VSI OpenVMS

ServiceControl V3.5

Users Guide

March 2019

Revision/Update Information	Updated Manual
Software Version	VSI ServiceControl V3.5
Operating System Version	OpenVMS Alpha V7.3-2 & higher OpenVMS I64 V8.2 & higher

March 2019

Contents

Preface 12
Introduction 14
Supported OpenVMS versions 16
2.1 Advisory for running VSI OpenVMS ServiceControl on OpenVMS V7.3-2 16
2.2 OSC Management GUI advisory 17
Installation 18
3.1 Pre-installation Tasks 18
3.1.1 Inspect the distribution kit 18
3.1.2 Backing up the System Disk 18
3.1.3 Checking the Disk Space 19
3.1.4 Shutdown VSI OpenVMS ServiceControl 19
3.2 Installing VSI OpenVMS ServiceControl 19
3.3 Post-installation Tasks 20
3.3.1 Load the license key 20
3.3.2 Convert existing OSC configuration databases 20
3.3.3 Increase SYSTEM account quota settings 21
3.3.4 Start VSI OpenVMS ServiceControl 21
3.3.3.1 New Installation 21
3.3.3.2 Upgrade Installation 21
Basic Concepts and Terminology 22
4.1 What is an OSC Cluster? 22
4.1.1 Switchover and Failover 23
4.1.1.2 Switchover 23
4.1.1.3 Failover 24
4.2 Understanding OSC components 25
4.2.1 Resources 25
4.2.1.1 Actions on Resources 25
4.2.1.2 Resource Dependencies 26
4.2.1.3 Resource Categories 27
4.2.2 Services 27
4.2.2.1 Service Dependencies 28
4.2.3 Service Groups 29
4.2.3.1 Service Group Categories 30
4.2.4 Transactions 31
4.3 OSC software architecture 31
4.3.1 OSC Agents 32
4.3.1.1 The OSC Agent Framework 33
4.3.1.2 OSC Agent Operations 33
4.3.2 OSC service engine 35

4.3.3 OSC master control engine 36
4.3.4 OSC management utility 37
4.3.5 OSC configuration utility 37
4.3.6 OSC configuration database 38
4.4 OSC user privileges 39
4.4.1 OSC configuration 39
4.4.2 OSC management 39
4.4.2.1 OSC$MANAGE_CLUSTER 40
4.4.2.2 OSC$MANAGE_ALL 40
4.4.2.3 OSC$MANAGE_*servicegroup-name* 41
4.4.2.4 OSC$MANAGE_*service-name* 41
4.4.2.5 OSC$MANAGE_*resource-type* 41
Controlling OSC behavior 43
5.1 OSC behavior on resource faults 43
5.1.1 Managing resource faults 43
5.1.2 Cleaning Resources 44
5.1.3 Critical and Non-Critical Resources 44
5.1.4 Fault Propagation 44
5.1.5 Service Priority 45
5.1.6 Service Group Failover 45
5.1.7 OSC Behavior Diagrams 46
5.1.7.1 Scenario: Critical resource fails 46
5.1.7.2 Scenario: Non-Critical resource fails 47
5.1.7.3 Scenario: Critical resource fails to come online 47
5.1.7.4 Scenario: Non-Critical resource fails to come online 48
5.1.7.5 Scenario: Critical resource fails and fault propagation is disabled for the service group 48
5.1.7.6 Scenario: Critical resource fails and fault management is disabled for the service group 49
5.2 Controlling OSC Failover Policy 50
5.1.7.1 STATIC Failover Policy 50
5.1.7.2 LOAD-BALANCING Failover Policy 51
5.3 Controlling OSC Behavior at the Service Group Level 54
5.3.1 Controlling Service Group Execution Nodes 54
5.3.2 Controlling Service Group Concurrent Execution 55
5.3.3 Controlling Service Group Workload Value 55
5.3.4 Controlling automatic Service Group startup 55
5.3.5 Controlling Clean Behavior on Resource Faults 56
5.3.6 Controlling Fault Propagation 56
5.3.7 Controlling Failover on Service Group or System Faults 57
5.3.8 Controlling Behavior on external Resource startup 57
5.3.9 Controlling online Behavior of MultiInstance and Parallel Service Group containing cluster locked resources 58
5.3.10 Freezing Service Groups 58

5.3.11 Disabling Service Groups 59
5.4 Controlling OSC Behavior at the Service Level 62
5.4.1 ServicePriority Attribute 62
5.4.2 Freezing Services 63
5.4.3 Disabling Services 64
5.5 Controlling OSC Behavior at the Resource Level 66
5.5.1 Resource Type Attributes 66
5.5.1.1 ClusterLocked Attribute 66
5.5.1.2 CluLckResDisAllow Attribute 69
5.5.1.3 RestartLimit Attribute 70
5.5.1.4 ToleranceLimit Attribute 70
5.5.1.5 FaultOnMonitorTmo Attribute 70
5.5.1.6 FaultOnMonitorTmoLimit Attribute 72
5.5.1.7 OnlineRetryLimit Attribute 72
5.5.1.8 OnlineWaitLimit Attribute 72
5.5.1.9 OnlineTmoWaitLimit Attribute 73
5.5.1.10 OfflineWaitLimit Attribute 73
5.5.1.11 OfflineTmoWaitLimit Attribute 74
5.4.1.12 ConfInterval Attribute 74
5.5.1.13 CleanRetryLimit Attribute 74
5.5.1.14 TimeOutRetryLimit Attribute 75
5.5.1.15 DisableManageFault Attribute 76
5.5.1.16 ScriptExecUser Attribute 76
5.5.2 Freezing Resources 77
5.5.3 Disabling Resources 78
5.5.4 OSC agent monitor routine return codes 79
5.6 How OSC Handles Resource Faults 80
5.6.1 OSC Behavior when an Online Resource faults 80
5.6.1.1 OSC monitor flowchart for Online Resources 84
5.6.2 OSC Behavior when a Resource fails to come Online 85
5.6.2.1 OSC online decision flowchart 87
5.6.3 OSC Behavior after a Resource is declared faulted 88
5.7 Clearing OSC states 89
5.7.1 Clearing FAULT states 89
5.7.2 Clearing ADMIN_WAIT states 90
5.8 OSC adaptive startup behavior 90
OSC event notification 93
6.1 Configuring OSC events 96
6.2 OSC event message file 97
6.3 User event script 98
6.3.1 Example 99
Managing OSC 102
7.1 How to start OSC 102
7.2 How to manage an OSC cluster 103

7.2.1 Required privileges to manage the OSC cluster 103
7.2.2 Importance of OSC votes 103
7.2.3 OSC reconnection interval 105
7.2.4 OSC state transition 105
7.2.5 How to adjust OSC quorum 106
7.2.6 How to display OSC cluster status 106
7.2.7 How to switch OSC master control 107
7.2.8 How to lock an OSC node 107
7.2.9 How to unlock an OSC node 108
7.2.10 How to shutdown OSC 108
7.2.10.1 OSC cluster member shutdown 108
7.2.10.2 OSC cluster shutdown 109
7.2.11 How to manage OSC console connections 109
7.3 How to manage OSC service groups 110
7.3.1 Required privileges to manage the OSC Service Groups 111
7.3.2 How to display Service Group states 111
7.3.3 How to online a Service Group 111
7.3.4 How to offline a Service Group 112
7.3.5 How to switch a Service Group 113
7.3.6 How to freeze a Service Group 114
7.3.7 How to unfreeze a Service Group 114
7.3.8 How to disable a Service Group 115
7.3.9 How to enable a Service Group 116
7.3.10 Clearing Service Group FAULT state 116
7.3.11 Clearing Service Group ADMIN_WAIT state 117
7.4 How to manage OSC Services 118
7.4.1 Required privileges to manage the OSC Services 118
7.4.2 How to display Service states 118
7.4.3 How to online a Service 119
7.4.4 How to offline a Service 120
7.4.5 How to freeze a Service 122
7.4.6 How to unfreeze a Service 122
7.4.7 How to disable a Service 123
7.4.8 How to enable a Service 124
7.4.9 Clearing Service FAULT state 125
7.4.10 Clearing Service ADMIN_WAIT state 125
7.5 How to manage OSC resources 126
7.5.1 Required privileges to manage the OSC resources 126
7.5.2 How to display resource states 127
7.5.3 How to online a Resource 127
7.5.4 How to offline a Resource 128
7.5.5 How to freeze a Resource 129
7.5.6 How to unfreeze a Resource 131
7.5.7 How to disable a Resource 131

