STABIL**IT**
Intelligente Informatik

SC

Service Connector

Configuration and Operation Guide

SC_4_Operation_E (Version V1.5)

This document describes topics related to the SC configuration, operation and troubleshooting.

This Document has been created with Microsoft Word 2003 (11) with template file C:\STABILIT\STANDARD\TEMPLATES\S_REP_E.DOT and printed at 30 August 2011 17:20.

Identification

Project:	SC
Title:	Service Connector
Subtitle:	Configuration and Operation Guide
Version:	V1.5
Reference:	SC_4_Operation_E
Classification:	Public
Keywords:	Configuration, Operation, Troubleshooting
Comment:	This document describes topics related to the SC configuration, operation and troubleshooting.
Author(s):	STABILIT Informatik AG Jan Trnka, Daniel Schmutz, Joël Traber
Approval (Reviewed by):	Signature .... Jan Trnka
Audience:	Project team, development team
Distribution:	Public
Filename	c:\stabilit\projects\eurex\sc\documents\sc_4_operation_e.doc

Revision History

Date	Version	Author	Description
11.02.2011	V1.0	Jan Trnka	Initial version
13.04.2011	V1.1	Jan Trnka	Corrections based on external review.
15.04.2011	V1.2	Joël Traber	Added format of function scVersion, serviceConfiguration to inspectCommand.
7.7.2011	V1.3	Jan Trnka	New configuration property added
18.7.2011	V1.4	Jan Trnka	Configuration property changed
16.8.2011	V1.5	Jan Trnka	GUI screen shots updated as of SV V1.3-014 SC.properties and log4j.properties updated.

Table of Contents

Tables

Figures

1   Preface

1.1 Purpose & Scope of this Document

This document describes issues related to the configuration, operation and troubleshooting of the Service Connector. Good understanding of network communication and the Service Connector concepts is prerequisite to this reading.

The SC message protocol is described in [1].

This document is particularly important to the operation & support staff..

1.2 Definitions & Abbreviations

Item / Term	Definition / Description
DMZ	Demilitarized Zone (Network segment between two firewalls)
DNS	Domain Name Server
DOS	Denial Of Service
GUI	Graphical User Interface
HTML	Hypertext Mark-up Language
HTTP	Hypertext Transport Protocol
HTTPS	HTTP over SSL, encrypted and authenticated transport protocol
IE	Internet Explorer – Microsoft Web Browser
J2EE	Java 2 Platform Enterprise Edition
Java	Programming language and run-time environment from SUN
JDK	Java Development Kit
Linux	Unix-like operating system, open source
Log4j	Standard logging tool used in Java
OpenVMS	HP Operating system
OSI	Open System Interconnection reference model http://en.wikipedia.org/wiki/OSI_model
RPC	Remote Procedure Call
SCMP	SC Message Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer – secure communication protocol with encryption and authentication
SC	Service Connector
TCP/IP	Transmission Control Protocol / Internet Protocol
URL	Uniform Resource Locator
Web Service	software designed to support interoperable machine-to-machine interaction over a network
Web-GUI	Graphical User Interface running in Web-Browser built with Web components. Also called Thin-Client.
Windows	Microsoft operating system family
XML	Extensible Mark-up Language
XSD	XML Schema Definition
XSLT	Extensible Style Sheet Language Transformation

Table 1 Abbreviations & Definitions

1.3 External References

References	Item / Reference to other Document
[1]	SC_1_SCMP-V1.1_E

Table 2 External references

1.4 Typographical Conventions

Convention	Meaning
text in italics	*USP feature or functionality or USP related comments*
`text in Courier font`	code example
[phrase]	In syntax diagrams, indicates that the enclosed values are optional
{ phrase1 | phrase2 }	In syntax diagrams, indicates that multiple possibilities exist.
…	In syntax diagrams, indicates a repetition of the previous expression

Table 3 Typographical conventions

The terminology used in this document may be somewhat different to other sources. The chapter Glossary includes a list of often used terms with the explanation of their meaning in this document.

1.5 Outstanding Issues

Following issues are outstanding at the time of the document release:
   1.

2   Concepts

Service Connector is a message oriented middleware connecting applications providing services to other applications consuming the services. It fits into an overall service oriented architecture.

Service Connector supports following service types

2.1 Session Service

Session service allows statefull message exchange between requesting client application and server application implementing the service. All actions are initiated by the client. When the client starts the session, SC allocates a server instance for this session. The server must be ready at this time. Multiple server instances serving the same service can be pre-started. Serving multiple sessions or multiple services within one server instance is possible only for multisession servers.

Clients and servers are **not** connected directly to each other. The client and the server are only the logical communication end-points. The SC acts like a message broker and is never a direct executor of the service. Clients and servers may be distributed across any number of nodes in the network.

All services are independent on each other. Clients and servers may be distributed across any number of nodes in the network. The server application may request another service and so play the client role.
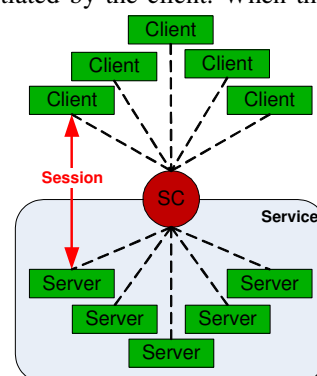
2.2 Publishing Service

Publishing service allows a server to send messages to many clients. In order to receive a published message, the client must subscribe to a service. The subscription mask allows fine granular definition of the client interest.

Clients and servers are **not** connected directly to each other. The application server which implements the service designates the contents of each message with a message mask and sends it to SC. When the message mask matches the client subscription mask, SC will deliver the message to the client. Multiple clients may subscribe for the same service at the same time and may get the same message. The publishing server does not know which client did will receive its messages and when. Sequence order of the published messages is preserved by SC.
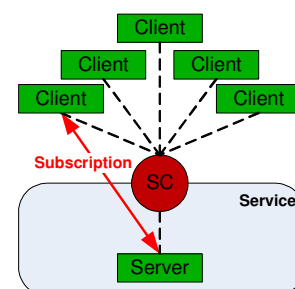
Multiple server instances may publish their messages to the same service. (Not allowed when large messages are published!) All services are independent on each other. Clients and servers may be distributed across any number of nodes in the network. The server application may request another service and so play the client role.

2.3 File Service

File service allows file exchange between client application and a web server. All actions are initiated by the client. The client can upload to a service or download file from a service or get a list of files in this service. Client and the Web Server are **not** connected directly to each other. The web server must define one <u>flat</u> directory for each service. Sub-directory structure is not supported. The same directory may be used by different services. .

All services are independent on each other. Clients and web servers may be distributed across any number of nodes in the network.

