STABILIT
Intelligente Informatik

SC_CC

Service Connector Cache Coherency

SC Cache Coherency Model

SC_CC-V1.2_E (Version V1.2)

This document describes the SC Cache Coherency Model.

This Document has been created with Microsoft Word 2003 (11) with template file C:\STABILIT\STANDARD\TEMPLATES\S_REP_E.DOT and printed at 31 August 2012 10:10.

Identification

Project:	SC_CC
Title:	Service Connector Cache Coherency
Subtitle:	SC Cache Coherency Model
Version:	V1.2
Reference:	SC_CC-V1.2_E
Classification:	Public
Keywords:	Architecture, Cache Concept, Coherency Model
Comment:	This document describes the SC Cache Coherency Model.
Author(s):	STABILIT Informatik AG Joël Traber
Approval (Reviewed by):	Signature ..... Jan Trnka
Audience:	Project team, Review team
Distribution:	Public
Filename	c:\stabilit\projects\eurex\sc\documents\sc_cc_e-v1.2.doc

Revision History

Date	Version	Author	Description
10.05.2012	D1.0	Joël Traber	Initial draft
12.06.2012	V1.0	Joël Traber	Change cache structure, appendix messages (SIX requirements). One cache per SC only
02.07.2012	V1.1	Jan Trnka	Preliminary document valid for the offer
16.08.2012	V1.1	Joël Traber	Description of Cache Structure. Add final version off SC Client API. Describe introduced header fields with SCMP V1.3.
31.08.2012	V1.2	Joël Traber	Value “static” for caching method not possible. Unset caching method or empty string makes data static (unmanaged).

Table of Contents

Tables

Figures

1 Preface

1.1 Purpose & Scope of this Document

This document describes the SC-CC **S**ervice **C**onnector **C**ache **C**oherency Model.

This version serves as basis for the implementation contract. The final version of this document will be integrated into Service Connector documentation.

This document is particularly important to all project team members and serves as communication medium between them.

1.2 Definitions & Abbreviations

Item / Term	Definition / Description
HTTP	Hypertext Transport Protocol
HTTPS	HTTP over SSL, encrypted and authenticated transport protocol

Table 1 Abbreviations & Definitions

1.3 External References

References	Item / Reference to other Document
[1]	SC_0_Specification_E – Requirement and Specifications for Service Connector
[2]	SC_0_SCMP_E – SC Message Protocol V1.2
[3]	SC_4_Operation_E – Configuration and Operation Guide
[4]	

Table 2 External references

1.4 Typographical Conventions

Convention	Meaning
text in italics	Features not implemented in the actual release or provisional text mentioned in open issues.
`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 exists.
…	In syntax diagrams, indicates a repetition of the previous expression

Table 3 Typographical conventions

The terminology used in this document may be somewhat different from 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:

- Look and feel of the Cache monitor is not finalized yet
- Subscription mask usage is provisional

2 Introduction

The SC supports message exchange between requesting application (client) and another application providing a service (server). Caching of messages can be activated on every SC.

Cache coherency refers to the consistency of data stored in local caches of every SC node. The coherency protocol described in following sections addresses the problem of maintaining the consistency of all caches in a system of distributed shared memory. Therefore a coherence model has been specified.

