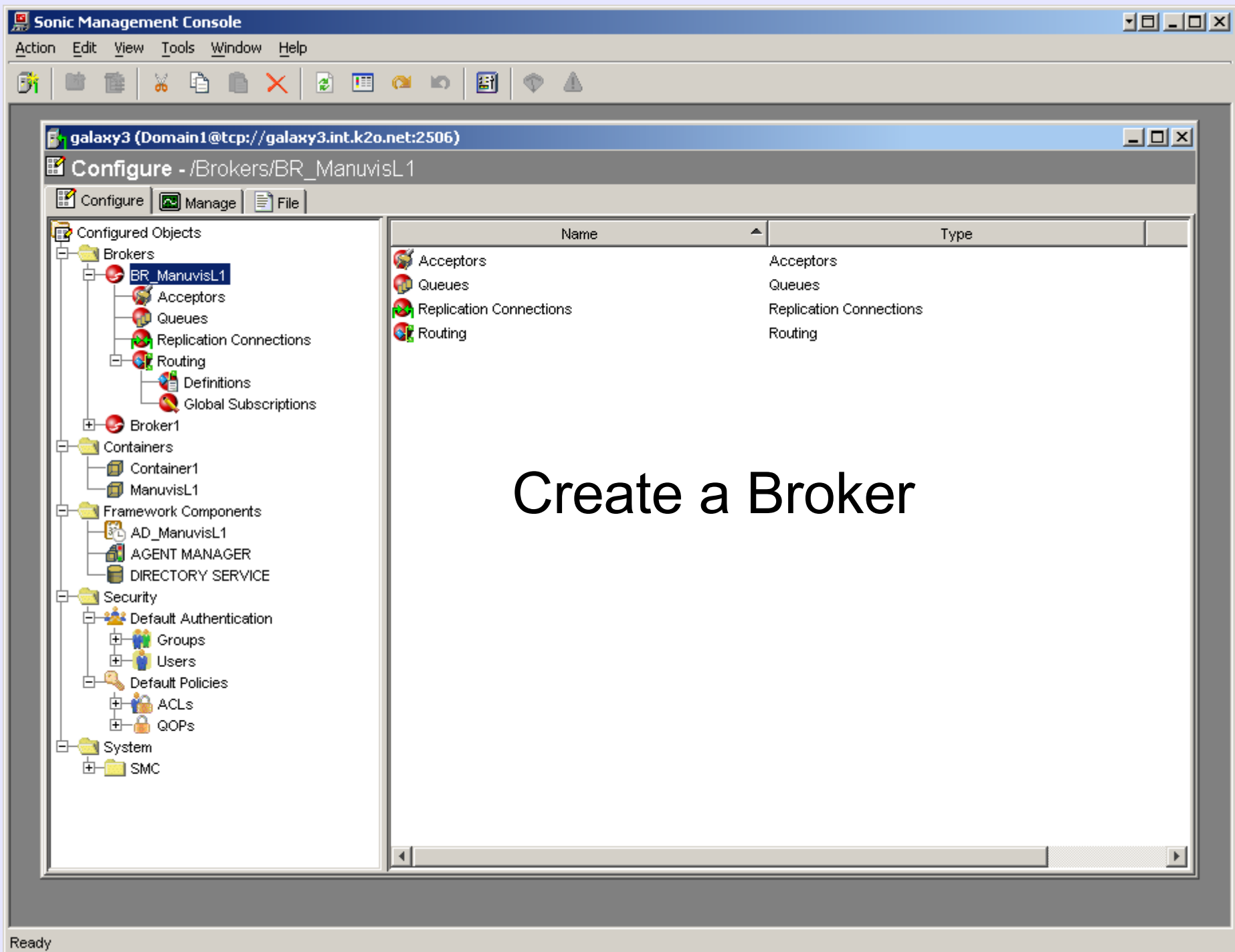
- Brokers
 - BR_ManuvisL1
 - Acceptors
 - Queues
 - Replication Connections
 - Routing
 - Definitions
 - Global Subscriptions
 - Broker1
- Containers
 - Container1
 - ManuvisL1**
- Framework Components
 - AD_ManuvisL1
 - AGENT MANAGER
 - DIRECTORY SERVICE
- Security
 - Default Authentication
 - Groups
 - Users
 - Default Policies
 - ACLs
 - QOPs
- System
 - SMC