7.5.8 How to enable a Resource ... 132
7.5.9 Clearing Resource FAULT state ... 132
7.5.10 Clearing Resource ADMIN_WAIT state ... 133
7.6 How to manage OSC transactions ... 134
7.6.1 Required privileges to manage OSC transactions ... 134
7.6.2 How to display OSC transactions ... 134
7.6.3 How to cancel OSC transactions ... 135
7.7 OSC event console ... 136
7.8 SYSTEM Startup and Shutdown ... 137
7.9 Managing OSC using DCL scripts ... 137
7.9.1 Return codes ... 138
7.9.2 DCL managing script example ... 139
OSC configuration guidelines ... 141
8.1 Steps to create a valid OSC configuration ... 141
8.2 OSC configuration rules ... 141
8.3 OSC configuration planning ... 144
8.3.1 Service Group, Service and Resource configuration check list ... 144
8.3.2 OSC cluster configuration check list ... 149
8.4 From the configuration plan to the OSC configuration ... 152
8.5 Configuration example ... 157
8.5.1 OSC configuration planning ... 159
8.5.1.1 Service Group, Service and Resource planning ... 159
8.5.1.1 OSC cluster ... 169
8.5.2 OSC configuration ... 171
Developing new OSC agents ... 187
9.1 Overview ... 187
9.2 Design considerations ... 187
9.3 Action routines ... 188
9.3.1 Resource type specific attributes ... 188
9.3.2 Open ... 189
9.3.3 Monitor ... 189
9.3.4 Online ... 190
9.3.5 Offline ... 191
9.3.6 Clean ... 191
9.3.7 Passing parameters to the action routines ... 193
9.3.7.1 PassFuncCode common resource attribute ... 193
9.3.7.2 Resource name ... 194
9.3.7.3 ArgList common resource attribute ... 194
9.3.7.4 Reason codes ... 194
9.3.7.5 Argument summary ... 195
9.4 Using C or DCL scripts ... 198
9.4.1 Advantages using C ... 198
9.4.2 Advantage using DCL scripts ... 198
9.4.3 Using C and DCL scripts ... 198

9.5 Implementing action routines using DCL scripts 198
9.5.1 Example: The FileOnOff OSC Agent using DCL Scripts 199
9.5.1.1 DCL monitor script 199
9.5.1.2 DCL online script 200
9.5.1.3 DCL offline script 200
9.5.1.4 DCL clean script 201
9.6 Implementing action routines using C 202
9.6.1 C code syntax 202
9.6.2 tAttrItemList descriptor 203
9.6.3 Compile and Link 204
9.6.4 Example: The FileOnOff OSC agent using C 205
9.7 Adding an OSC agent to the configuration database 209
9.7.1 DCL script based OSC agent FileOnOff 211
9.7.2 Compiled OSC agent FileOnOff 214
9.8 Sending agent specific OSC event 217
9.8.2 DCL script OSC agent 217
9.8.3 Compiled OSC agent 218
Bundled OSC agents 219
10.1 OscAgtDSK - OSC disk agent 219
10.1.1 Description 219
10.1.2 Default agent attributes 219
10.1.3 Resource type specific attributes 220
10.1.4 Default common Resource attributes 220
10.1.5 How to define a DSK resource 221
10.2 OscAgtFSYS - OSC single disk volume agent 222
10.2.1 Description 222
10.2.2 Default agent attributes 223
10.2.3 Resource type specific attributes 223
10.2.4 Default common Resource attributes 224
10.2.5 How to define a FSYS resource 224
10.3 OscAgtSHD – OSC shadow set agent 226
10.3.1 Description 226
10.3.2 Default agent attributes 227
10.3.3 Resource type specific attributes 229
10.3.4 Default common Resource attributes 229
10.3.5 How to define a SHD resource 230
10.4 OscAgtPRC - OSC process agent 232
10.4.1 Description 232
10.4.2 Default agent attributes 232
10.4.3 Resource type specific attributes 233
10.4.4 Default common Resource attributes 233
10.4.5 How to define a PRC resource 234
10.5 OscAgtETH – OSC Ethernet adapter agent 237
10.5.1 Description 237

10.5.2 Default agent attributes ... 237
10.5.3 Resource type specific attributes ... 238
10.5.4 Default common Resource attributes ... 238
10.5.5 How to define a ETH resource ... 239
10.6 OscAgtIP – OSC service IP agent ... 240
10.6.1 Description ... 240
10.6.2 Default agent attributes ... 241
10.6.3 Resource type specific attributes ... 241
10.6.4 Default common Resource attributes ... 242
10.6.5 How to define a IP resource ... 243
10.7 OscAgtFailIP – OSC failSAFE IP agent ... 244
10.7.1 Description ... 244
10.7.2 Default agent attributes ... 245
10.7.3 Resource type specific attributes ... 245
10.7.4 Default common Resource attributes ... 246
10.7.5 How to define a FailIP resource ... 247
10.8 OscAgtORA – OSC Oracle 10 agent ... 248
10.8.1 Description ... 248
10.8.2 Default agent attributes ... 249
10.8.3 Resource category specific attributes ... 250
10.8.4 Default common Resource attributes ... 250
10.8.5 How to define an ORA resource ... 251
10.10 OscAgtRDB - OSC RDB agent ... 252
10.10.1 Description ... 252
10.10.2 Default agent attributes ... 252
10.10.3 Resource type specific attributes ... 253
10.10.4 Default common Resource attributes ... 254
10.10.5 How to define a RDB resource ... 255
10.9 OscAgtORALS – OSC Oracle 10 Listener agent ... 256
10.9.1 Description ... 256
10.9.2 Default agent attributes ... 257
10.9.3 Resource type specific attributes ... 257
10.9.4 Default common Resource attributes ... 258
10.9.5 How to define an ORALS resource ... 259
10.11 OscAgtPERF - OSC VSI PERFDAT agent ... 260
10.11.1 Description ... 260
10.11.2 Default agent attributes ... 260
10.11.3 Resource type specific attributes ... 261
10.11.4 Default common Resource attributes ... 261
10.11.5 How to define a PERF resource ... 262
10.12 OscAgtSQLSRV - OSC SQL service agent ... 264
10.12.1 Description ... 264
10.12.2 Default agent attributes ... 264
10.12.3 Resource type specific attributes ... 265

10.12.4 Default common Resource attributes 265
10.12.5 How to define a SQLSRV resource 266
10.13 OscAgtSQLDIS - OSC SQL dispatcher agent 267
10.13.1 Description 267
10.13.2 Default agent attributes 267
10.13.3 Resource type specific attributes 268
10.13.4 Default common Resource attributes 268
10.13.5 How to define a SQLDISP resource 269
10.14 OscAgtMYSQL - OSC MySQL agent 270
10.14.1 Description 270
10.14.2 Default agent attributes 270
10.14.3 Resource type specific attributes 271
10.14.4 Default common Resource attributes 271
10.14.5 How to define a MYSQL resource 272
10.15 OscAgtDECnet - OSC DECnet alias agent 274
10.15.1 Description 274
10.15.2 Default agent attributes 274
10.15.3 Resource category specific attributes 275
10.15.4 Default common Resource attributes 275
10.15.5 How to define a DECNET alias resource 276
10.16 OscAgtNCLObj - OSC DECnet object agent 278
10.16.1 Description 278
10.16.2 Default agent attributes 278
10.16.3 Resource category specific attributes 279
10.16.4 Default common Resource attributes 279
10.16.5 How to define a NCLOBJ resource 280
OSC simulation 281
11.1 OSC test open script 282
11.2 OSC test monitor script 283
11.3 OSC test online script 288
11.4 OSC test offline script 290
11.5 OSC test clean script 293
11.6 Starting OSC in simulation mode 294
Appendix 295
A.1 OSC cluster and system states 295
A.1.1 OSC Cluster membership states 295
A.1.2 OSC master control states 296
A.2 OSC service group states 298
A.2.1 OSC service group status keywords 298
A.2.2 OSC service group transaction 299
A.3 OSC service states 301
A.3.1 OSC service status keywords 301
A.3.2 OSC service transactions 302
A.4 OSC resource states 303

A.4.1 OSC resource status keywords 303
A.4.2 OSC resource transactions 304
A.5 OSC agent attributes 306
A.5.1 Mandatory OSC agent attributes 306
A.5.2 Optional OSC agent attributes 307
A.6 OSC master control and service engine attributes 309
A.6.1 Mandatory OSC master control attributes 309
A.6.2 Read-Only OSC master control attributes 312
A.6.3 Optional OSC service engine attributes 312
A.7 OSC resource attributes 315
A.7.1 Mandatory common OSC resource attributes 315
A.7.2 Optional common OSC resource attributes 316
A.8 OSC service attributes 324
A.8.1 Mandatory OSC service attributes 324
A.9 OSC service group attributes 325
A.9.1 Mandatory OSC service group attributes 325
A.9.2 Optional OSC service group attributes 326
A.10 OSC directory structure 330

Preface

This user's guide provides information on how to use and configure VSI OpenVMS ServiceControl (OSC) version 3.5 on an OpenVMS cluster.