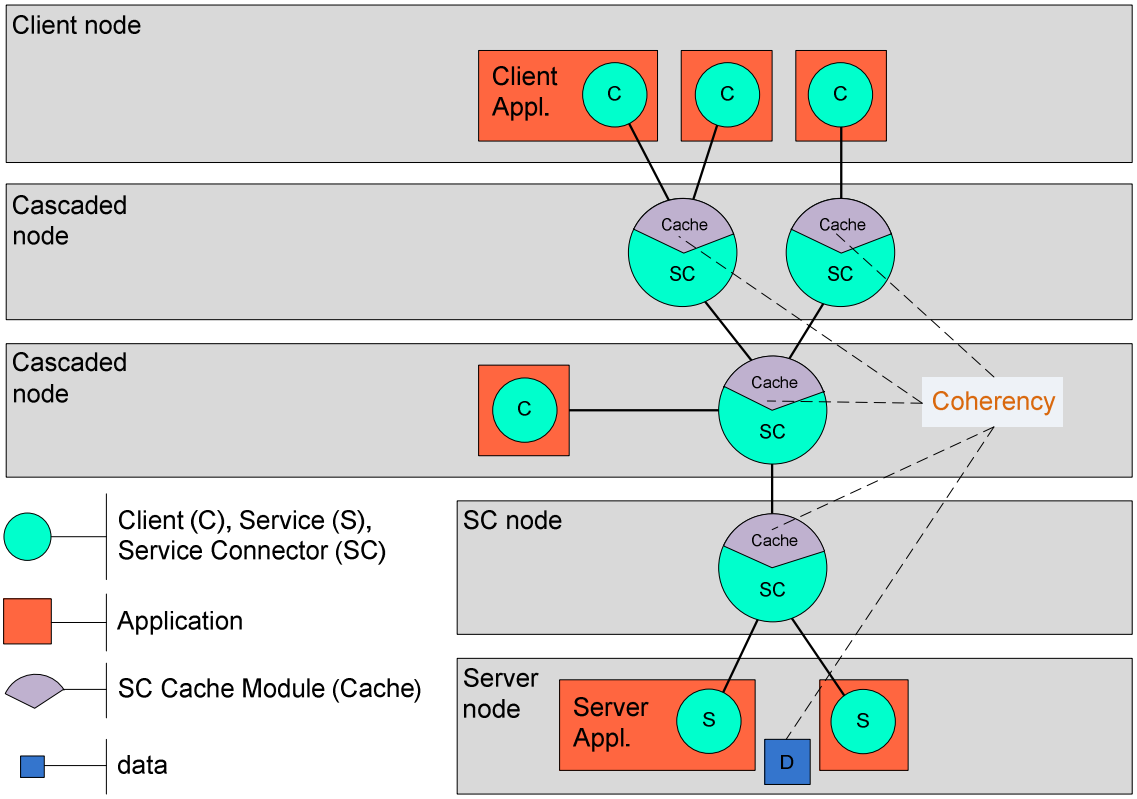

Figure 1 Topology & Caching in Service Connector

2.1 Cache coherence problem

The decision to cache or not cache a message is matter of an agreement between the client and the server. Keeping message in cache until it expires is the **most common lifecycle.** However lots of messages don't have a known lifecycle or caching is dependent on other client actions. Caching of such messages is only possible if there is a model to keep data consistent. This means that when the data is changed on server node, cached messages must immediately be updated in order to prevent clients to get obsolete data. Main target of the concept is to keep time between "data change" and "cache update" as small as possible.

It is impossible to avoid the coherence problem completely. For a short period of time the data in the cache may be older than the data on the server.

3 Cache Coherence Model

Cache-Guardians are introduced to ensure cache consistency. They work on the communication principles of a publish service. Any change of data on a server node, known as cached content, must be published to a Cache-Guardian. By using the fan out mechanism data update gets populated to all SC nodes and to the Clients. Updating the cache content is done inside the SC and works according to 3.1.3.

Error! Objects cannot be created from editing field codes.

Figure 2 Cache Coherence Model

3.1 Cache Coherence

3.1.1 Fundamental caching concept

As mentioned earlier, caching or not caching of data is an agreement of client and server. A client indication (cacheId) for caching of a message is needed. Afterwards the server confirms caching by returning the same cacheId. Indication of the client allows blocking other clients with the same request. If the server denies returning the cacheId, data is not cached.

The granularity of cached messages exactly correlates to the SCMP messages sent over the wire. Neither restructuring of messages nor modification of bodies will be done!