Component	Name
BR_ManuvisL1 (/Brokers)	BR_ManuvisL1_a

Create a Container

Ready





Sonic Management Console

Action Edit View Tools Window Help

galaxy3 (Domain1@tcp://galaxy3.int.k2o.net:2506)

Configure - /Brokers/BR_ManuvisL1/Acceptors

Configure Manage File

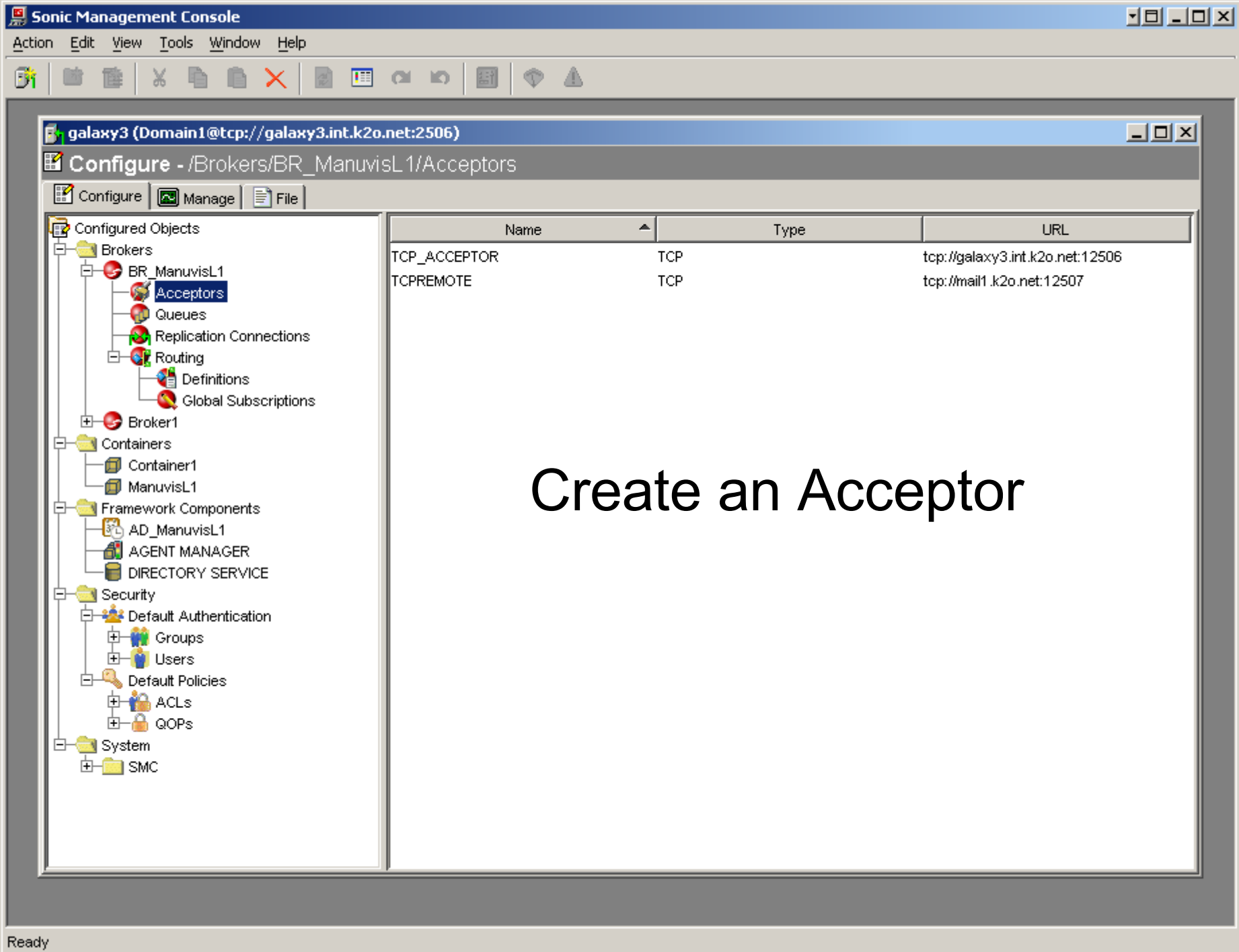
Configured Objects

- Brokers
 - BR_ManuvisL1
 - Acceptors
 - Queues
 - Replication Connections
 - Routing
 - Definitions
 - Global Subscriptions
 - Broker1
- Containers
 - Container1
 - ManuvisL1
- Framework Components
 - AD_ManuvisL1
 - AGENT MANAGER
 - DIRECTORY SERVICE
- Security
 - Default Authentication
 - Groups
 - Users
 - Default Policies
 - ACLs
 - QOPs
- System
 - SMC

Name	Type	URL
TCP_ACCEPTOR	TCP	tcp://galaxy3.int.k2o.net:12506
TCPREMOTE	TCP	tcp://mail1.k2o.net:12507

Create an Acceptor

Ready



Sonic Management Console

galaxy3 (Domain1@tcp://galaxy3.int.k2o.net:2506)

Configure - /Brokers/BR_ManuvisL1/Queues

Configure Manage File

Configured Objects

- Brokers
 - BR_ManuvisL1
 - Acceptors
 - Queues