PHP is required on the Web Server in order to support file services. The provided PHP upload script may be adapted e.g. for sending notification mail after file upload. See <u>http://www.radinks.com/upload/config.php</u> for PHP parameters related to file upload.
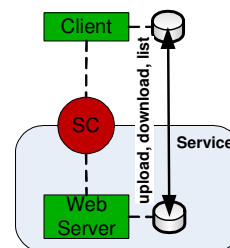
2.4 HTTP Redirection Service

SC HTTP redirection service allows redirection of http traffic through the SC infrastructure to a designated target URL (web server). This is possible without a session or subscription context. It can be used by any application also by a browser. SC receives the http requests on a dedicated port and passes it to the configured server. <u>It does not work like a HTTP proxy</u>. .

Client and the Web Server are **not** connected directly to each other. Clients and targets (web servers) may be distributed across any number of nodes in the network. The requested target URL must be resolved (by DNS or other means) by the underlying network infrastructure. SC does not provide any name resolution.

Multiple http redirection services may be defined to different targets, each one listening on a different TCP/IP port.
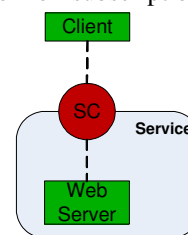
2.5 SC Cascading

Multiple SCs may be running in a network. Any SC service may be defined as cascaded. This means that the message traffic for such service is redirected to another SC. Cascading is fully transparent to the client. Clients can use services at any cascade level. Servers are allowed on the bottom level only.

Major benefits of cascading are:
- Improved security by using cascaded SC in a DMZ as a single access point for all local clients.
- Reduced communication costs by sharing infrastructure (VPN tunnel) by multiple clients
- Much lower outbound network traffic by utilizing message fan-out.
- Improved performance by utilizing SC cache near to the clients.
- Routing of services into other network segments.
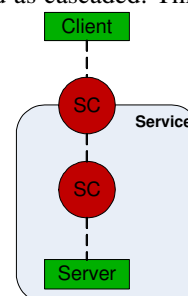
2.6 Message Caching

For session services SC implements caching of message sent from servers to clients. The cached message is available to all clients sending a request for it. Each service has its own cache and each cached message has an absolute expiration date and time. The cache is loaded by the first client requesting the messages. All subsequent clients will get the message from the cache, bypassing the server processing. Security is not affected because message exchange is possible only in a scope of a valid session.

The definition, which message should be cached, how it is identified and how long it is valid, is matter of agreement between the communicating parties.

SC caching can be enabled or disabled per SC instance. In cascaded configurations caching is supported on cascaded session service. This allows caching near to the message consumer.
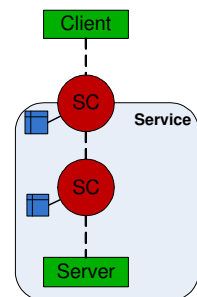
2.7 Message Fan-Out

For publishing services SC implements message fan-out in cascaded configurations. Message published by a server is distributed to the clients on the most upper cascade level. Between the SC nodes the message is transported only once. This allows efficient distribution of a message to large number of clients within a very short time.

The efficient usage of the fan-out is dependent on the design and structure of the subscription mask. It must contain items reflecting the cascade hierarchy.
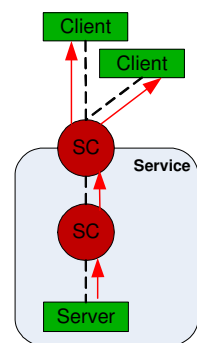
2.8 SC Console

SC console is a tool which allows sending of simple management commands to SC from the command line. It may be used in scripts embedded into other management tools.

2.9 Web GUI

Web-GUI is a simple monitoring instrument showing the current SC activity. It can be disabled in the configuration. It utilizes http and can be used from any remote node having access to SC.

2.10 Restrictions

2.10.1 Load balancing

The SC does not provide any load balancing features. Established communication session will not be redirected to another server or service node during its life time.

2.10.2 Failover

The SC does not provide any failover features. Aborted communication must be re-established by the communicating partners. The client application must find out a service which is ready.

2.10.3 Security

The SC does not implement any security feature. The environment where SC is used must provide all required authentication, authorization, encryption, tunneling etc. features. Message transport over https is not supported.

The IP address of the all nodes on the network path between the client and the server is available to the server when client starts a new session or the client subscribes. This can be used to authenticate and authorize the client when VPN tunnel is used.

2.10.4 Virus and Intrusion Protection

The entire network where SC is used is assumed to be safe and secure. No virus protection is embedded in SC. Screening firewall can be used on any segment of the network. It is recommended to use SC within a DMZ.

SC is not designed to withstand network attacks like DOS or SYNC flood, or any other.

3   Network

Clients and servers are not connected directly to each other. They are both connected to the SC. The Java API provided with SC and the communication between cascaded SCs uses a connection pool in order to save network resources.

Connection pool is established for each two communication end points. There is no fix allocation of a connection to a certain session or subscription within the pool. Each request may use a different connection. Connections are created on demand and closed when they are idle for longer time. Connection pool can be configured to use TCP/IP or HTTP protocol. Using HTTP adds an extra overhead to SCMP but may be simpler to configure in foreign networks. SCMP over HTTP will pass a screening firewall placed between the end-points.

Sending of keep alive messages can be configured for each connection pool. This is required whenever a firewall is configured on this network segment.
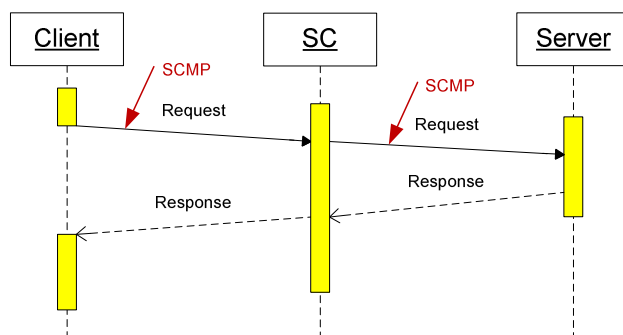

Figure 1 Example of a connection (simplified)

Cascaded SCs are connected hierarchically.
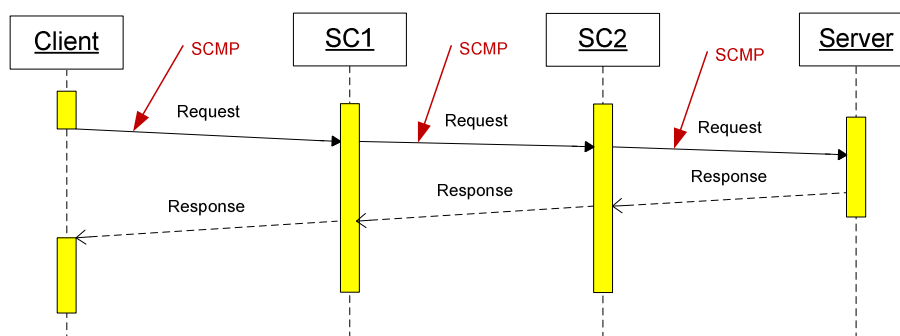
Figure 2 Example of a cascaded connection (simplified)

3.1 Connection Topology

Client application, server application and the SC may reside on the same node or on separate nodes. The connection can either utilize HTTP protocol or direct TCP/IP communication. No assumption about the physical network topology is made. Multiple firewalls can be located on the path between the communicating partners. In order to demonstrate SC features, following connection topologies are shown:

1.   Client ⇔ SC ⇔ Server   = Direct connection
2.   Client ⇔ SC ⇔ SC ⇔ Server   = Single cascade
3.   Client ⇔ SC ⇔ SC ⇔ SC ⇔ Server   = Double cascade

Other advanced cascaded topologies are possible.