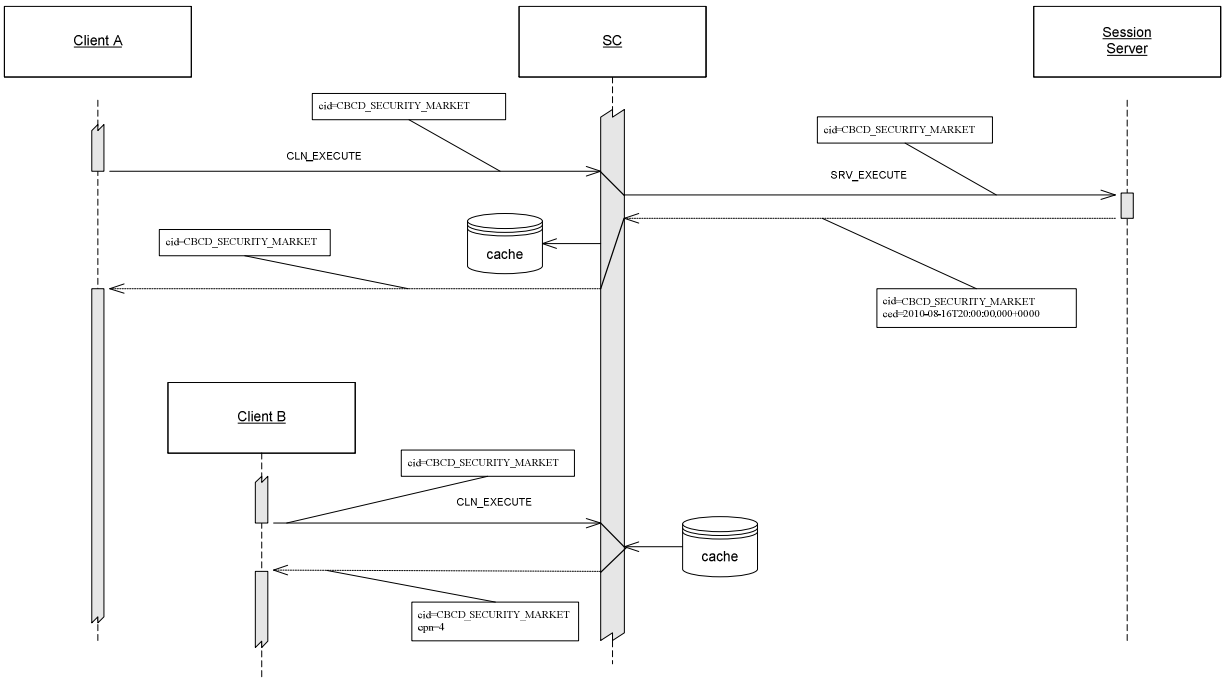

Figure 3 Caching Concept – Caching data

Caching is completely based on the cacheId. The cache may be structured by structuring the cacheId. The concept of used identifiers (cacheId) must be agreed between client and server.

The cache coherency model supports sending of Appendix, Removes and Initial Data (Replacements). After registering (normal service register procedure) a Publish Server to a

Cache Guardian a client may subscribe to receive data. Any data published to a Cache Guardian is populated to the clients and applied to the cache.