Audience

The reader should be familiar with:

- OpenVMS cluster concepts
- How to configure and run an OpenVMS Cluster
- TCP/IP for OpenVMS
- VSI OpenVMS ServiceControl V3.5 – Release Notes

Document Structure

- Chapter 1 Introduction to VSI OpenVMS ServiceControl
- Chapter 2 Supported OpenVMS versions
- Chapter 3 Installing VSI OpenVMS ServiceControl
- Chapter 4 Basic Concepts and Terminology
- Chapter 5 Controlling VSI OpenVMS ServiceControl behavior
- Chapter 6 VSI OpenVMS ServiceControl event notification
- Chapter 7 VSI OpenVMS ServiceControl management
- Chapter 8 VSI OpenVMS ServiceControl configuration
- Chapter 9 How to develop new Agents
- Chapter 10 Agents bundled with VSI OpenVMS ServiceControl
- Appendix:
    - OSC cluster and system states
    - OSC service group states
    - OSC service states
    - OSC resource states
    - OSC agent attributes
    - OSC master control and service engine attributes
    - OSC resource attribute
    - OSC service attributes
    - OSC service group attributes
    - OSC event messages

Conventions Used in this Manual

Special	in examples indicates text that the system displays or user type input.
UPCASE	in a command represents text that you have to enter as shown.
Lowercase Italics	indicates variable information that a user supplies.
[ ]	in a command definition, enclose parts of the command that a user can omit.
Key	indicates a named key on the keyboard; for example, RETURN
CTRL/x	is the symbol used to represent the pressing of a control key. It indicates that the user holds down the key marked Ctrl and press the appropriate key.

1

Introduction

VSI OpenVMS ServiceControl (OSC) provides an availability management framework for any type of application running on an OpenVMS cluster. It can manage both - cluster aware and non-cluster aware applications. Using VSI OpenVMS ServiceControl it is possible to make non-cluster aware applications highly available to their clients with minimal effort.

Non-cluster aware applications do not synchronize access to shared resources within an OpenVMS cluster. Thus, such applications can only run on one cluster member at any time. Without using OSC, or another application management software with the same functionality, a non-cluster aware application has to be restarted manually on the same or another cluster member node if it fails due to a hardware or software fault.

VSI OpenVMS ServiceControl can monitor applications running on an OpenVMS cluster and all their required resources. If the cluster member on which an application is running fails, or if a particular required resource fails, VSI OpenVMS ServiceControl relocates or restarts the application depending on the type of failure and the failover policy applied. OSC guarantees that if an application has to failover to another cluster member the application will only be started on a node which has the required resources and where these required resources can be activated.

Any hardware or software entity can be defined a resource, such as a disk, file system, network card (NIC), IP address, a database and/or any kind of application.

All resources of a particular type are controlled by one OSC agent. Controlling a resource means monitoring the status of a resource, bringing it online (starting) and taking it offline (stopping). VSI OpenVMS ServiceControl provides several agents for controlling a wide range of resources (disks, shadow-sets, processes, NIC, failSAFE IP, Oracle DB ...).

Due to the design of VSI OpenVMS ServiceControl it is very simple to create a new OSC agent for a particular resource type, by providing monitor, start and stop command scripts or C functions that are compiled and linked against the OSC agent framework library.

VSI OpenVMS ServiceControl provides most of the features available with the VERITAS™ cluster server (VCS) for UNIX systems, plus some additional OpenVMS specific features. The semantics of the OSC management and configuration utilities are also similar to VCS. Hence, if you already know VCS the VSI OpenVMS ServiceControl training effort required is minimal.

2

Supported OpenVMS versions

VSI OpenVMS ServiceControl V3.5 is supported on:

- OpenVMS V7.3-2 AXP with HP TCP/IP Services V5.4 ECO5 and scalable kernel enabled
- OpenVMS V8.2 AXP
- OpenVMS V8.3 AXP
- OpenVMS V8.4 AXP
- OpenVMS V8.2 IA64
- OpenVMS V8.2-1 IA64
- OpenVMS V8.3 IA64
- OpenVMS V8.3-1H1 IA64
- OpenVMS V8.4 IA64
- VSI OpenVMS V8.4-1H1 IA64
- VSI OpenVMS V8.4-2 IA64
- VSI OpenVMS V8.4-2L1 IA64

2.1 Advisory for running VSI OpenVMS ServiceControl on OpenVMS V7.3-2

If one OpenVMS cluster member is running OpenVMS V7.3-2 with HP TCP/IP Services V5.4 please ensure that at least ECO5 of TCP/IP V5.4 is installed and that the HP TCP/IP Services scalable kernel is enabled.

Important

Do not install and run VSI OpenVMS ServiceControl on a node if either the traditional kernel of TCP/IP V5.4 is enabled or the ECO level is below ECO5.

Due to a bug in HP TCP/IP Services for OpenVMS V5.4, that has been fixed with ECO5 in the scalable kernel, but that has never been fixed in the traditional kernel, any attempt to establish a console connection via the OSC$MGR utility or the OSC management GUI to OSC master control process will fail.

To enable the OpenVMS HP TCP/IP Services V5.4 scalable kernel add the following lines to the SYS$MANAGER:SYSLOGICALS.COM command procedure.

```
$ ! ONLY the argument "PERF=ALL" is supported.
$ ! Other values may cause unpredictable results
$ ! to disable scalable kernel support, comment out next line and reboot.
$ DEFINE/SYSTEM/EXECUTIVE TCPIP$STARTUP_CPU_IMAGES "PERF=ALL"
```

If HP TCP/IP Services has already been started when this change is made, a reboot of the system is required for the parameter settings to take effect.

2.2 OSC Management GUI advisory

One of the key new features in this release is that now, the user has full control of all transactions currently in progress on the OSC cluster. The new SHOW TRANSACTION command provides an overview of all ongoing transactions and the CANCEL TRANSACTION can be utilized to cancel either a specific or all transactions in progress.

If a transaction is canceled the state of all resources currently executing an action routine (i.e. ONLINE) and their entire parent Services and Service Groups are marked as CANCELED and ADMIN_WAIT.

This new CANCELED state is unknown to the OSC Management GUI V2.1.1 or lower versions. Hence, it is strongly recommended that the OSC management GUI is upgraded to V2.2.0 or higher.

3

Installation

Installing or upgrading VSI OpenVMS ServiceControl takes just a few minutes. No reboot is required after the VSI OpenVMS ServiceControl installation.

3.1 Pre-installation Tasks

Step	Tasks to perform …
1	Inspect the distribution kit
2	Back up the system disk
3	Check the disk space parameters
4	Shutdown VSI OpenVMS ServiceControl cluster-wide

3.1.1 Inspect the distribution kit

Make sure you have a complete software distribution kit. It should contain the following files.

VSI OpenVMS ServiceControl Installation Kit:

- OSC034.A

VSI OpenVMS ServiceControl documentation:

- ServiceControl_Users_Guide_V34.pdf – this document

3.1.2 Backing up the System Disk

Before you install VSI OpenVMS ServiceControl, VSI recommends that you back up the system disk using the backup procedures established at your site.

For information about backing up a system disk, see the *VSI OpenVMS System Manager's Manual Essentials*.

3.1.3 Checking the Disk Space

Disk space required for installing VSI OpenVMS ServiceControl on a cluster common disk:

- Approx. 20,000 blocks

3.1.4 Shutdown VSI OpenVMS ServiceControl

If this is an upgrade, shutdown VSI OpenVMS ServiceControl cluster-wide without shutting down the managed resources, service and service groups using the OSC$MGR command:

```
OSC$MGR> SHUTDOWN/CLUSTER
```

To start the OSC$MGR utility, run the image OSC$BIN:OSC$MGR.EXE.

If you do not shutdown VSI OpenVMS ServiceControl in advance of installing/upgrading VSI OpenVMS ServiceControl the installation procedure will fail.

3.2 Installing VSI OpenVMS ServiceControl

Make sure that you are logged into the system as a privileged user. VSI recommends that you are logged into the SYSTEM account. Install VSI OpenVMS ServiceControl using the VMSINSTAL utility:

```
$ @SYS$UPDATE:VMSINSTAL OSC034 disk:[kit-directory]
```

The installation procedure prompts for the disk where VSI OpenVMS ServiceControl should be installed.

It is strongly recommended to install VSI OpenVMS on a cluster common disk. Otherwise VSI OpenVMS ServiceControl has to be installed on each specified OSC cluster member separately. In addition, if VSI OpenVMS ServiceControl is not installed on a cluster common disk, you will be responsible for distributing changes of the OSC configuration database within the OSC cluster by copying the OSC configuration database file into the OSC$CFG directory on each of OSC cluster members.

3.3 Post-installation Tasks

3.3.1 Load the license key

During the installation process of VSI OpenVMS ServiceControl a license valid for a 32 node OpenVMS cluster is automatically applied.

If you have an additional license key (i.e. you want to run VSI OpenVMS ServiceControl on an OpenVMS cluster that contains more than 32 nodes) load the license key using the OSC$MGR command:

OSC$MGR> LOAD LICENSE *license-key*

To start the OSC$MGR utility, run the image OSC$BIN:OSC$MGR.EXE.