3.1.1 Direct connection

This setup is suitable for applications with individual (mobile) clients connected directly to the service provider. In order to achieve higher application availability and scalability the server infrastructure can have redundant components.

The client can choose between connection to SC1 or 2 depending on their availability. Services X1 and X2 are identical and provided by servers 1 and 2. In case server 1 is not available service X2 can be used. Clients can use both X1 and X2 service and so distribute the load across all available servers. The logic to choose the access point and the appropriate service name is quite simple and must be implemented in the client.

The separation of SC nodes and the server nodes allows simple relocation of the services within the server site LAN and also allows scale-up the service performance by adding additional servers.
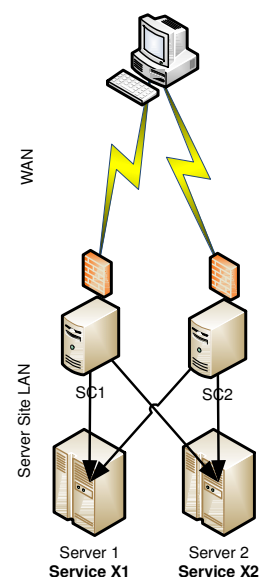
3.1.2 Single cascade

This setup is suitable for client applications running in enterprise environments. The clients are <u>not</u> connected directly to the service provider, but use an SC proxy placed within the enterprise DMZ as the local access point. In this way the enterprise security polices can be enforced. Utilization of the message cache in SC proxy can massively speed up the application. In order to achieve higher application availability SC proxy can be redundant. SC proxy can also be used to provide specific local services.

The client can choose between connection to SC proxy 1 or 2 depending on their availability. The logic to choose the access point and the appropriate service name is quite simple and must be implemented in the client.

Services X1 and X2 are identical and provided by server 1 and 2. In case server 1 is not available service X2 can be used. The network infrastructure must allow access between SC proxy 1, 2 and SC1 and 2.

The separation of SC nodes and the server nodes allows simple relocation of the services within the server site LAN and also allows scale-up the service performance by adding additional servers.
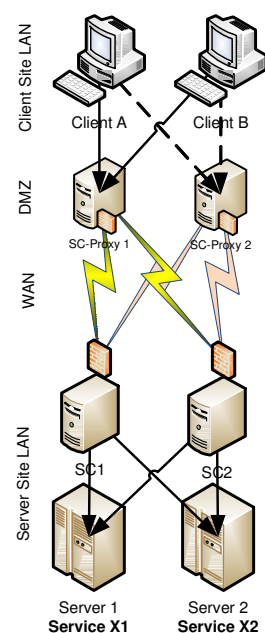
3.1.3 Double cascade

This setup is suitable for service providers wrapping the original service with some added value. The clients are <u>not</u> connected to the original service provider, but use an LAN SC proxy of the added value provider. Specific local services can be implemented here. The SC Proxy placed in the DMZ ensures the security polices and isolates the logical network segments. Utilization of message cache in the LAN SC proxy can massively speed up the application. In order to achieve higher application availability SC proxies can be redundant.

The client can choose between connection to LAN SC proxy 1 or 2 depending on their availability. The logic to choose the access point and the appropriate service name is quite simple and must be implemented in the client.

Services X1 and X2 are identical and provided by server 1 and 2. In case server 1 is not available service X2 can be used. The network infrastructure must allow access between SC proxy 1, 2 and SC1 and 2.

The separation of SC nodes and the server nodes allows simple relocation of the services within the server site LAN and also allows scale-up the service performance by adding additional servers.
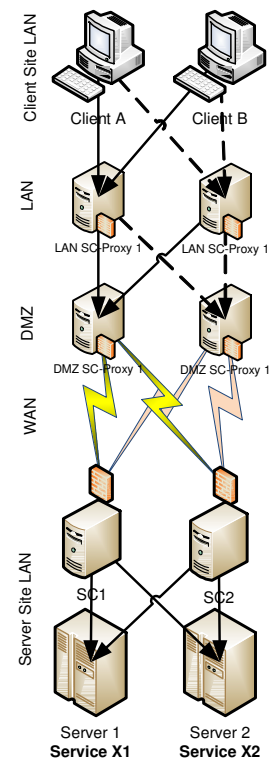
4   Installation, Update, Removal

SC is Open Source software under Apache License 2. It is available as a complete source code in the SVN repository http://www.stabilit.ch/svn/repos/sc and as a kit on http://www.stabilit.ch/download/sc/

Kit Structure   The kit is available in two flavors:
  - sc-bin_V9.9-999.zip   kit containing only binaries
  - sc-src_V9.9-999.zip   kit containing binaries and additional resources required for development

The structure of the kit is described in the *readme.txt* file included in both zip-files.

4.1 Installation

For pre-requisites and products required by the specific SC version please see the *readme.txt* file included in the kit. These are usually:
  - Java 1.6.18 or later
  - Apache 2.0 or later (on node running file services)
  - PHP 4.0 or later (on node running file services)
  - Apache Tomcat 6.0 or later (for services implemented as servlets)

The installation is done by extracting the kit file into a directory of your choice. After the installation a verification procedure described in *readme.txt* file included in both zip-files.

4.2 Update

If you made changes to the SC configuration, please preserve these files before upgrade. The upgrade is simply done by extracting the kit file into a directory of the previous installation.

Version compatibility   If you install new SC version you may see get error message of type ***"Incompatible SC version nr."*** This may happen when you clients or server has a higher version (is newer) then the SC (is older). The opposite situation (SC version is higher than the client/ server) is valid and the software will run without change.

For description of the compatibility issues please read SC_0_SCMP_E.PDF Chapter: SCMP header Attributes - scVersion.

4.3 Removal

You must stop SC before you want to remove the software.

If you use SC as Windows service then you must remove the service first. To do so, please read the corresponding readme file.

If you made changes to the SC configuration you must find out where you put the temporary files and delete them including the directories. These are:
  - log directory
  - root.pidPath
  - root.dumpPath
  - cache.diskPath

Then you delete the directory where did you extract the SC kit file.

5　　　　　　　　　　　　　　　　　Configuration

The SC is configured by property files in the conf/ directory. These are:
- **sc.properties**　　　　　　　　　　**sc configuration**
- **sc-specific.properties**　　　　　　**site specific sc configuration**
- **log4j-sc.properties**　　　　　　　**logging properties for sc**
- log4j-console.properties　　　　　　logging properties for sc-console
- log4j-demo-client.properties　　　　logging properties for demo-client
- log4j-demo-server.properties　　　　logging properties for demo-server

5.1 SC Properties

The SC is configured with a single configuration file sc.properties. Additional configurations may be included within this file.