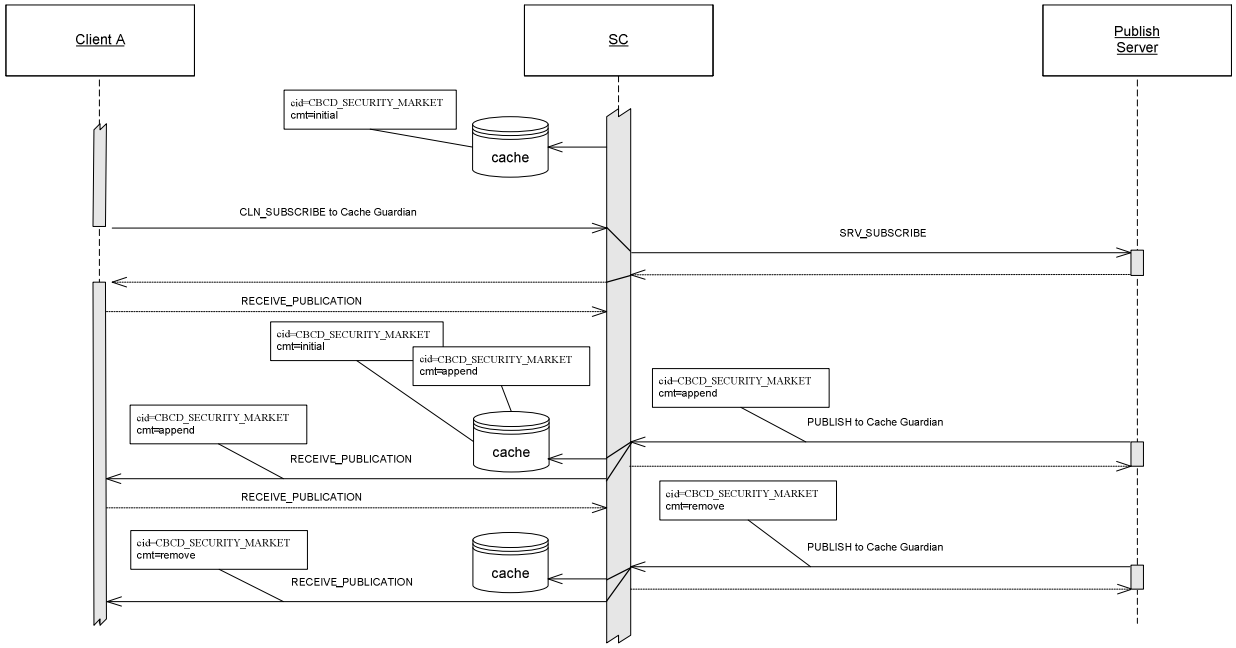

Figure 4 Caching Concept – Sending appendix and remove

3.1.2 Cache-Guardian

Cache Structure The cache is divided into static and managed data. Static data is kept in cache until its "expiration time" expires or a "remove" is received. Neither publishing appendices nor replacements are allowed for static data.
For managed data the SC is responsible for handling the cache coherency. Appendices or replacements may be applied. Managed data may also have an expiration time.

SC Client API The SC Client API subscribes to a Cache-Guardian. Any data published to this Cache-Guardian by the publish server will be populated up to the client. As long as an API user keeps the Cache-Guardian active, data updates will automatically be received over a callback. At the time the Cache-Guardian is inactive, data cached in SC is still consistent but client is not informed about updates anymore. **In order to use the coherence model correctly clients have to establish connections to a Cache-Guardian as a first step!**

SC Cache-Guardians are defined in the configuration file (sc.properties) of the connected SCs. The cache is managed by Cache-Guardians. On a cascaded SC more than one Cache-Guardian might be active. Messages get loaded by a session request of a client. The first Cache-Guardian applying an appendix to a cached message is responsible to keep data consistent. Different messages may be assigned to different Cache-Guardians.

A Cache-Guardian stops in following cases:

- No client is online (nobody is interested in updates)
- Connection is lost between client and SC (potential lack of updates)
- Connection is lost between SC and SC (potential lack of updates)
- Connection is lost between SC and Publish Server (potential lack of updates)

Stopping of a Cache-Guardian triggers a clean-up procedure in the cache module. Any message the inactive Cache-Guardian has treated will be removed to avoid cache inconsistency. Next client requesting a deleted message causes a new load process**. In a sophisticated topology the request will never end up on server level!**

Publish Server To send updates to a Cache-Guardian a publish server needs to complete the normal service register procedure. Sent messages are populated up to the clients and applied to the caches on the way.
The same update might be published from more than one publishing Server to different Cache-Guardians. Cached messages are only updated by one Cache-Guardian. Other updates are ignored.