3.3.2 Convert existing OSC configuration databases

If you have upgraded VSI OpenVMS ServiceControl to V3.5 all OSC configuration databases have to be converted to the new format required by VSI OpenVMS ServiceControl V3.5 before starting OSC again.

Important

OSC V3.5 will fail to start if the default OSC configuration database has not been converted to the new database format.

To convert all existing (default and working) OSC configuration databases start the OSC configuration utility:

```
$ RUN OSC$BIN:OSC$CFG
```

During the initialization phase of the OSC$CFG utility all OSC configuration databases are converted to the new format. No additional manual actions are required.

3.3.3 Increase SYSTEM account quota settings

VSI recommends the following minimum quota settings (or higher) for the SYSTEM account on all OSC cluster members:

Quota	Value
WSdef	8192
WSQuo	32768
WSExtent	65536
Pgflquo	1500000

3.3.4 Start VSI OpenVMS ServiceControl

3.3.3.1 New Installation

If you have installed VSI OpenVMS ServiceControl the first time on your cluster, do not start VSI OpenVMS ServiceControl. The startup will fail since no valid OSC configuration exists. Configure VSI OpenVMS ServiceControl first.

3.3.3.2 Upgrade Installation

If you have upgraded VSI OpenVMS ServiceControl and a valid configuration OSC database exists start VSI OpenVMS ServiceControl with either of the commands listed below:

```
$ RUN OSC$BIN:OSC$MGR
OSC$MGR> START/CLUSTER/[MODE=SIMULATION]
or
$ @SYS$STARTUP:OSC$STARTUP.COM
```

For detailed information about how to start VSI OpenVMS ServiceControl cluster-wide please refer to section 7.1 How to start OSC.

4

Basic Concepts and Terminology

VSI OpenVMS ServiceControl (OSC) provides a framework for application management and availability running on an OpenVMS cluster. OSC monitors the state of applications (services) and all their required resources and restarts applications (services) on a different system automatically whenever hardware and/or software fails.

This chapter describes the various components of OSC and how they interact with one another.

4.1 What is an OSC Cluster?

An OSC cluster may consist of all or a subset of the cluster members of an OpenVMS cluster. OSC is supported to run on any supported OpenVMS cluster size (2 – 96 nodes).

OSC is typically deployed to keep business critical applications online and available to users. Thus, OSC has to detect application and node failures amongst OSC cluster members.

OSC provides a mechanism to detect failure of an application by issuing specific commands, scripts or images that monitor the overall state of an application. OSC also determines the state of underlying resources supporting the application, such as shadow sets, disk and network interfaces.

Any OSC decision and subsequent actions OSC performs, rely on the exit codes of theses commands, scripts and images OSC uses to determine the actual state of resources.

OSC utilizes the SCS layer of an OpenVMS cluster for reliable node failure detection.

4.1.1 Switchover and Failover

Failover and switchover are the terms given to starting up or shutting down application services on a different member node in a OSC cluster. In both cases, an application and its network identity are started up on the selected node. Client systems access a service IP address that moves with the service. Thus, client systems are unaware of which server the application they are using is running on.

A service IP address is a network address defined in addition to the base addresses of the member systems in the OpenVMS cluster. For example, in a 2-node cluster, where each member node is running a database application, a service IP address can be associated with the database application. Clients who use this service's IP address to access the database application are unaware of which physical server actually hosts the database application.

4.1.1.2 Switchover

A switchover is an orderly shutdown of a service (application and its supporting resources) on one server and a controlled startup on another server.

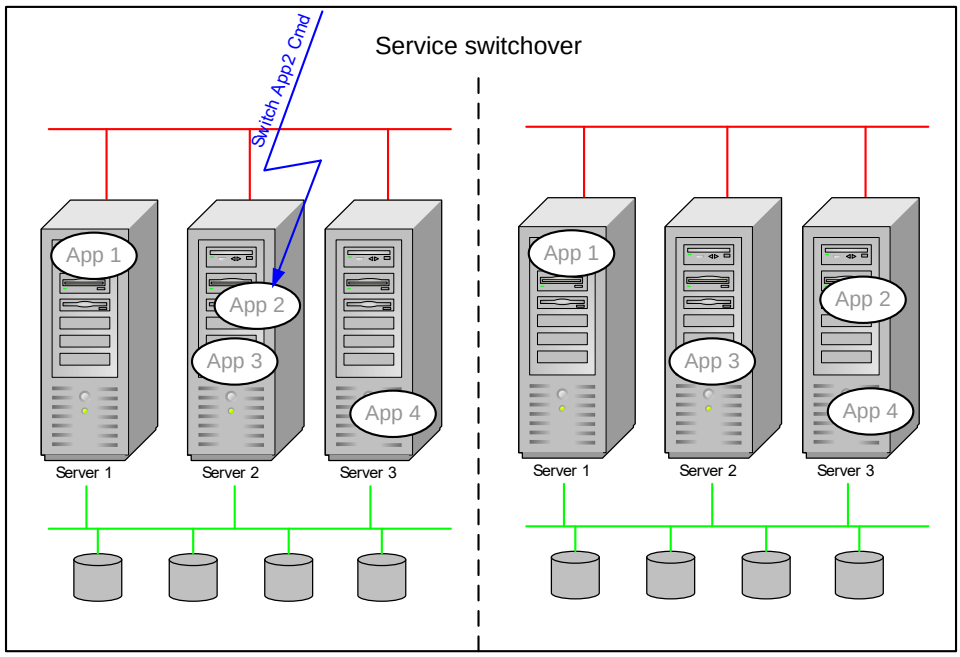


Fig. 4.1: Service switchover of application App 2.

Typically this means the de-assigning of the service IP address, the shutdown of the application and the release of allocated resources. On the other server the reverse happens. Disks and shadow sets are mounted, if necessary, the service IP address is assigned and the application is started.

4.1.1.3 Failover

A failover is a forced application switchover due to a service failure (= application failure or a failure of its supporting resource(s)) and/or a node failure.

If OSC detects an unrecoverable service failure it cleans up the faulted service component(s) (resources), initiates the orderly shutdown of all non-faulted components of a service and starts the service on another node. To clean up a service component (resource) means running a specific command, script or image that shuts down the faulted component –forcibly if required. After the cleanup job has completed successfully the faulted component is said to be offline.

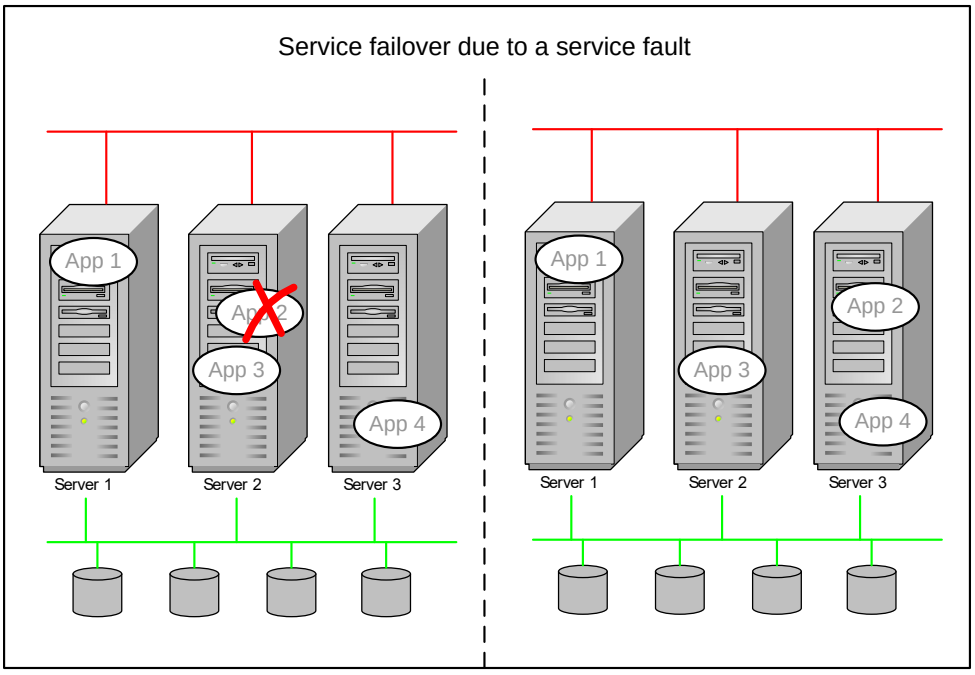


Fig. 4.2: Service failover due to an unrecoverable service failure.

If OSC detects a node failure (= cluster member has left the OpenVMS cluster) it starts all services that were running on the faulted node on other nodes within the OSC cluster.