The most important concepts used within the property file are described within the property file itself. They are:
- Multiple definitions of the same property but different value
- Include of specific configurations
- Usage of environment variables declared externally
- Usage of Java system properties within the property file

The meaning of the specific properties is explained in the configuration file itself.

The properties are grouped into 6 blocks:
1. Root = general parameters
2. Cache = caching parameters
3. Web = Web-GUI parameters
4. Listeners and their configuration
   Listeners have symbolic names and configuration parameters references these names.
5. Remote nodes and their configuration
   Remote nodes have symbolic names and configuration parameters references these names.
6. Services and their configuration

The block sequence is just a logical order. It is fully free.

```
#   Instructions for inclusion of other configurations.
#   ============================================================================
# include=propertyFile
# allows inclusion of other or customer specific configuration.
# If the same property is defined twice, the first found property defines the value!
# Subsequent property duplicate will NOT overwrite the value!
# Exception of this rule:
# For list properties (values separated by comma) the subsequent values
# will be appended!
# Properties which must not be overwritten must be placed before inlcude.
# Properties which may be overwritten must be placed after include.


#   Instructions for use of external parameters and java system properties.
#   ============================================================================
# External parameters can be passes on the command line like
#        -Dparameter=c:/temp
#  OS Environment variables can be passes on the command in the same way like
#        -Dparameter=%OS_VARIABLE%
#    Within this file the parameter value can be used as ${sys:parameter}
#    E.g. log path defined as OS environment variable LOG_DIR can be passed
#    on the command line as -Dlogdir=%LOG_DIR% and then used in this file as:
#            root.pidPath=${sys:logdir}
```

```
#
#       Java system properties can be used with the same syntax ${sys:property}
#       E.g. java system property file.separator can be used in this file as:
#               fs=${sys:file.separator}
#
#       For list of available java system properties see:
#       http://download.oracle.com/javase/tutorial/essential/environment/sysprop.html
# --------------------------------------------------------------------------------

# variables
# ---------
fs=${sys:file.separator}

# general parameters
# ------------------
# writePID (OPTIONAL, default=false) defines if a pid-file  should be written or not.
# pid-file is created at SC startup to signal a running SC. It contains the SC PID. The
# file is deleted on exit.
root.writePID=true


# pidPath (MANDATORY if writePID=true) defines directory where the pid file is written to.
# It can be relative or absolute path. Relative path begins with . or ..
root.pidPath=..${fs}logs${fs}sc


# dumpPath (MANDATORY) defines directory where the dump file is written to.
# It can be relative or absolute path. Relative path begins with . or ..
root.dumpPath=..${fs}logs${fs}sc


# commandValidationEnabled (OPTIONAL, default=true) defines if header attributes in
# incomming SCMP requests are validated or not. Disabling validation has only minimal
# impact on performance and is usefull for testing (it disables also the version check).
# ** ATTENTION ** TURNING OFF VALIDATION IS NOT RECOMMENDED IN PRODUCTIVE ENVIRONMENT
root.commandValidationEnabled=true


# operationTimeoutMultiplier (OPTIONAL, default=0.8) defines factor applied to oti
# in SCMP request. This will cause oti expiration on SC before oti expiration on the
# requester.
root.operationTimeoutMultiplier=0.8


# echoIntervalMultiplier (OPTIONAL, default=1.2) defines factor applied to eci
# in SCMP request. This will ensure session timeout expiration on SC before session
# timeout expiration on the requester
root.echoIntervalMultiplier=1.2


# checkRegistrationIntervalMultiplier (OPTIONAL, default=1.2) defines factor applied to cri
# in SCMP request of register server. This gives the allowed maximum time between
# subsequent checkRegistration requests from server until the server gets destroyed.
#root.checkRegistrationIntervalMultiplier=1.2

# connectionTimeoutMillis (OPTIONAL, default=1000) defines the maximum time SC will
# wait when it creates a new connection. SC creates connection to server, to another SC,
# to a web-server or to Tomcat.
#root.connectionTimeoutMillis=10000


# subscriptionTimeoutMillis (OPTIONAL, default=10000) defines the maximum time
# between subsequent client requests for new publications. If this timeout expires,
# client subscription is deleted.
# This parameter is analogous to echoInterval in session services.
#root.subscriptionTimeoutMillis=10000


# keepAliveOTIMillis (OPTIONAL, default=2000) defines the maximum time
# for receiving response to keepalive message. If this timeout expires,
# the connection is closed.
#root.keepAliveOTIMillis=2000


# serverAbortOTIMillis (OPTIONAL, default=60000) defines the maximum time
# for receiving response to abort session/subscription message sent to the server.
# Proxy uses value to send abort subscription to next SC.
```