As long as no large messages are published, it's possible to have more than one server sending updates to the same Cache- Guardian. Apparently sending the same update twice from each server is nonsense and invalidates the cached message!

3.1.3 Interacting with the cache

Final decision of making a message cacheable is taken by the session server. SCMP V1.3 introduces a new header attribute cachingMethod, which allows the server interacting with the cache. Above actions are performed by the Cache-Guardian depending on the values of cachingMethod.

cachingMethod = "" **or missing**, static data

- **Can only be set by session server in response message.**
- Data is cached as static data (no updates possible) until expire time.
- Remove possible by publish server.

cachingMethod = "initial"

- Session server declares data in cache as **managed** data (updates possible).
- Append, initial (replace existing) or remove possible by publish server.
- An initial message published replaces correlating existing initial message and possible appendices in cache.

cachingMethod = "append", appendix

- **Can only be set by publish server in publish message.**
- Message is appended to cached initial message.
- Appendix ignored if no existing message found in cache.
- Appendix will be populated to the client.
- An Appendix might be a large message.

cachingMethod = "remove"

- **Can only be set by publish server in publish message.**
- Removes correlating initial message and possible appendices in cache.
- Remove will be populated to the client.

3.1.4 SCMP Version 1.3

SCMP V1.3 introduces new header attributes in order to support the coherency model.

Caching method "cmt"
Caching method indicates the process the cache has to complete when message arrives. Header attribute has to be set by the session/publish servers. Following values are allowed:

- initial (marks data as managed data, causes replacement of existing data in cache)
- append (current message gets appended if base message is already in cache)
- remove (removes any existing data in cache)

Number of appendix "nra"
Header attributes used **between clients and proxies** to support loading appendix process. Initial message stored in cache carries the number of appendices which were applied since initial load.

Appendix number “anr”
Appendices are received in sequence. This header attribute indicates what position current message takes in. This supports the process of loading appendices between **clients and proxies**.

The publish message sent by the server has been extended by the header attribute cacheId (cid) in order to publish appendix and deletions to the cache.

3.1.5 Caching identifiers

The caching module needs to structure messages and their parts and applied appendices. For this purpose the cacheId is base identifier.
A given cache id identifies a message as a complete unit. Appendices are stored by adding an index to the base cache id. For part messages a second index is concatenated.

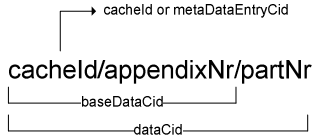

Figure 5 Structure of caching identifiers

The cache id with appendix number zero and part number zero (e.g. 700/0/0) is called initialDataCid.

Therefore using of the slash in cacheId by the user is forbidden! CacheId must be unique over all services!

3.1.6 Clear managed data in cache

Managed data in caches are bound to a publish Server. Shutting down (unregister) or aborting a publish server clears any managed data in the whole topology bound to this publish server.

4 Cache Coherence - SC Client API

The SC Client API supports usage of the cache coherence model by the interface described in following section.

4.1 SC Client API – Cache Coherence

Starting / Stopping Cache-Guardian
Method calls are synchronous and can be done after attaching SCClient successfully. Only one Cache-Guardian can be started for a SCClient instance.

```
startCacheGuardian(String cacheGuardianName, int
operationTimeoutSeconds, SCSubscribeMessage scSubscribeMessage,
SCGuardianMessageCallback scGuardianCallback, int
receivePublicationTimeoutSeconds) throws SCServiceException,
SCMPValidatorException
```

- `cacheGuardianName`: Identifies the Cache-Guardian.
- `operationTimeoutSeconds`: Time until starting of Cache-Guardian aborts. (Operation timed out)
- `scSubscribeMessage`: Subscribe message (see publish service for more details)
- `scGuardianCallback`: Callback to receive cache updates.
- `receivePublicationTimeoutSeconds`: Time to wait for completion of a receive publication request.