 - Replication Connections
 - Routing
 - Definitions
 - Global Subscriptions
 - Broker1
- Containers
 - Container1
 - ManuvisL1
- Framework Components
 - AD_ManuvisL1
 - AGENT MANAGER
 - DIRECTORY SERVICE
- Security
 - Default Authentication
 - Groups
 - Users
 - Default Policies
 - ACLs
 - QOPs
- System
 - SMC

Name	Global	Exclusive	Save Threshold	Maximum Size
FMRI.EVENT.ACTION	No	No	1536	20480
FMRI.EVENT.CHECK	No	No	1536	20480
FMRI.EVENT.JOBCALC	No	No	1536	20480
FMRI.EVENT.JOBEND	Yes	No	1536	20480
FMRI.EVENT.LOGOUT	Yes	No	1536	20480
FMRI.EVENT.NEW	No	No	1536	20480
FMRI.EVENT.PROCESS	No	No	1536	20480
FMRI.EVENT.TEST	Yes	No	1536	20480
IAFCount	Yes	No	1536	40960
IAFParmData	Yes	No	1536	40960
IAFReply	Yes	No	1536	20480
IAFRequest	Yes	No	1536	20480
SampleQ1	Yes	No	1536	20480
SampleQ2	No	No	1536	1000
SampleQ3	No	No	1536	1000
SampleQ4	No	No	1536	1000
SonicMQ.deadMessage	No	No	1536	16384
SonicMQ.routingQueue	No	No	1536	1024

Create Queues

Ready

Sonic Management Console

galaxy3 (Domain1@tcp://galaxy3.int.k2o.net:2506)