Page 16   SC
SC_4_Operation_E   Service Connector
V1.5   Configuration

```
# If this timeout expires, server registration is deleted.
#root.serverAbortOTIMillis=60000

# cache parameters
# ----------------
# enabled (OPTIONAL, default=true) enables or disables caching in SC
cache.enabled=true


# diskPath (MANDATORY if cache.enabled = true) defines location on disk where
# cache will write temporary files holding the data.
# It can be relative or absolute path. Relative path begins with . or ..
cache.diskPath=..${fs}cache


# maxElementsInMemory (OPTIONAL, default=10000, 0 = no limit) defines maximum
# number of elements (messages or parts) in memory.
#cache.maxElementsInMemory=10000


# maxElementsOnDisk (OPTIONAL, default=1000000, 0 = no limit) defines maximum
# number of elements (messages or parts) on disk.
#cache.maxElementsOnDisk=1000000


# expirationCheckIntervalSeconds (OPTIONAL, default=30) defines interval in seconds
# the cache will check disk for expired messages and clean them up.
cache.expirationCheckIntervalSeconds=30


# web parameters
# --------------
# xslTransformationCache (OPTIONAL, default=false) enables or disables caching of
# xsl transformations done in web-gui. During development of the gui it is usefull
# to turn this on, otherwise browser must be restarted all the time.
#web.xslTransformationCache.enabled=false


# pageHeaderPrefix (OPTIONAL, default="") allows to define text displyed
# in page header and title. This is usefull to distinguish SC instances running
# at the same time
#web.pageHeaderPrefix=P01


# scDownloadService (OPTIONAL, default="") allows to define service used for
# downloading of files via SC mgmt GUI
web.scDownloadService=file-1

# scUploadService (OPTIONAL, default="") allows to define service used for
# uloading of files via SC mgmt GUI
web.scUploadService=file-1

# colorSchema (OPTIONAL, default="") defines color schema used for the mgmt GUI
# Is useful o distinguish multiple running SCs
# possible values = (blue, green, yellow, brown, red)
#web.colorSchema=blue

# include customer specific configuration
# ---------------------------------------
include = sc-specific.properties


# SC listeners
# ------------
# listeners (MANDATORY) defines symbolic names for listeners. Each listener must be
# configred later in this file. This is list property!
listeners=sc-http, sc-tcp, mgmt-http, http-proxy

# http listener for SCMP messaging
# port (MANDATORY) on which the listener listens
sc-http.port=7000

# interfaces (OPTIONAL) list of interfaces the listener should be included
# if missing, all interfaces will be included. This is list property!
#sc-http.interfaces=localhost

# connectionType (MANDATORY, allowed values = netty.http, netty.tcp, netty.web, netty-proxy.http)
```

```
# the connection type (protocoll) used by this listener
sc-http.connectionType=netty.http

# tcp listener for SCMP messaging
sc-tcp.port=9000
sc-tcp.connectionType=netty.tcp

# listener for SC Mgmt-Gui
mgmt-http.port=81
mgmt-http.connectionType=netty.web
# credentials for SC Mgmt-Gui
mgmt-http.username=admin
mgmt-http.password=admin

# listener for http proxy, traffic is redirected to one remote host
http-proxy.port=8080
http-proxy.connectionType=netty-proxy.http
http-proxy.remoteNode=apache


# remoteNodes (MANDATORY for cascading and file services or for http-proxy listener)
# defines symbolic names for remote nodes.
# Each host must be configred later in this file. This is list property!
remoteNodes=sc1,sc2,apache,fileServer

# remote SC connected via tcp (cascaded services)
# type (MANDATORY allowed values = cascadedSC, webServer, fileServer) defines
# the type of the remote host
sc1.type=cascadedSC

# port (MANDATORY) defines where the remote node listens
sc1.port=7001

# host (MANDATORY) defines the adress of the remote host
sc1.host=localhost

# connectionType (MANDATORY) defines the connection type (see above)
sc1.connectionType=netty.http

# maxConnectionPoolSize (OPTIONAL, default = 100) defines number of connections to this host
sc1.maxConnectionPoolSize=20

# keepAliveIntervalSeconds (OPTIONAL, default = 60) defines interval for sending keepalive messages
sc1.keepAliveIntervalSeconds=10


# remote SC connected via http (cascaded services)
sc2.type=cascadedSC
sc2.host=localhost
sc2.port=9001
sc2.connectionType=netty.tcp
sc2.maxConnectionPoolSize=20
sc2.keepAliveIntervalSeconds=10

# Web-Server for http-proxy traffic
apache.type=webServer
apache.host=www.stabilit.ch
apache.port=80
apache.connectionType=netty.http
apache.keepAliveIntervalSeconds=0
apache.maxConnectionPoolSize=20

# Web-Server for file services
fileServer.type=fileServer
fileServer.host=localhost
fileServer.port=80
fileServer.connectionType=netty.http
fileServer.keepAliveIntervalSeconds=0
fileServer.maxConnectionPoolSize=20
# maxSessions (OPTIONAL, default = 10) defines maximal number of internal sessions used for this file service
fileServer.maxSessions=10

# serviceNames (MANDATORY) defines names of services.
# Each service must be configred later in this file. This is list property!
serviceNames=session-1,session-2,publish-1,publish-2,file-1,sc1-session-1,sc2-session-2,sc1-publish-1,sc2-
publish-2
```

```
# local services
# type (MANDATORY, allowed values = session, publish, file)
session-1.type=session

# enabled (OPTIONAL, default=enabled) enables or disables session creation
session-1.enabled=true

session-2.type=session
session-2.enabled=true

publish-1.type=publish
publish-1.enabled=true

publish-2.type=publish
publish-2.enabled=true

# file services
file-1.type=file
file-1.enabled=true
# remoteNode (MANDATORY) defines the remote host on which a Web-Server must be running
file-1.remoteNode=fileServer
# path (MANDATORY) defines the url path used for this service.
# The path is added to the host address when the http request is constructed
file-1.path=sc/file-1/
# uploadScript (MANDATORY) name of the script residing on the web-server that handles file upload
file-1.uploadScript=scupload.php
# uploadScript (MANDATORY) name of the script residing on the web-server that handles file list
file-1.listScript=sclist.php

# cascaded services
sc1-session-1.type=session
sc1-session-1.enabled=true
# remoteNode (MANDATORY) defines the remote SC to which the service is cascaded
sc1-session-1.remoteNode=sc1

sc2-session-2.type=session
sc2-session-2.enabled=true
sc2-session-2.remoteNode=sc2

sc1-publish-1.type=publish
sc1-publish-1.enabled=true
sc1-publish-1.remoteNode=sc1
sc1-publish-1.noDataIntervalSeconds=10

sc2-publish-2.type=publish
sc2-publish-2.enabled=true
sc2-publish-2.remoteNode=sc2
sc2-publish-2.noDataIntervalSeconds=10
```

5.2 Logging

Standard open source package log4j is used for SC logging. It is configured with a single configuration file log4j-sc.properties.

For concepts of log4j please read http://logging.apache.org/log4j/
The complete log4j manual is available as http://www.qos.ch/shop/products/log4jManual

The most important configurations within the property file belong to:
- Usage of log levels
- Definition of appenders
- Usage of environment variables declared externally
- Usage of Java system properties within the property file

There are 5 loggers configured for logging:
1. **Default logger**
   This is a hierarchical logger used to produce regular log output to console and file. All errors will appear here. All levels are in use

2. **connectionLogger**
   This is specific logger used to log connection problems. It has only TRACE or DEBUG level

3. **messageLogger**
   This is specific logger used to log messages. It has only TRACE or DEBUG level.

4. **sessionLogger**
   This is specific logger used to log session events. It has INFO, DEBUG or TRACE level.

5. **subscriptionLogger**
   This is specific logger used to log subscription events. It has INFO, DEBUG or TRACE level.

6. **cacheLogger**
   This is specific logger used to cache events. It has only TRACE level.