Client needs to have an active Cache-Guardian in order to receive updates over the callback. If no client is interested in updates (no Cache-Guardian active) no managed data will be cached.

Following method stops the cache subscription service in a shutdown scenario.

```
stopCacheGuardian(int operationTimeoutSeconds)
```

Receiving updates or removes
The callback to receive updates provides following methods:

```
// Inherited method gets called when initial data is received.
public abstract void receive(SCMessage reply);
// Inherited method gets called when an error shows up in communication process.
public abstract void receive(Exception ex);
// Method gets called when an appendix is received.
public abstract void receiveAppendix(SCAppendMessage appendix);
// Method gets called when a remove message is received.
public abstract void receiveRemove(SCRemovedMessage remove);
```

SCAppendMessage
Indicates that message is an appendix to an initial message.

SCRemovedMessage
Indicates that message has been removed from cache. The cacheId as a header attribute identifies deleted message. It does not make sense to deliver content in this message.

SCManagedMessage
Type of message for managed cache message read by the SC Client API. Beside the initial message in the body it contains an ordered list of Appendices as type of SCAppendMessage.

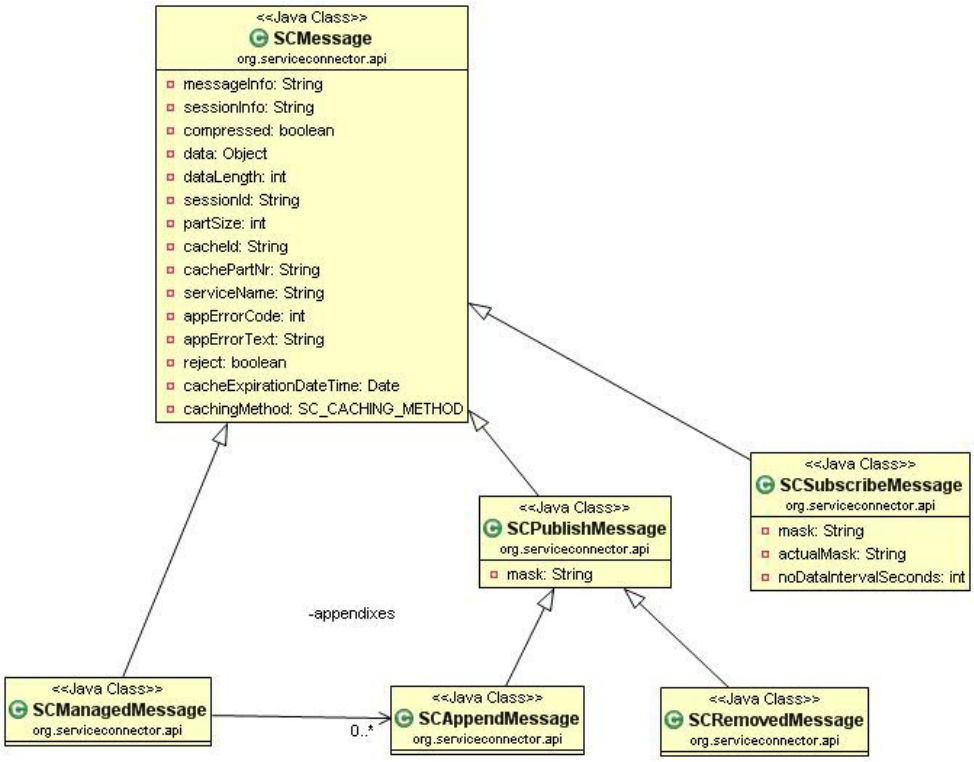

Figure 6 SCMessage - Class Diagram

4.2 SC Client API – Loading of large messages

During loading of a message, client can receive a CACHE_LOADING exception. This can happen in two cases:

a) The message is being loaded into cache by some other client.
b) The Cache-Guardian is appending a new large appendix to the message in cache.