Configure - /Brokers/BR_ManuvisL1/Routing/Definitions

Configure Manage File

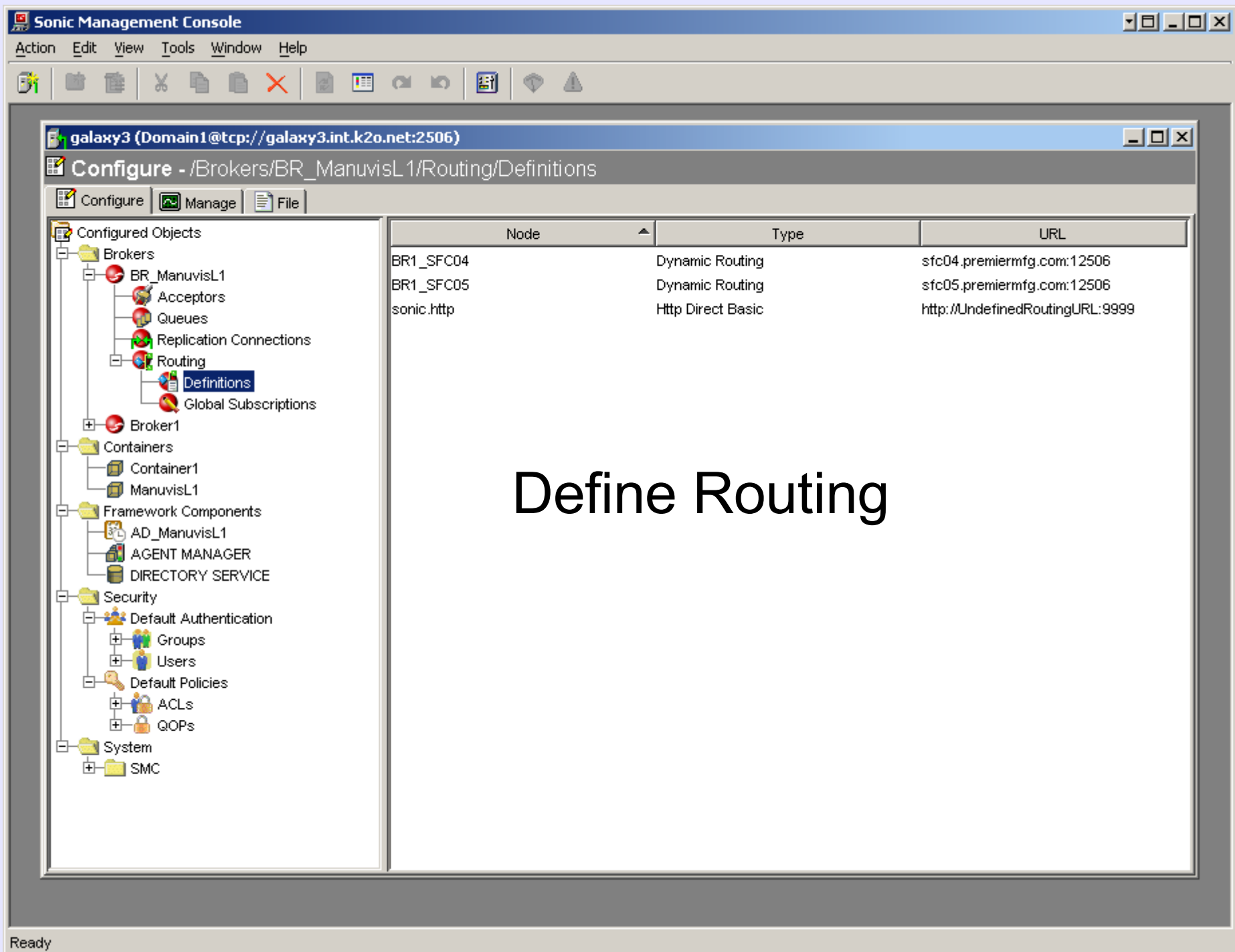
Configured Objects

- Brokers
 - BR_ManuvisL1
 - Acceptors
 - Queues
 - Replication Connections
 - Routing
 - Definitions
 - Global Subscriptions
 - Broker1
- Containers
 - Container1
 - ManuvisL1
- Framework Components
 - AD_ManuvisL1
 - AGENT MANAGER
 - DIRECTORY SERVICE
- Security
 - Default Authentication
 - Groups
 - Users
 - Default Policies
 - ACLs
 - QOPs
- System
 - SMC

Node	Type	URL
BR1_SFC04	Dynamic Routing	sfc04.premiermfg.com:12506
BR1_SFC05	Dynamic Routing	sfc05.premiermfg.com:12506
sonic.http	Http Direct Basic	http://UndefinedRoutingURL:9999

Define Routing

Ready



Sonic Management Console

galaxy3 (Domain1@tcp://galaxy3.int.k2o.net:2506)