7. **performanceLogger**
   This is specific logger used to log messages. It has only TRACE level.

```
#
# Usage of log levels inside the service connector:
# ============================================
# Log levels: ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF
#
#  FATAL -   fatal error, the execution cannot continue. The only possible action
#                    after this error is System.exit(1);
#                    The cause must be investigated and corrected before restart.
#  ERROR -   serious error which can be recovered. The execution may continue
#                    but error cause must be investigated and corrected.
#  WARN -    expected error that has been successfully recovered.
#                    Signal to take a corrective action in the future.
#                    This is the production set-up for maximum performance.
#  INFO -    Highly important event, documenting a significant activity.
#                    This is the production set-up for normal performance producing
#                    reasonable (not too high) amount of output.
#  DEBUG -   Importance event, used to document regular activity.
#                    Can be enabled for trouleshooting.
#  TRACE -   Event used to track down problem in a particular area.
#                    Must be selectively enabled for a particular class or branch
#
# Read log4j documentation at: http://logging.apache.org/log4j/1.2/manual.html
#
#
#  Instructions for use of external parameters and java system properties.
#  =========================================================================
#  External parameters can be passes on the command line like
#         -Dparameter=c:/temp
#  OS Environment variables can be passes on the command in the same way like
#         -Dparameter=%OS_VARIABLE%
#     Within this file the parameter value can be used as ${parameter}
#     E.g. log path defined as OS environment variable LOG_DIR can be passed
#     on the command line as -Dlogdir=%LOG_DIR% and then used in this file as:
#            root.pidPath=${logdir}
#
#     Java system properties can be used with the same syntax ${property}
#     E.g. java system property file.separator can be used in this file as:
#             fs=${file.separator}
#
#     For list of available java system properties see:
#     http://download.oracle.com/javase/tutorial/essential/environment/sysprop.html
# --------------------------------------------------------------------------

fs=${file.separator}
logPath=..${fs}logs${fs}sc
# It can be relative or absolute path. Relative path begins with . or ..

# All logging output sent to standard out and a file
# WARN is production set-up, INFO produces maximal acceptable output
log4j.rootLogger=INFO, console, generalLog

# console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSSZ} [%t] %-5p (%c:%L) - %m%n
```

Page 20   SC
SC_4_Operation_E   Service Connector
V1.5   Configuration

```
# generalLog
log4j.additivity.generalLog=false
log4j.appender.generalLog=org.serviceconnector.log.CustodianDailyRollingFileAppender
log4j.appender.generalLog.DatePattern='.'yyyy-MM-dd
log4j.appender.generalLog.File=${logPath}${fs}sc.log
log4j.appender.generalLog.layout=org.apache.log4j.PatternLayout
log4j.appender.generalLog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSSZ} [%t] %-5p (%c:%L) – %m%n
log4j.appender.generalLog.MaxNumberOfDays=7


# connections
# DEBUG level for connection creation and deletion
# TRACE level for read buffer, write buffer, send keepalive
log4j.logger.ConnectionLogger=OFF, connectionLog
log4j.additivity.ConnectionLogger=false
log4j.appender.connectionLog=org.serviceconnector.log.CustodianDailyRollingFileAppender
log4j.appender.connectionLog.DatePattern='.'yyyy-MM-dd
log4j.appender.connectionLog.File=${logPath}${fs}connection.log
log4j.appender.connectionLog.layout=org.apache.log4j.PatternLayout
log4j.appender.connectionLog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSSZ} [%t] %-5p – %m%n
log4j.appender.connectionLog.MaxNumberOfDays=7


# messages
# DEBUG important message attributes
# TRACE all message attributes
log4j.logger.MessageLogger=DEBUG, messageLog
log4j.additivity.MessageLogger=false
log4j.appender.messageLog=org.serviceconnector.log.CustodianDailyRollingFileAppender
log4j.appender.messageLog.DatePattern='.'yyyy-MM-dd
log4j.appender.messageLog.File=${logPath}${fs}message.log
log4j.appender.messageLog.layout=org.apache.log4j.PatternLayout
log4j.appender.messageLog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSSZ} %m%n
log4j.appender.messageLog.MaxNumberOfDays=7


# sessions
# INFO level for session timeout and abortion
# DEBUG level for session creation, deletion, abortion and timeout
# TRACE level for all session events
log4j.logger.SessionLogger=DEBUG, sessionLog
log4j.additivity.SessionLogger=false
log4j.appender.sessionLog=org.serviceconnector.log.CustodianDailyRollingFileAppender
log4j.appender.sessionLog.DatePattern='.'yyyy-MM-dd
log4j.appender.sessionLog.File=${logPath}${fs}session.log
log4j.appender.sessionLog.layout=org.apache.log4j.PatternLayout
log4j.appender.sessionLog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSSZ} [%t] %-5p – %m%n
log4j.appender.sessionLog.MaxNumberOfDays=7


# subscriptions
# INFO level for subscription timeout and abortion
# DEBUG level for subscribe, change subscription, unsubscribe, abortion and timeout
# TRACE level for all subscription events
log4j.logger.SubscriptionLogger=DEBUG, subscriptionLog
log4j.additivity.SubscriptionLogger=false
log4j.appender.subscriptionLog=org.serviceconnector.log.CustodianDailyRollingFileAppender
log4j.appender.subscriptionLog.DatePattern='.'yyyy-MM-dd
log4j.appender.subscriptionLog.File=${logPath}${fs}subscription.log
log4j.appender.subscriptionLog.layout=org.apache.log4j.PatternLayout
log4j.appender.subscriptionLog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSSZ} [%t] %-5p – %m%n
log4j.appender.subscriptionLog.MaxNumberOfDays=7


# performance
# TRACE level for begin and end points
log4j.logger.PerformanceLogger=OFF, performanceLog
log4j.additivity.PerformanceLogger=false
log4j.appender.performanceLog=org.serviceconnector.log.CustodianDailyRollingFileAppender
log4j.appender.performanceLog.DatePattern='.'yyyy-MM-dd
log4j.appender.performanceLog.File=${logPath}${fs}performance.log
log4j.appender.performanceLog.layout=org.apache.log4j.PatternLayout
log4j.appender.performanceLog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSSZ} [%t] %-5p – %m%n
log4j.appender.performanceLog.MaxNumberOfDays=7


# cache
# TRACE level for begin and end points
log4j.logger.CacheLogger=TRACE, cacheLog
log4j.additivity.CacheLogger=false
log4j.appender.cacheLog=org.serviceconnector.log.CustodianDailyRollingFileAppender
log4j.appender.cacheLog.DatePattern='.'yyyy-MM-dd
log4j.appender.cacheLog.File=${logPath}${fs}cache.log
```

```
log4j.appender.cacheLog.layout=org.apache.log4j.PatternLayout
log4j.appender.cacheLog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSSZ} [%t] %-5p - %m%n
log4j.appender.cacheLog.MaxNumberOfDays=7
```

6   Operations

This chapter describes the normal SC operations done by the operators.

Don't make changes to the scripts coming from the SC kit. Otherwise you will lose them after the next software update. It is recommended to wrap the scripts or to write your own scripts.

6.1 Starting SC

The Service connector is started with the script bin/start-sc.bat on Windows or start-sc.sh on unix or linux platforms.

It is possible to declare environment variables which are passed to SC where they can be used for configuration. The script shows an example how to do this. The property files in directory conf/ contain the description how to use the environment variables and java system properties inside the configuration files.

On Windows 32 and Windows 64 SC can be started as Windows service. The directories *bin/win32* and *bin/win64* contain the necessary tools and guide. The created Windows service is configured as manual start by default. If you want to start SC at boot time automatically, you can change the Windows service set-up with the standard Windows tools.

On unix or linux platforms SC can be started as a deamon. The directories *bin/unix* contains the necessary tools and guide.