Consequently the client may receive a CACHE_LOADING exception even it has received the first parts of a large message already. In such case the client must retry retrieving the message.

5 Monitoring and Troubleshooting

Basic concepts of SC monitoring and troubleshooting are described in SC_4_Operation_E.pdf. Subsequent section covers monitoring and troubleshooting for the context of cache coherence.

5.1 Cache Coherence Logging

The **cacheLogger** is responsible for logging the events in context of caching.

Following events will be logged:

- New subscription on Cache-Guardian.
- New managed message received.
- New Appendix message received.
- New Initial message received, existing data replaced.
- New Remove messaged received, existing data completely removed.
- Appendix error because no corresponding “cacheId” found.
- Broken Cache-Guardian -> Managed data removed.
- No more subscriptions on Cache-Guardian -> Managed data removed.

5.2 Cache Coherence Monitor

Below mentioned information in the context of cache coherence model are visible in the Web UI.

- Static data with expiration time.
- Managed data
  - Creation time of managed message (initial message received)
  - Available appendices, total number of
  - Number of parts for appendix
  - Cache-Guardian assigned to this message
- State of Cache-Guardian (active or inactive)
  - Active or inactive
  - Nr of subscriptions to the Cache-Guardian

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	

Figure 7 WebUI - Cache Overview

6 Configuration of Cache-Guardian

The SC is configured with a single configuration file sc.properties. More details are described in SC_4_Operation_E.pdf. Configuration of Cache-Update-Retriever is also done in sc.properties.

6.1 Cache-Guardian Configuration

Names of the Cache-Guardians have to be listed as shown below. A remote node defines which cascaded SC (only one) is publishing messages to a specific Cache-Guardian.

```
…...
serviceNames=sc1-cacheGuardian, sc2-cacheGuardian
…...
sc1-cacheGuardian.type=CacheGuardian
sc1-cacheGuardian.enabled=true
sc1-cacheGuardian.remoteNode=sc1
sc1-cacheGuardian.noDataIntervalSeconds=10

sc2-cacheGuardian.type=CacheGuardian
sc2-cacheGuardian.enabled=true
sc2-cacheGuardian.remoteNode=sc2
sc2-cacheGuardian.noDataIntervalSeconds=10
…...
```

7 Best Practice

7.1 Proper separation of static and managed data

Chapter 3.1.2 and 3.1.3 explain the handling of managed data when a lack of updates occurs. Basically the deletion of managed data by a Cache-Guardian it's a normal procedure to avoid cache coherency problem. For this reason the separation of static and managed data is very important. Data without the need to be updated for a longer time period should be declared static. Apparently the granularity of the messages has impact to the concept as well.

7.2 Use of the subscription mask

Like a publish service does a Cache-Guardian support the usage of the subscription mask. Published updates will be broadcasted according to the subscription mask. It's up to server and clients agreement to wisely use it.
If server and client agree an ordered list of all cacheIds and use the particular positions to identify the mask bit, client may precisely subscribe for the updates he needs. This also reduces network traffic between proxies.

8 Glossary

Term	Explanation
Appendix	Append message referring to an initial message.
Cache Coherency / Coherence	Cache coherence (also cache coherency) refers to the consistency of data stored in multiple caches of a shared resource
cacheId	Message identifier in the cache. Must be unique in the cache. Controlled by client and server.
Cache-Guardian	SC Module to guarantee data consistency in a cache. Responsible for the treatment of specific messages in a cache.
Data Consistency	Data consistency summarizes the validity, accuracy, usability and integrity of related data.
Initial message	Base message appendices may be applied to.
Managed data	Data in cache the SC takes care of consistency. Initial and append messages are managed data.
Static data	Data valid until expiration time. No appendices able to apply to.

Appendix

Index

No index entries found.