Configure - /Framework Components/AD_ManuvisL1

Configure Manage File

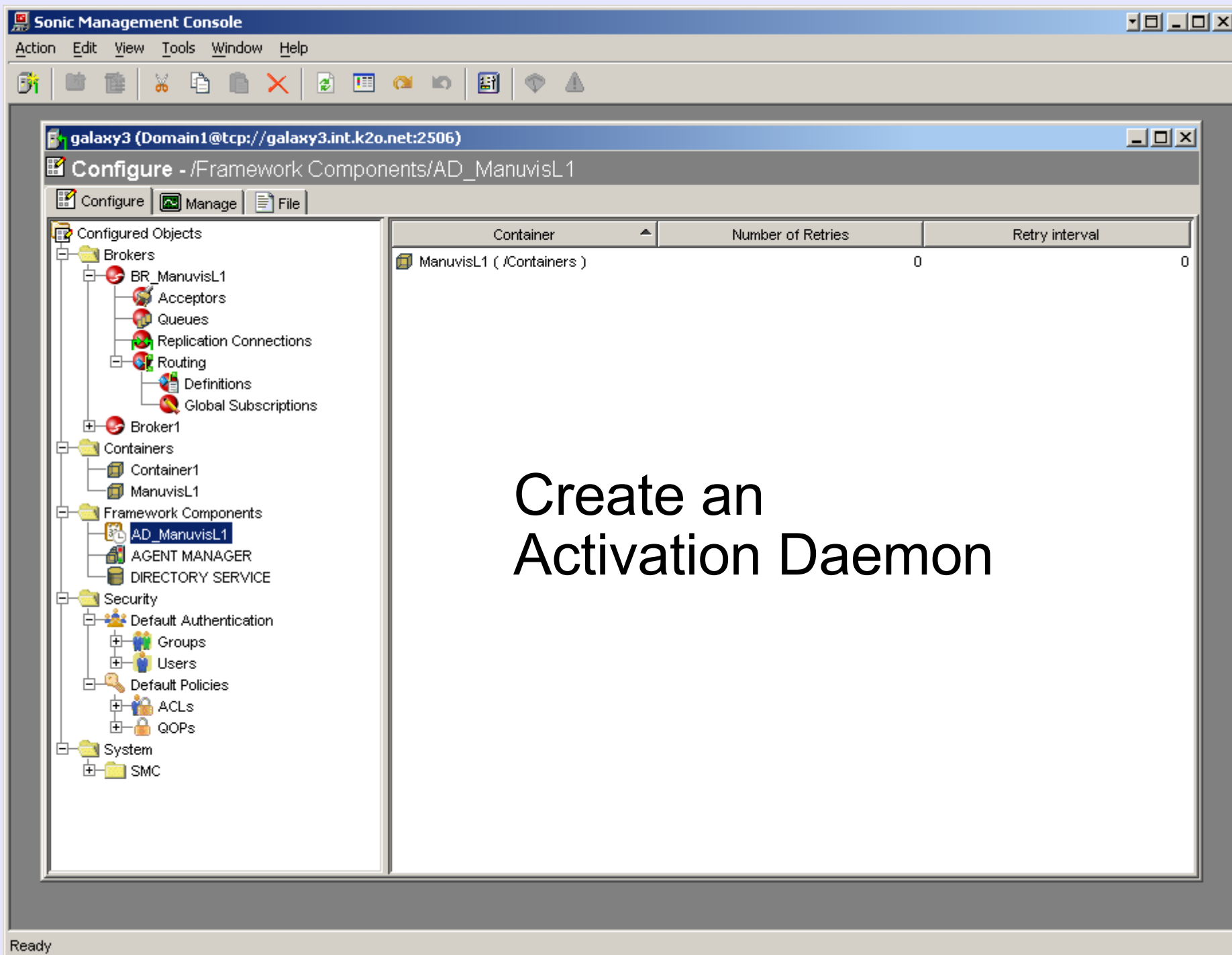
Configured Objects

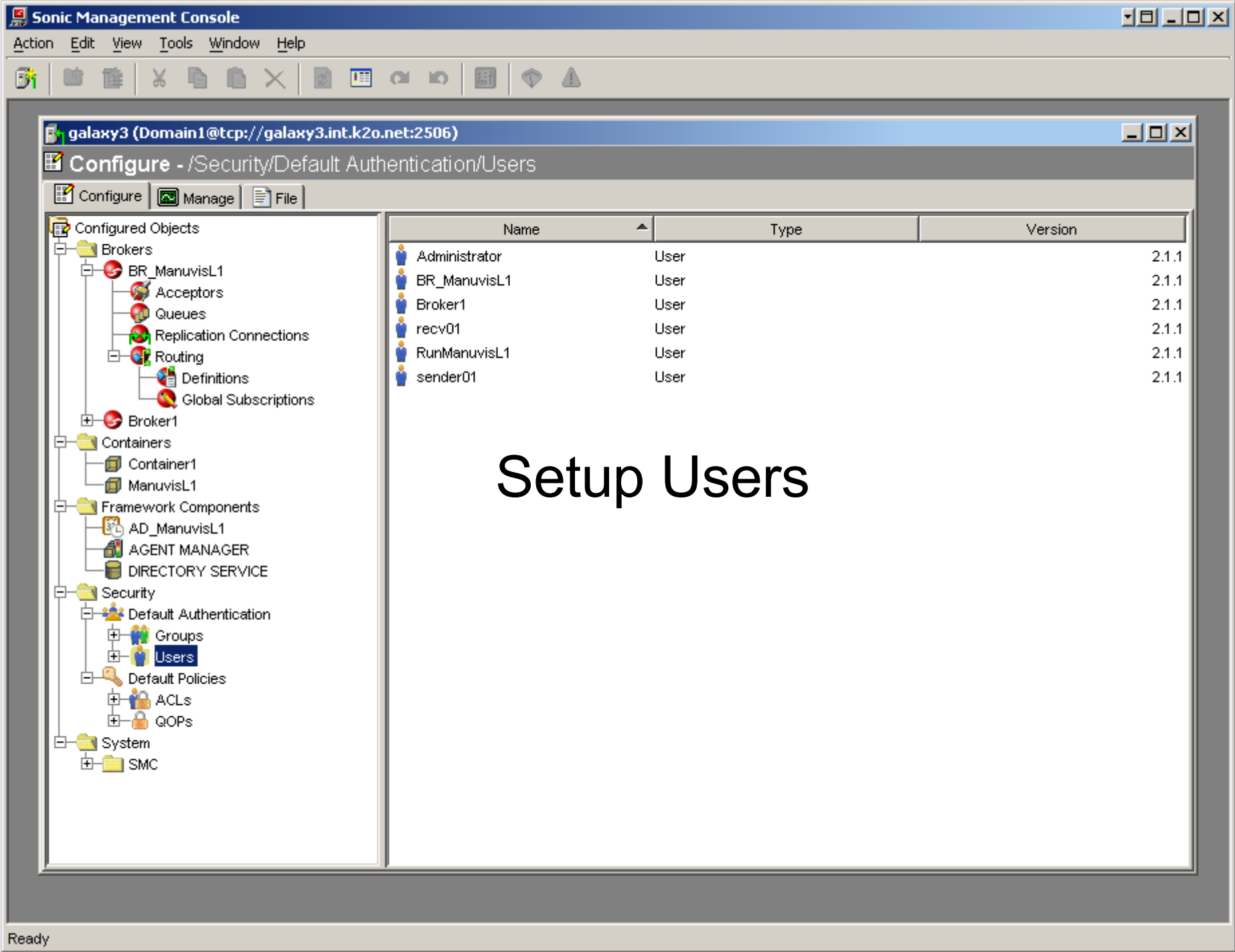
- Brokers
 - BR_ManuvisL1
 - Acceptors
 - Queues
 - Replication Connections
 - Routing
 - Definitions
 - Global Subscriptions
 - Broker1
- Containers
 - Container1
 - ManuvisL1
- Framework Components
 - AD_ManuvisL1
 - AGENT MANAGER
 - DIRECTORY SERVICE
- Security
 - Default Authentication
 - Groups
 - Users
 - Default Policies
 - ACLs
 - QOPs
- System
 - SMC

Container	Number of Retries	Retry interval
ManuvisL1 (/Containers)	0	0

Create an Activation Daemon

Ready





Review: Setup Sonic Messaging

- Create a Container
- Create a Broker
- A Broker needs an Acceptor
- Queues need to be defined
- Topics are created dynamically
- Define Routing to other brokers
- Create an Activation Daemon to start the Broker
- Create Users
- Create Policies (ACLs) to allow routing
- Warning: Everything is CASE sensitive!