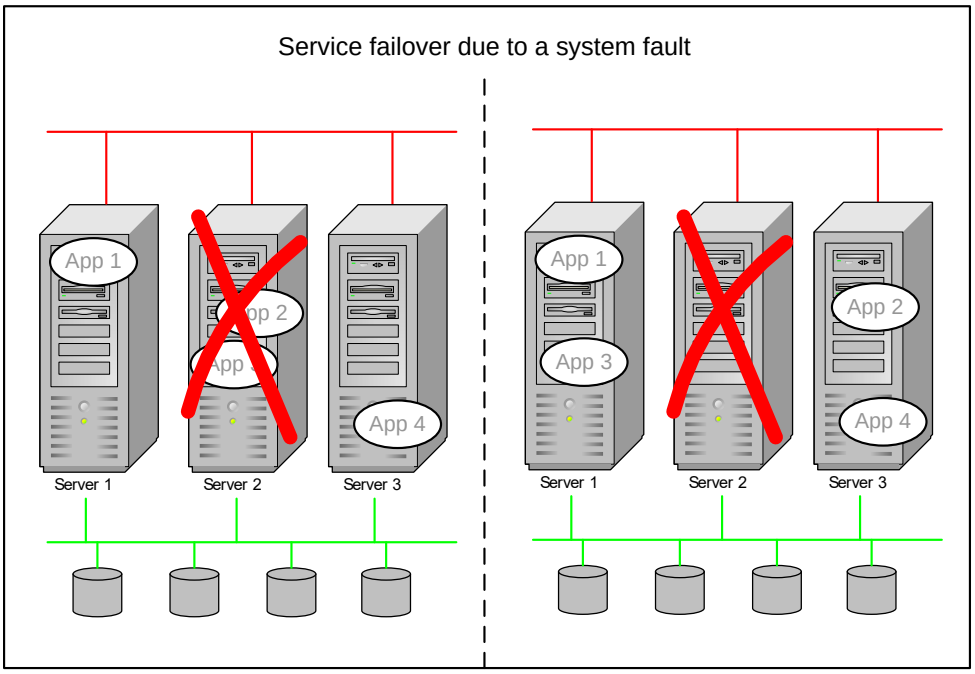


Fig. 4.3: Service failover due to a system failure.

4.2 Understanding OSC components

4.2.1 Resources

Resources are hardware or software entities, such as disks, shadow sets, network interface cards (NIC), IP addresses, and applications.

4.2.1.1 Actions on Resources

Controlling a resource means monitoring the resource, bringing it online (starting), taking it offline (stopping) and, in case of a fault condition, cleaning it (= forced shutdown).

4.2.1.2 Resource Dependencies

Resource dependencies determine the order in which resources are brought online or taken offline when their associated service (application) is brought online or is taken offline. For example, a disk device has to be online and mounted before applications can access files on that disk device. Conversely, all applications that access files on a disk device have to be stopped before the disk device can be dismounted.

In OSC terminology, resources are categorized as either parent or child resources. Child resources must be online before parent resources can be brought online, and conversely parent resources must be taken offline before child resources can be taken offline.

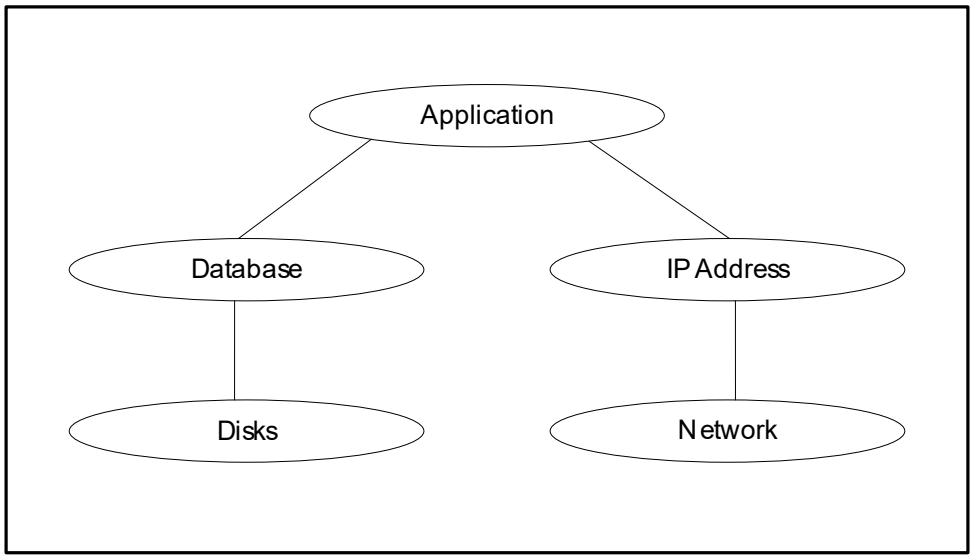


Fig. 4.4: Resource dependency example of an application that accesses a database and uses an IP-interface for inter-process communication.

Fig. 4.4 shows an example of a resource dependency tree. In this example the database and the IP address can be brought online concurrently because they have no interdependencies. After all child resources required by a parent are brought online, the parent is brought online and so on up the dependency tree, until finally the application program is started. Conversely, when deactivating a service, OSC starts at the top of the resource dependency tree. In this example, the application is stopped first, followed by the database and the IP address, and so on down the tree until all the child resources are stopped.

4.2.1.3 Resource Categories

Different types of resources require different levels of control. OSC manages three resource categories:

- **On-Off:** OSC starts and stops **On-Off** resources as required. For example, OSC assigns a dedicated service IP address to a network adapter when required, and removes the service IP-address when the associated service is stopped. All resources of a service that are not allowed to be online concurrently on different nodes within the OSC cluster have to be defined as **On-Off**.
    - **Cluster Locked:** **On-Off** resources can be defined as cluster locked resources. A cluster locked resource is a resource that will be started only one OSC cluster node regardless if the service group that owns this resource is configured to run concurrently on different nodes (**MultiInstance** or **Parallel** service groups - see section 4.2.3.1 Service Group). For detailed information about cluster locked resources please refer to the section 5.5.1 Resource Type Attributes.
- **On-Only:** OSC starts **On-Only** resources, but does not stop them. For example, an application requires access to files on a particular volume. The volume has to be mounted before the application can be started. Thus, OSC checks the volume status and mounts the volume if it has not already been mounted, but OSC does not dismount the volume if the associated service is taken offline. The prerequisite for defining a resource as **On-Only** is that such a resource is allowed to be online concurrently within the OSC cluster.
- **Persistent:** These resources cannot be brought online or taken offline. For example, a network adapter (NIC) cannot be started or stopped. OSC monitors **Persistent** resources to ensure their availability and operation.

4.2.2 Services

A service is a logical grouping of resources and resource dependencies that are required to run a dedicated service (application). It is the management unit that controls the resources.

For example (see Fig: 4.5), an Oracle database service may consist of the Oracle database and the shadow sets accessed by the database, the Oracle listener process and its service IP address.

OSC monitors all resources of a service. Whenever the status of a resource of the service changes the service status will be updated and the service status is propagated to the associated service group (see below).

Service operations initiate administrative operations for all resources within a service. For example, when a service is brought online, all resources that are managed by the service are brought online. All administrative operations on a service are initiated by the service group the service is a member of.

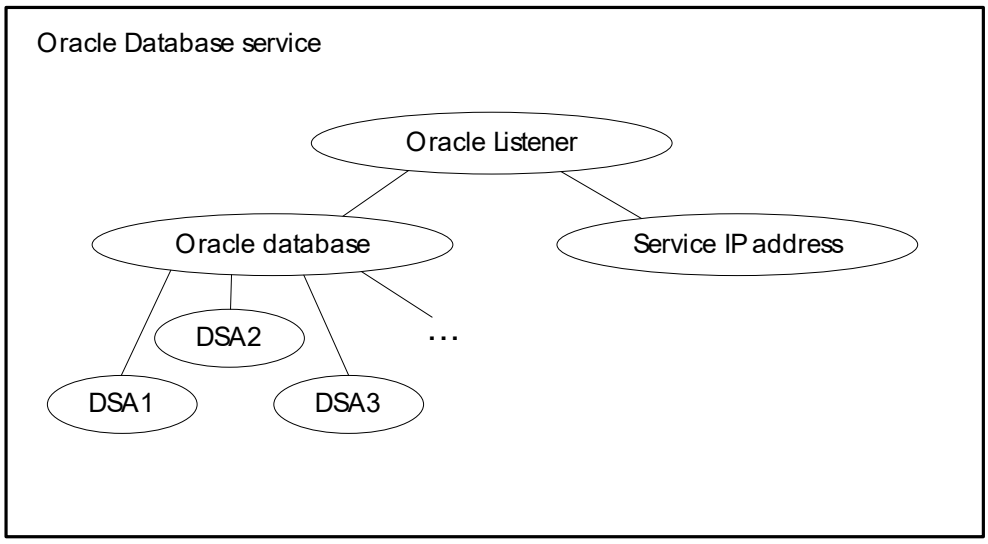


Fig. 4.5: Example of an Oracle database service.