6.2 Stopping SC

The Service connector is stopped with the script bin/stop-sc.bat on Windows or stop-sc.sh on unix or linux platforms. The script invokes the sc-console which in turn sends a *kill* message to the SC.

SC not configured to listen on TCP port cannot be stopped by the sc-console!

Even if sc-console can connect to remote SC, stopping SC is possible only from the local node where SC is running.

6.3 Other console operations

The sc-console can be used to manage services and to get information about the SC. SC not configured to listen on TCP port cannot communicate with the sc-console.
The syntax is:

```
java –jar sc-console.jar –h <host> –p <port>  <sc-command>
or alternatively
java –jar sc-console.jar -config <config-file>  <sc-command>
```

The possible sc-commands are:
- enable?serviceName=abc
  Enables services. Service must be enabled in order to accept new sessions or subscriptions.

- disable?serviceName=abc
  Disables services. Disables service will reject new sessions or subscriptions.

- `state?serviceName=abc`
  Shows state (enabled / disabled) of the services.

- `sessions?serviceName=abc`
  Shows number of allocated and free sessions for this service

- `inspectCache?serviceName=abc&cacheId=700`
  Shows the state of the message cache.

- `serviceConfiguration?serviceName=abc`
  Returns the configuration of given service name. **No** regex allowed here.

- `clearCache`
  Clears the message cache

- `scVersion`
  Returns the version of the SC the client is communicating to.

- `dump`
  Creates a dump file in the log directory containing information valuable for troubleshooting.

- `kill`
  Terminates the SC. This command can be issued only from node where the SC is running.
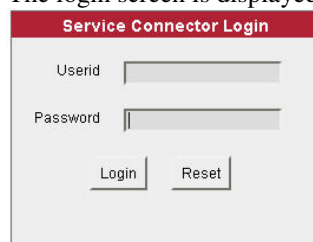
For serviceName value regular expression can be used (exception: serviceConfiguration). E.g
`state?serviceName=.*`
Will display state of all services.

6.4 Monitoring SC with the Web GUI

By default the mgmt-port is enabled and configured on port 81. Entering the URL http://localhost:81/ in your browser will connect with the SC.

Login

The login screen is displayed when a user connects to SC.

The credentials are defined in the configuration file.

Home

The home page shows the summary of the runtime environment and is refreshed periodically.

| SC 1.3-014 on canto | 2011-08-15 18:16:11 |
| | Service Connector provided by Stabilit |

| Home | | Logout |

Services
Sessions
Subscriptions
Servers
Listeners
Cache
Maintenance
Logs

| **Status** | ● |

Runtime	
Web Sessions	1
Processors	4
Free Memory	896984
Total Memory	16252928
Max Memory	259522560
Thread Count	51
Daemon Thrds	5
Peak Threads	53
Run GC	

Statistics	
Startup Time	2011-08-15 18:07:16
Runtime (min)	8' 59
Total Messages	750.0
Total Bytes	16734.0
Cached Messages	0

System Info	
configFileName	sc.properties
javaVersion	1.6.0_26
vmVersion	20.1-b02
localHostId	canto/10.0.0.129
os	Windows XP
osPatchLevel	Service Pack 3
cpuType	x86
userDir	C:\stabilit\projects\EUREX\SC\java\service-connector
countrySetting	US
userTimezone	Europe/Berlin
utcOffset	3600000
startupDateTime	2011-08-15T18:07:09.812+0200
availableProcessors	4
maxMemory	259522560
freeMemory	2125000
totalMemory	16252928
availableDiskSpace	1296729456640
useDST	true
threadCount	51
daemonThreadCount	5
peakThreadCount	53

The display has several sections:

- The header shows the actual version number and the node name on which SC is running. Actual menu is shown in the navigation line.
- There is a menu structure in the left column
- The status show if SC is running or not. When the GUI loses connection to the SC the green point will be red.
- The runtime box shows summary of the most important runtime parameters.
- The statistic shows an overall SC statistic.
- The System Info areas shows information related to the runtime environment. The link on the configuration file will display its content.

Services

The service form shows all configured services.

List of Services [9]						
Service State	Service Name	Service Type	Servers	Message Queue	Allocated Sessions / Subscriptions	Available Sessions
Enabled	file-1	FILE_SERVICE	-	-	-	-
Enabled	publish-1	PUBLISH_SERVICE	1	0	0	10
Enabled	publish-2	PUBLISH_SERVICE	0	0	0	0
Enabled	sc1-publish-1	CASCADED_PUBLISH_SERVICE	-	0	-	-
Enabled	sc1-session-1	CASCADED_SESSION_SERVICE	-	-	-	-
Enabled	sc2-publish-2	CASCADED_PUBLISH_SERVICE	-	0	-	-
Enabled	sc2-session-2	CASCADED_SESSION_SERVICE	-	-	-	-
Enabled	session-1	SESSION_SERVICE	1	-	0	100
Enabled	session-2	SESSION_SERVICE	0	-	0	0

The link on service state allows enabling or disabling the service interactively. When services are active, counters become links to a drill down and will display detailed information on sessions and servers.

Sessions

The session form shows all active sessions.

List of Sessions [10]							
Service	Session ID	IP Adresslist	Session Timeout (ms)	Server	Creation Time	Last Execute Time
session-1	f3d30c14-41e4-4fb1-a218-e32d6486a740	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:30	2011-08-15T18:20:00
session-1	6544cdc0-025f-43ba-9c11-872628fcf978	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:26	2011-08-15T18:20:00
session-1	527127a2-f315-41ed-8250-b852a130ccdb	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:28	2011-08-15T18:20:00
session-1	a604c7a2-72cc-47bc-a8df-4787e10c9c8c	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:26	2011-08-15T18:20:00
session-1	8f5434ad-507b-48aa-86fe-e65a43afb6f0	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:26	2011-08-15T18:20:00
session-1	bbfff743-c923-4185-9539-cf7ec81d760f	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:29	2011-08-15T18:20:00
session-1	8543d250-7fe9-44ea-876f-de94246817c9	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:28	2011-08-15T18:20:00
session-1	302c857c-9907-49b1-a482-cd966deb2418	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:27	2011-08-15T18:20:00
session-1	b42446e0-5c24-4a8a-a8e8-c4e9fe2fd8e7	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:26	2011-08-15T18:20:00
session-1	69878f40-cd50-4e23-98f2-b913d60e7ff4	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	12000.0	localhost:30000	2011-08-15T18:19:27	2011-08-15T18:20:00

The link on service will display service details, the link on server the server details

Subscriptions

The subscription form shows all active subscriptions.

List of Subscriptions [1]							
Service	Subscription ID	Subscription Mask	IP Addresslist	Subscription Timeout (ms)	No Data Interval (ms)	Server	Creation Time
publish-1	b79a4e41-c584-4409-bede-0419b9f8c4ea	0000121ABCDEFGHIJKLMNO-----------X-----------	10.0.0.129/127.0.0.1/10.0.0.129/127.0.0.1	60000.0	300000	localhost:30001	2011-08-15T18:17:56

c:\stabilit\projects\eurex\sc\documents\sc_4_operation_e.doc

The link on service will display service details, the link on server the server details

Servers The server form shows all active servers.