The Sending Side

Write record data to a Queue

Need procedures for:

- Connecting to the Broker
- Writing the XML data to a Queue
- Session Trigger to trap the event

Start the Session

```
PROCEDURE start-mq-session:
  /* Creates a session object. */
  RUN  jms/jmssession.p PERSISTENT SET pubsubsession (jms_conn).
  IF ERROR-STATUS:ERROR THEN RETURN.

  RUN  setBrokerURL IN pubsubsession (br_conn) NO-ERROR.
  IF ERROR-STATUS:ERROR THEN RETURN.

  RUN  setUser in pubsubsession(my_recv_user).
  RUN  setPassword in pubsubsession(my_recv_pwd).
  RUN  setClientID in pubsubsession(myclient_id).

  RUN  beginSession IN pubsubsession NO-ERROR.
  IF ERROR-STATUS:ERROR THEN
    DO:
      ASSIGN
        jmsIsOk = NO
      RETURN.
    END.

  DEFINE VARIABLE conn-id AS CHARACTER NO-UNDO.
  ASSIGN
    conn-id = DYNAMIC-FUNCTION('getConnectionId':U IN pubsubsession) NO-ERROR
  ASSIGN
    jmsIsOk = YES
  END PROCEDURE. /* start-mq-session */
```



Send the XML

```
FUNCTION f-event-send-xml-to-queue RETURNS LOGICAL
( INPUT z-params AS CHARACTER EXTENT,
  INPUT z-xml AS MEMPTR,
  INPUT q-dest AS CHARACTER )
:
DEFINE VARIABLE q-msgtype AS CHARACTER NO-UNDO.
DEFINE VARIABLE q-ttl AS INTEGER NO-UNDO.
DEFINE VARIABLE q-dmode AS CHARACTER NO-UNDO.
DEFINE VARIABLE q-priority AS INTEGER NO-UNDO.
DEFINE VARIABLE mesgH AS HANDLE NO-UNDO.

ASSIGN
  q-msgtype = z-params[3]
  q-ttl = 60 * 20 * 1000
  q-dmode = "PERSISTENT":U
  q-priority = 0
.

IF EXTENT(z-params) >= 4 THEN
DO:
  ASSIGN
    q-ttl = INTEGER( z-params[4] ) NO-ERROR
  .
END.

RUN createBytesMessage IN mymqsession (OUTPUT mesgH).
RUN setStringProperty IN mesgH ( "PRODUCER":U, PROGRAM-NAME(1) ).
RUN setJMSType IN mesgH (q-msgtype).
RUN setMemptr IN mesgH(z-xml, ?, ?).
RUN sendToQueue IN mymqsession ( q-dest, mesgH, q-priority, q-ttl, q-dmode ) NO-ERROR.
RUN deleteMessage IN mesgH NO-ERROR.

END FUNCTION. /* f-event-send-xml-to-queue */
```



The Session Trigger

```
ON WRITE OF mytable NEW BUFFER newc2 OLD BUFFER oldc2
DO:
  DEFINE VARIABLE thand AS HANDLE NO-UNDO.
  DEFINE VARIABLE bhand AS HANDLE NO-UNDO.
  DEFINE VARIABLE fname AS CHARACTER NO-UNDO.
  DEFINE VARIABLE destq AS CHARACTER NO-UNDO.
  DEFINE VARIABLE xmem AS MEMPTR NO-UNDO.
  DEFINE VARIABLE p-params AS CHARACTER EXTENT 50 NO-UNDO.

  fname = BUFFER newc2:NAME.
  CREATE TEMP-TABLE thand.
  thand:CREATE-LIKE( fname ).
  thand:TEMP-TABLE-PREPARE( "t_" + fname ).
  bhand = thand:DEFAULT-BUFFER-HANDLE.
  bhand:BUFFER-CREATE().
  BUFFER newc2:RAW-TRANSFER(TRUE, bhand).
  /* we can now write the temp table data to xml and send it via Sonic */
  thand:WRITE-XML("memptr", xmem, YES, ?, ?, YES).
  ASSIGN
    destq = "SOMBROKER::SOMEQ"
    p-params[1] = "":U
    p-params[2] = destq
    p-params[3] = "QPROCMSG":U
    p-params[4] = "0":U /* ttl */
  .
  DYNAMIC-FUNCTION('f-event-send-xml-to-queue':U, p-params, xmem, destq ) NO-ERROR.
  DELETE OBJECT thand NO-ERROR.
  DELETE OBJECT bhand NO-ERROR.
  SET-SIZE(xmem) = 0.
  RETURN.
END.
```