4.2.2.1 Service Dependencies

As with resource dependencies, service dependencies determine the order in which services are brought online or are taken offline when their associated service group is brought online or taken offline. For example a database service has to be online before Web-services can access the database. Conversely, all Web-services that access a database have to be stopped before the database can be stopped.

In OSC terminology, services are categorized as either parent or child services. Child services must be online before parent services can be brought online, and parent services must be taken offline before child services can be taken offline.

4.2.3 Service Groups

A service group is a logical grouping of services and service dependencies. It is the management unit that controls service sets.

For example, a Web service group may consist of several Web applications that access the same database as shown in Fig. 4.6. Each of the Web-applications and the database can be configured as isolated OSC services.

A service group contains at least one service. A service group is the OSC management and failover entity.

A single OSC node may host any number of service groups. Each service group is monitored and managed independently. A service group can failover due to a resource fault or can be switched over to another node within the OSC cluster without necessarily affecting other service groups. If a node crashes, all service groups active on that node are failed over to other OSC members, as defined by the OSC configuration.

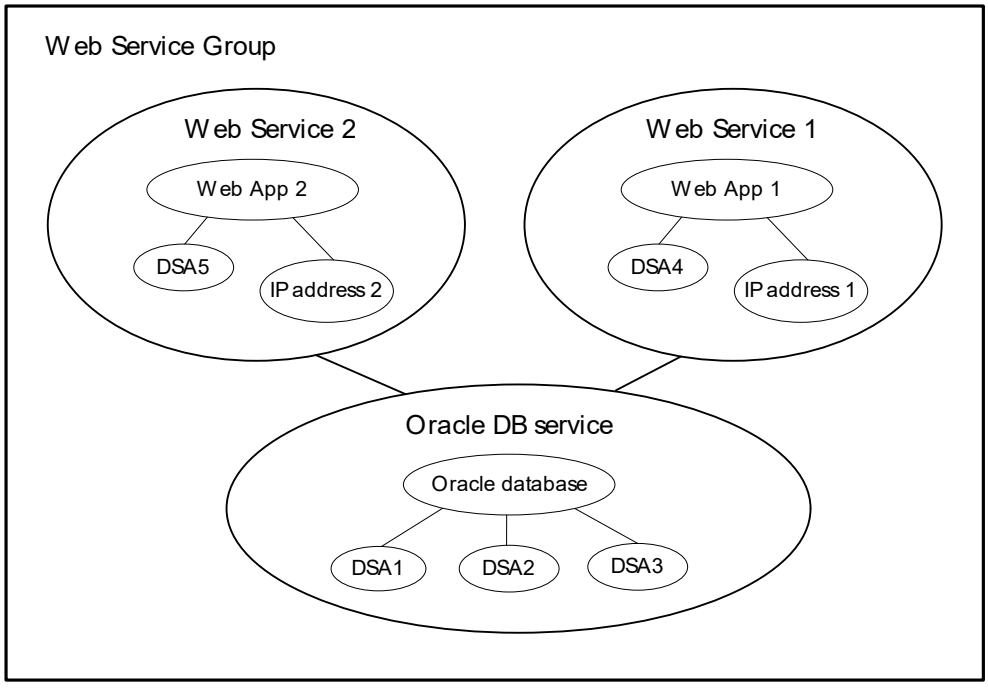


Fig. 4.6: Example of a Web service group.

If a failure of a critical resource is detected, the fault condition is propagated up to the service group level and the whole service group is restarted according to the OSC service group configuration. This could

mean restarting the service group locally or moving it to another OSC node and then restarting it. The chosen method is defined by the type of failure and the applied OSC failover policy and configuration parameters. In case of a local restart, the entire service group may have to be restarted. It could be that only a single resource within the service group has to be restarted to restore the service(s) of the service group.

Service group operations initiate OSC administrative operations for all services within the service group. For example, if a service group is brought online, all services within the service group are also brought online. When a resource fault is detected by OSC, resources and services never failover individually – the entire service group fails over.

4.2.3.1 Service Group Categories

Three OSC service group categories exist:

- **Failover** Service Group
  A **Failover** service group runs on only one system in the OSC cluster at any time. **Failover** service groups are used for non-OpenVMS cluster aware applications (=applications that are not designed to maintain data consistency when multiple copies are started).
- **MultiInstance** Service Group
  A **MultiInstance** service group can be active concurrently on more than one, but not on all systems within the OSC cluster. One can configure failover target nodes for a **MultiInstance** service group. In case of a service group failure on one node the service group instance will be restarted on one of the OSC failover nodes defined. Please note that only cluster aware applications (=applications that are designed to maintain data consistency when multiple copies are started) can be defined as **MultiInstance** service groups.
- **Parallel** Service Group
  A **Parallel** service group runs concurrently on all OSC cluster members. No failover target nodes can be configured for **Parallel** service groups. If a **Parallel** service group fails on one node the service group will only be restarted locally, if it is allowed to. Please note that only cluster aware applications (=applications that are designed to maintain data consistency when multiple copies are started) can be defined as **Parallel** service groups.

4.2.4 Transactions

A change of state of a single resource, service or service group from the current state to another state is done in the context of an OSC transaction. Thus, a single OSC transaction is always tied to a particular service group, service or resource and during such an OSC transaction all required actions are performed on the selected OSC entity and its child entities to switch the state of the selected OSC entity.

If an ONLINE transaction is started on a service group all of its child services and resources are started.

If an ONLINE transaction is started on a service all of its child resources are started, but not its parent service group.

If an ONLINE transaction is started on a resource only the resource is started, but not its parent services or parent service groups.

The same applies with an OFFLINE transaction.

Note

Since most of the OSC management commands triggering OSC transactions support wildcard parameter input lists, several OSC transactions can be started with a single management command (see section 7 Managing OSC for a detailed description of the OSC management commands available)

4.3 OSC software architecture

The software components of VSI OpenVMS ServiceControl are:

- OSC agents
- OSC service engine
- OSC master control engine
- OSC management utility – OSC$MGR
- OSC configuration utility – OSC$CFG
- OSC configuration database

Fig. 4.7 shows the overall software architecture of VSI OpenVMS ServiceControl.

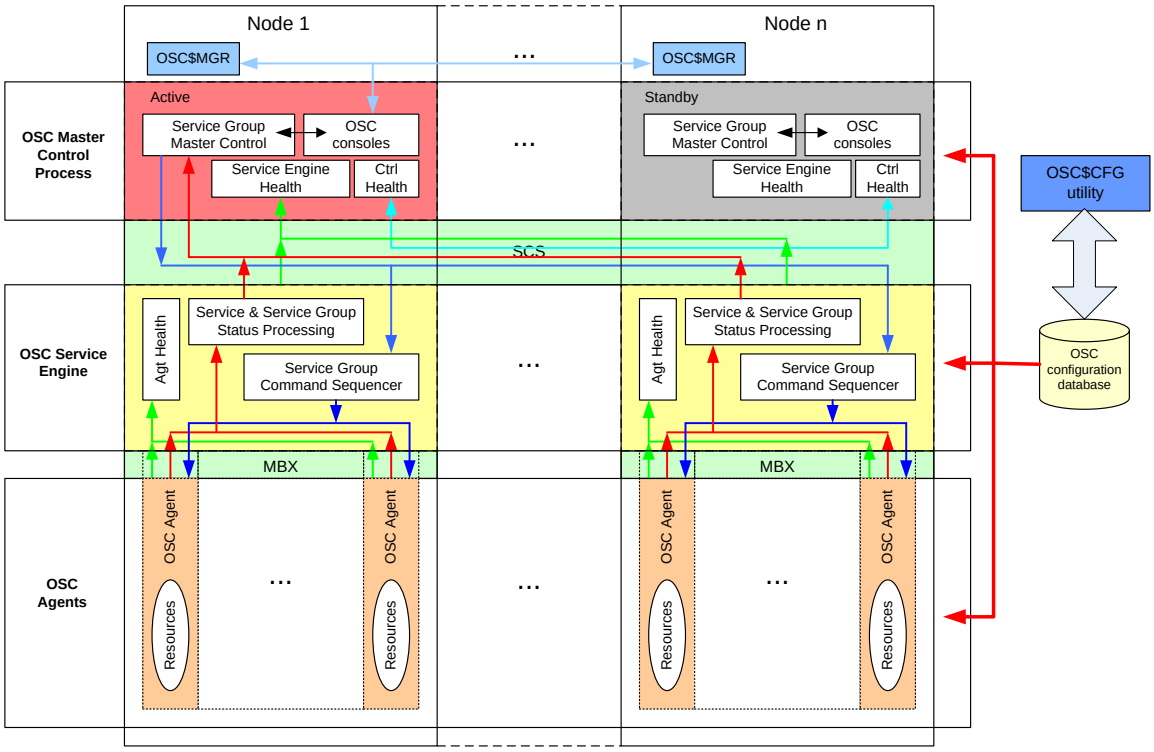


Fig. 4.7: VSI OpenVMS ServiceControl software architecture

4.3.1 OSC Agents

An OSC agent is an OSC process that monitors resources of a specific resource type and reports any status change of such resources to the OSC service engine. In addition, an OSC agent is responsible to bring a resource online, to take it offline or to clean it up. The online and offline actions are never triggered by the OSC agent itself, but only on request by the OSC service engine. One agent per resource type runs on each of the OSC cluster members and monitors all resources of that type. For example, a single OSC IP agent manages all service IP addresses.