List of Servers [5] 1					
Host	Port	Server Key	Server Type	Max Sessions	Max Connections
localhost	9001	sc2	CASCADED_SC	-	20
localhost	7001	sc1	CASCADED_SC	-	20
localhost	80	fileServer	FILE_SERVER	10	20
localhost	9001	publish-1_localhost/23346	STATEFUL_SERVER	10	5
localhost	9002	session-1_localhost/23295	STATEFUL_SERVER	100	10

Listeners The listener form shows all active listeners.

List of Listeners [4] 1					
Host	Port	Name	Connection Type	Interfaces	Remote Node
127.0.0.1	7000	sc-http	netty.http	127.0.0.1 10.0.0.129 192.168.56.1	-
127.0.0.1	81	mgmt-http	netty.web	127.0.0.1 10.0.0.129 192.168.56.1	-
127.0.0.1	9000	sc-tcp	netty.tcp	127.0.0.1 10.0.0.129 192.168.56.1	-
127.0.0.1	8080	http-proxy	netty-proxy.http	127.0.0.1 10.0.0.129 192.168.56.1	apache

The link on remote node will display node details

Cache The cache form shows the state of the cache.

Cache Configuration				
Status	Disk Path	Max Messages in Memory	Max Messages on Disk	Loading (Sessions/MP)
ENABLED	cache\sc0	10000	1000000	0/0

List of Caches [2] 1			
Cache Name	Message Count	Messages in Memory	Messages on Disk
META_DATA_CACHE	1	1	1
DATA_CACHE	52	52	52

Maintenance The maintenance form allows performing maintenance actions.

SC Maintenance
SC Dump   Show SC Dump List   Delete All SC Dumps   Clear Cache
Reset Translet
Property File Download - file-1   Logfile Upload - file-1

Logs The log form shows the logs and allows their view.

Log files << 2011-08-15			
Logger Name	Appender Name	File	Log Level
CacheLogger	cacheLog	logs\sc0\cache.log (363241)	TRACE
ConnectionLogger	connectionLog	logs\sc0\connection.log (323567937)	TRACE
root	generalLog	logs\sc0\sc.log (4213243)	INFO
MessageLogger	messageLog	logs\sc0\message.log (201114405)	TRACE
PerformanceLogger	performanceLog	logs\sc0\performance.log (0)	OFF
SessionLogger	sessionLog	logs\sc0\session.log (188622544)	TRACE
SubscriptionLogger	subscriptionLog	logs\sc0\subscription.log (11491676)	TRACE

The link left of the date allows browsing previous days. The link on the log file will display its content.

6.5 Monitoring SC with jconsole or jvisualvm

Note: jconsole or jvisualvm can be used only when JDK 1.6.0 or later is installed. Remote monitoring is allowed, e.g. SC and jconsole may be on different nodes.

The script example/jconsole.bat and jvisualvm.bat show how to start jconsole or jvisualvm. When SC is running on your local node, you can directly select the SC from the process list. If SC is running on remote node you need to know the node, port of the node. The script bin/start-sc.sh shows how to define JMX remote port.

The most interesting parameters are:
- Heap Memory Usage
- Number of Threads
- CPU usage
- VM Summary

With jconsole or jvisualvm it is also possible to set the log level at run-time.

7   Troubleshooting

7.1 General

If you encounter a problem, please work systematically through the following checklist:

1) Connect to SC with the Web-GUI and check that it is responding. You can also use the console and get state of the services with *state?serviceName=.** command.
2) Look for ERRORs in sc.log

If you find suspicious log entries:
1) Use the Web-GUI and create a dump file
2) Upload log files and the dump file as a zip-file to the Web Server
3) Send a description of the issue and the zip-file to info@stabilit.ch

Note: **We understand the user frustration in a case of an error. However we need a precise description of the problem. In order to increase the chance to find and correct the error <u>soon,</u> please do:**
- **Keep cool, handle the incident <u>without</u> emotions.**
- **Work systematically. Do not jump from one topic to other.**
- **Communicate precisely, use SC terminology.**
- **Write down what you have done, later you will not remember the exact steps.**
- **Do not continue if not absolutely necessary. Other errors may subsequently happen and will blur the trace.**
- **Do not repeat the case. It will be more difficult to trace down the problem origin.**
- **Save all material, logs, notes, screen shots in one place.**

7.2 The most often situations

SC does not start.

Start SC manually from the command line and see the displayed error.
If there is lot of output before SC crashes see the error in the log file.

8   Glossary

Client

Piece of an application consuming services and initiating actions.

Client node

Client node is physical network node (computer) on which the client application resides.

Server

Piece of an application providing services to clients.

Server node

Server node is physical network node (computer) on which the server application resides.

Service

Abstract unit of work provided by the server and delivered to the client in order to implement a specific functionality. SC supports session, publishing and file services.

Session

Temporary allocation of a dedicated server to a client. Session ensures information flow between a client and the allocated server. SC supports request/response sessions and subscription sessions.

Call

A call represents a pair of a request and response.

Command

A command dispatches specific actions on a server according to the incoming request.

Request

Data structure created by the Client or SC in order to initiate an information exchange. It may contain one or more messages.

Response

Data structure created by the Server or SC in order to deliver the requested information. It may contain one or more messages.

Message

Basic transport instrument to exchange information between client, SC and the server. It belongs to a request or to a response.

Message Part

Message part is a piece of a large message. Large message is broken into parts.

Message payload

Message payload is part of the transported message containing the application data. SC does not inspect, transform or interpret the payload. It is the responsibility of the communicating partners to define the format, syntax and semantic of the payload.

Message transport format

Format of messages transported messages between client, SC and the server. It consists of a header and a payload.

Composite

Data structure prepared to hold all message parts of a large message

Registry

Common list of known objects that ensures their uniqueness, organized in a way to find them easily.

Requester

Piece of code in SC or client initiating the information exchange

Responder

Piece of code in SC or server delivering the requested information

Connection

Network communication between client and SC or SC and the server

Endpoint

Network communication part on SC or server

SCMP => SC Message Protocol

Is a publically presented protocol used for communication between client, SC and the server. In the OSI layer model, message transport protocol is immediately <u>above</u> the application protocol layer 7.

RR => Request / Response

This is basic communication style. The client requests an execution of a (remote) service and gets back the results in form of a response. The execution may be blocking - clients waits for the response, or non-blocking (asynchronous) – the clients does not wait for the response. Sometimes it is also called Remote Procedure Call = RPC.

SP => Subscribe / Publish

This is another popular communication style. The client tells the server what he is interested in and gets the data from the server later when a event arise. The execution is always non-blocking (asynchronous) – the client does not wait for the data receipt. This is because the event occurs on the server.

Index