The Receiving Side

Read the data from the Queue

Need procedures for:

- Connecting to the Broker
- Connect to Queue(s)
- Receive messages from Queue
- Process message

Start the Session

```
PROCEDURE start-my-session:
  DEFINE INPUT-OUTPUT PARAMETER mq-session AS HANDLE NO-UNDO.

  RUN jms/jmssession.p PERSISTENT SET mq-session (ps_conn).
  RUN setBrokerURL IN mq-session (br_conn).

  RUN setUser in mq-session(mq_user).
  RUN setPassword in mq-session(mq_pwd).
  RUN setClientID in mq-session(mq_clientid).
  RUN beginSession IN mq-session.

  RUN createMessageConsumer IN mq-session
    (THIS-PROCEDURE, "errorHandler", OUTPUT errorConsumerH).
  RUN setErrorHandler IN mq-session (errorConsumerH).
END PROCEDURE. /* start-my-session */
```

Connect to Queue(s)

```
/* Creates a session object. */
RUN start-my-session( INPUT-OUTPUT mysession_h ).

DEFINE VARIABLE myq2 AS CHARACTER NO-UNDO.
DEFINE VARIABLE cons2H AS HANDLE NO-UNDO.

myq2 = "MYQUEUE".
RUN createMessageConsumer IN mysession_h (
    THIS-PROCEDURE, /* This proc will handle it
*/
    "RecvMessageHandler":U, /* name of internal
procedure */
    OUTPUT cons2H
).

RUN receiveFromQueue IN mysession_h (
    myq2, /* name of topic */
    ?, /* No message selector */
    cons2H). /* Handles the incoming messages*/
```

Receive the Message(s)

```
PROCEDURE RecvmessageHandler:
DEFINE INPUT PARAMETER messageH AS HANDLE.
DEFINE INPUT PARAMETER msgConsumerH AS HANDLE.
DEFINE OUTPUT PARAMETER replyH AS HANDLE.

DEFINE VARIABLE mProp AS CHARACTER NO-UNDO.
DEFINE VARIABLE msgReplyTo AS CHARACTER NO-UNDO.

RUN setReusemessage IN msgConsumerH.
mProp = DYNAMIC-FUNCTION('getPropertyNames':U IN messageH).
msgReplyTo = DYNAMIC-FUNCTION('getJMSReplyTo':U IN messageH).

ReplyH = DYNAMIC-FUNCTION( 'f-process-queue-event-message':U,
    messageH, msgConsumerH, mysession_h ) NO-ERROR.

RUN clearBody IN messageH NO-ERROR.
RUN clearProperties IN messageH NO-ERROR.
END PROCEDURE. /* RecvmessageHandler */
```

Process the message

```
FUNCTION f-process-queue-event-message RETURNS HANDLE
( INPUT messageH AS HANDLE,
  INPUT messageConsumerH AS HANDLE,
  INPUT sesH AS HANDLE ):

DEFINE VARIABLE hTset AS HANDLE NO-UNDO.
DEFINE VARIABLE memptrDoc AS MEMPTR NO-UNDO.

memptrDoc = DYNAMIC-FUNCTION('getMemptr':U IN messageH).
CREATE TEMP-TABLE hTset.
retvalue = hTset:READ-XML( 'memptr':U, memptrDoc, ?, ?, ?).

/* the data is now in a Temp Table - do the work */

DELETE OBJECT hTset NO-ERROR.
SET-SIZE(memptrDoc) = 0.
RUN deletemessage IN messageH.
RETURN replyH.
END FUNCTION. /* f-process-queue-event-msg */
```

Notes

- Sending and receiving processes do not have to be on the same host.
- Use a ProDataset instead of TEMP-TABLE.
- Can become the gateway to an SOA based solution
- CBR (Content Based Routing) can also be used (Progress vs. Sonic setup)

Sonic Notes

- Topics are one to many.
- Queues are one to one.
- Global Subscriptions can be used to route topics automatically.
- Durable subscriptions can make copies of data for each subscriber (topics).
- Set the TTL (Time To Live) correctly
- Extremely Fast
- Needs memory as you scale up the use