When the agent is started, it obtains the required configuration information from the OSC configuration database. It then starts monitoring the resources periodically and reports resource status changes to the OSC service engine.

The OSC agent provides the type-specific logic to control resources. The action required to monitor and clean up a resource or to bring a resource online or take it offline differs significantly for each resource type. OSC employs different agents to handle this functional disparity between resource types. For example, bringing an IP-address online requires

configuring the IP-address on an active NIC, but bringing a database online requires starting the database manager process and issuing the appropriate startup commands.

OSC agents are multithreaded, meaning a single OSC agent monitors all resources of the same resource type on one host in parallel. For example, the IP agent monitors all IP resources. OSC monitors resources when they are online and offline to ensure they are not started on systems on which they are not supposed to run on. OSC starts an agent of a particular resource type on any OSC cluster where resources of that type may be potentially online.

If no OSC resources of a particular type are configured, the agent is not started. For example, if the OSC configuration database contains no Oracle resources for a system, the Oracle agent is not started on that system.

4.3.1.1 The OSC Agent Framework

OSC agents provide the capability to control a wide array of hardware and software resources. The agent abstraction makes it simple for a developer to create a new agent for controlling resources that are not handled by any of the OSC agents bundled with OSC, or to modify the behavior of existing OSC agents.

The OSC agent framework is a set of common, predefined functions compiled into each agent. These functions include the ability to connect to the OSC service engine and to understand common configuration attributes. The agent framework frees the developer from developing OSC workflow functionality. Instead the developer can focus only on the resource type specific action routines to monitor, online, offline and clean up (forced shutdown) resources of a particular type. For detailed information about developing agents using C or DCL command scripts, see section 9 Developing new OSC agents of this manual.

4.3.1.2 OSC Agent Operations

OSC agents monitor resources of a particular type, carry out specific operations on resources on behalf of the OSC service engine (online, offline a resource) and cleans up resources in response to resource faults. The OSC agent framework provides the processing logic. The resource type specific logic to monitor, online, offline and cleanup a resource is

provided by resource type specific action routines called by the OSC agent framework. These resource type specific action routines can be either compiled into the agent itself or can be implemented as DCL scripts (see section 9 Developing new OSC agents). The action routines called by the OSC agent framework are listed below.

- Open
  The open action routine is called to initialize a particular resource whenever the OSC agent starts managing the resource:
  - When the OSC agent starts
  - When a disabled resource is enabled again.

  When an OSC agent starts the open action routine for each managed resource it is guaranteed that this routine is called before its Monitor, Online, Offline or Clean action routine is called.
- Online
  The online action routine brings a specific resource online from the offline state. OSC agents never execute the online action routine for **Persistent** resources. **Persistent** rources are marked as online after the open action routine has completed.
- Offline
  The offline action routine shuts down online resources. The OSC agents never execute the offline action routine for **On-Only** and **Persistent** resources.
- Monitor
  The monitor action routine tests the status of a resource to determine if the resource is online or offline.
  The monitor action routine is periodically called to determine the resource state and to verify whether the resource state has changed. The online and offline monitoring intervals can be configured resource specific.
  In addition the monitor action routine is called for all managed resources whenever the OSC agent is re-started, and after every attempt to online or offline a resource in order to verify if that operation was successful.
  The monitor action routine is mandatory for all resource categories.
- Clean
  The clean action routine is called (forced shutdown) for a resource after a resource has failed to come online, failed to go offline, or failed while in an online state. The clean action routine has to be designed to forcibly shutdown a resource when it has failed, in order to ensure that the resource does not remain in an undefined state. The clean action routine will be executed only for **On-Off** resources, since these resources are typically not cluster aware.

On-Off resources have to be offline before they can be brought online on another OSC cluster member.
For example, if an OSC resource represents an application that consists of several processes, the clean entry point ensures that all processes of that application are stopped whenever one of these processes fail.

Note

OSC workflow decisions are based on the return codes of these action routines. It is a key requirement, that these action routines (code sections) provide reliable return codes regardless of the node state. Thus, do not add user defined OSC agents unless their actions routines have been extensively tested.

4.3.2 OSC service engine

The OSC service engine OSC$SRV is started on all OSC cluster members. The main tasks of the OSC service engine are:

- It monitors and controls the OSC agents on the local node.
    - It starts all required OSC agents on a node whenever the OSC environment is started on a particular node.
    - It stops all required OSC agent processes when the OSC environment is shutdown on a node.
    - It guarantees that all pending service group, service and resource transactions complete before the shutdown request is executed.
    - It monitors the health of the OSC agents running on the node. The OSC service engine periodically checks the receipt of heartbeat signals from the OSC agents. If this check fails (it received no heartbeat message from an agent within a predefined time interval) for an OSC agent, the OSC service engine automatically restarts the appropriate OSC agent if it is allowed to. The agent fault processing behavior of the OSC service engine can be defined agent specific. For detailed information about these agent specific control attributes please refer to the section A.5 OSC agent attributes.
- It maintains/updates the status of services and service groups locally configured according to the status information received from the OSC agents.

- It forwards the resource, service and service group status information to the OSC master control process.
- It guarantees that all resource, service and service group administrative commands received from the OSC master control process are executed according to the configured dependencies. For example, if the OSC service engine receives the online command for a service group from the OSC master control process, the OSC service engine ensures that all services of the service group and all resources defined within each of these services are brought online bottom-up according to the configured dependencies.

The OSC service engine communicates with the OSC agents via mailboxes. The process name of the OSC service engine is OSC$SRV.

4.3.3 OSC master control engine

The OSC master control engine OSC$CTRL is started on all OSC cluster members, but it is only active on one node. On all other OSC cluster members the OSC master control process is in standby mode waiting to be activated due to a shutdown of the active OSC$CTRL process or a system failure of the node that runs the active OSC master control process. VSI OpenVMS ServiceControl utilizes the OpenVMS distributed lock manager to activate an OSC master control process on a particular node.

The active OSC master control process periodically checks if OSC$CTRL processes exist on all OSC cluster members. If a standby OSC master control process is missing on an OSC cluster member, the OSC master control process is automatically restarted on that OSC node.

The active OSC master control process is the only OSC component that knows the status of all resource, service and service group on all OSC cluster members. Thus, the active OSC master control process is the one that decides whether to online, offline or failover a service group.

The OSC master control engine checks the health of the OSC service engines running on the OSC cluster members by checking the heartbeat signals sent by the OSC service engines. If the OSC master control process does not receive a heartbeat within a predefined time period from a particular OSC service engine it automatically tries to restart that OSC service engine according to the OSC service engine control parameters. For detailed information about OSC service engine control parameters

please refer to section A.6 OSC master control and service engine attributes.

The OSC master control process communicates reliably with all the OSC service engines running on each OSC cluster member via the OpenVMS SCS layer.

The active OSC master control process also provides the console interface for interactive OSC management. Up to 64 console links are supported by the OSC master control process.

4.3.4 OSC management utility

VSI OpenVMS ServiceControl can be managed with the OSC management utility OSC$MGR. To start the OSC management utility run the image OSC$CFG:OSC$MGR.EXE.

The OSC management utility automatically connects to one of the console interfaces of the active OSC master control process. Depending on the user privileges the user can:

- Display the status of the OSC cluster environment
- Display the status of all configured service groups, services and resources configured
- Manage dedicated service groups
- Manage all service groups, services and resources
- Manage the OSC cluster environment

For more detailed information about the user privileges required to manage the OSC environment please refer to the section 4.4 OSC user privileges.

For more detailed information about the management commands and the command syntax please refer to section 7 Managing OSC or to the OSC$MGR utility online help.

4.3.5 OSC configuration utility

The OSC$CFG utility is the VSI OpenVMS ServiceControl common configuration utility. To start the OSC configuration utility run the image OSC$BIN:OSC$CFG.EXE.

In order to modify an existing OSC configuration database or to create a new one, the user has to be logged in with the required privileges. For more detailed information regarding the user privileges required please refer to the section 4.4 OSC user privileges.

For detailed information about how to configure VSI OpenVMS ServiceControl please refer to the section 8 OSC configuration guidelines.

For detailed information about the available configuration commands and the command syntax please refer to the online help of the OSC$CFG utility.

4.3.6 OSC configuration database

The OSC configuration database contains:

- OSC cluster definition
- OSC master engine control parameters
- OSC service engine control parameters
- OSC agent control parameters
- All resource, service and service group definitions
- OSC event format definitions
- OSC event notification control parameters

Several OSC configuration databases can co-exist on an OpenVMS cluster. A user can manage all of these configuration databases with the DCL based configuration utility OSC$CFG. Before a user can start the OSC environment the user has to activate one of the OSC configuration databases as the default configuration database using the OSC$CFG utility. OSC configuration databases that are not active (=used and accessed by OSC control components – OSC agents, OSC service engine, OSC master control engine) are called *working* OSC configuration databases.

A working OSC configuration database has to contain a valid configuration that satisfies the OSC configuration rules (see 8 OSC configuration guidelines) before it can be activated. Once a particular OSC configuration database has been activated the content of the configuration database cannot be subsequently modified, even though the OSC environment has been shutdown. This restrictive behavior eliminates the risk that the OSC environment will not start up due to misconfiguration issues.

In order to modify the contents of the default OSC configuration database the user has to create a copy of the content of the default OSC configuration database using the COPY DATABASE command of the OSC$CFG utility. The copy of the default configuration is a new working configuration database. This new working OSC configuration database can now be modified and subsequently activated as the new default configuration database.

All OSC configuration databases (default and working) are located in the OSC$CFG directory. The file name of the default configuration database is always OSC$CONFIG.DAT.

If OSC is not installed on a cluster common disk, copy this configuration database file into the OSC$CFG directory on all OSC cluster nodes before the OSC environment is started. Otherwise OSC may not start on a particular OSC node due to a version mismatch fault.

4.4 OSC user privileges

4.4.1 OSC configuration

Only privileged users can modify existing or create new working OSC configuration databases. The required privileges are:

- CMKRNL
- OPER
- SYSLCK
- SYSPRV
- WORLD

Unprivileged users only have read access to existing OSC databases.

4.4.2 OSC management

The OSC environment is managed by the use of the OSC$MGR command line utility or the graphical user interface OscMgrGUI (Windows utility). Privileged users have full control of the OSC environment. A privileged OSC user requires the privileges listed below:

- CMKRNL
- OPER
- SYSLCK

- SYSPRV
- WORLD

Unprivileged users are allowed to view the status of the OSC cluster, all service groups, services and resources managed by OSC.
Unprivileged users can manage particular OSC items only if one of the OSC identifiers listed below has been assigned to the account the user is logged into:

- OSC$MANAGE_CLUSTER
- OSC$MANAGE_ALL
- OSC$MANAGE_*servicegroup-name*
- OSC$MANAGE_*service-name*
- OSC$MANAGE_*resource-type*
    - OSC$RES_ALL
    - OSC$RES_*resource-name*

These identifiers are not automatically applied when VSI OpenVMS ServiceControl is installed. These identifiers have to be added manually to the rights database and assigned to unprivileged accounts if required.

4.4.2.1 OSC$MANAGE_CLUSTER

This identifier grants the user the right to manage OSC cluster behavior. The user is permitted to perform the following operations:

- Switch the OSC master control process from one OSC cluster member to another one
- Shutdown OSC on all cluster members or on a particular OSC node
- Startup OSC on a particular OSC node
- Adjust OSC quorum. For detailed information about the OSC quorum scheme please refer to section 7.2 How to manage an OSC cluster.

The user is not permitted to manage (execute administrative operations) any service group, service or resource.

4.4.2.2 OSC$MANAGE_ALL

This identifier grants the user the right to manage all service groups, services and resources but the user is not permitted to manage OSC cluster behavior.

4.4.2.3 OSC$MANAGE_*servicegroup-name*

This identifier grants the user the right to manage a particular service group defined by the *servicegroup-name* extension assigned to the prefix OSC$MANAGE_. The user is not permitted to manage any other service group or to manage OSC cluster behavior.

4.4.2.4 OSC$MANAGE_*service-name*

This identifier grants the user the right to manage a particular service defined by the *service-name* extension assigned to the prefix OSC$MANAGE_. The user is not permitted to manage any other services, service group or to manage OSC cluster behavior.

4.4.2.5 OSC$MANAGE_*resource-type*

Unprivileged users can be enabled to manage either all resources of a particular type or to manage dedicated resources without being privileged to manage services, service groups or the OSC cluster.

Two identifiers are required to grant resource management privileges to unprivileged users:

- An unprivileged user is allowed to manage all resources of a particular type if the user has both identifiers
  - OSC$MANAGE_*resource-type*
  - OSC$RES_ALL

  assigned. The resource type is defined by the *resource-type* extension of the OSC$MANAGE_*resource-type*. The identifier OSC$RES_ALL signals that the user is allowed to manage all resources of that type.

  Example:

  An unprivileged user has the following identifiers assigned:
  - OSC$MANAGE_PRC
  - OSC$MANAGE_MYSQL
  - OSC$RES_ALL

  In this example the user is privileged to manage all process (PRC is the resource type of process resources – see section 10.4 OscAgtPRC - OSC process agent) and all MySQL databases (MYSQL is the resource type for MySQL

database resources – see section 10.14 OscAgtMYSQL - OSC MySQL agent).

- An unprivileged user is allowed to manage a dedicated resources only if the user has the identifiers:
  - OSC$MANAGE_*resource-type*
  - OSC$RES_*resource-name*

  assigned. The *resource-type* extension of the identifier OSC$MANAGE_*resource-type* defines the resource type of the resource the user is allowed to manage. The *resource-name* extension of the identifier OSC$RES_*resource-name* defines the name of the resource.

  Example:

  An unprivileged user has the following identifiers assigned:
  - OSC$MANAGE_PRC
  - OSC$RES_MY_PROCESS

  In this example the user is only privileged to manage the resource PRC::MY_PROCESS but not any other process resource.

5

Controlling OSC behavior

OSC provides an enhanced set of options to configure service groups, services and resources. These options allow greater flexibility and control when service groups failover in response to resource faults.

5.1 OSC behavior on resource faults

An OSC agent executes specific action routines to determine the actual state of a resource, to bring a resource online or to take it offline

A resource is considered faulted when:

- the resource state changes unexpectedly. For example, an online resource goes offline.
- a required state change does not occur. For example, a resource fails to go online or offline.

In many situations, OSC agents take predefined actions to correct the issue in question before reporting a resource fault condition to the OSC service engine. For example, the OSC agent may try to bring a resource online several times before reporting a fault.

This section provides an overview of the configuration options available to influence the OSC workflow behavior on resource faults and some examples that illustrate the effect of these basic configuration options.

5.1.1 Managing resource faults

OSC takes automatic actions on a resource fault only if the service group of the failed resource is configured to manage faults.

If fault management is disabled at the service group level and one of its resources fails, OSC takes no actions on the resource. It “hangs” the resource and the respective service(s) and service group(s) until system management has intervened. OSC marks the state of the failed resource, the affected service(s) and the affected service group(s) as ADMIN_WAIT and removes the failed resource from status monitoring until system

management has fixed the fault condition and the ADMIN_WAIT state has been manually cleared via the OSC$MGR utility.

If fault management is enabled, OSC tries to clean up the failed resource.

5.1.2 Cleaning Resources

When a resource fails and fault management is enabled, OSC calls a clean action routine to forcibly shutdown the resource. If the clean action routine fails to shutdown a failed resource the resource state is undefined and so service group failover processing will not be initiated. Failover of a service group that contains undefined resources may cause concurrency violations.

The clean routine is only called for **On-Off** resources since these resources are – according to the OSC terminology - not cluster aware. All other resource categories (**On-Only**, **Persistent**) are cluster aware resources that can be online concurrently on multiple nodes. Thus, a service group failover due to a fault of an **On-Only** or **Persistent** resource can never cause concurrency violations.

5.1.3 Critical and Non-Critical Resources

The **Critical** attribute of a resource defines whether or not a resource fault initiates fault propagation at the service group level. The attribute determines whether OSC tries to online a service group on another OSC member node if a resource fails to come online.
If a resource is configured as non-critical (by setting the **Critical** resource attribute to FALSE), services and service groups that contain this resource will not be marked as faulted. Failover processing will be initiated only if the service group is declared as FAULTED. Thus, faults on non-critical resources will not cause service group failover. OSC takes a failed, non-critical resource offline and updates the service and service group status to ONLINE|PARTIAL.

5.1.4 Fault Propagation

Fault propagation is the process of marking a service group as FAULTED when a resource fails.

When a resource fails, OSC propagates the fault condition up to the service and service group level if fault propagation is enabled. If fault propagation is disabled the services and service groups of the resource will not be marked as FAULTED and thus, the service groups will not failover.

5.1.5 Service Priority

Once a service is marked FAULTED, OSC checks the service priority of all independent service trees configured within the service group. If the service priority of an online service tree is greater than the value of the failed service tree, no service group failover is initiated.

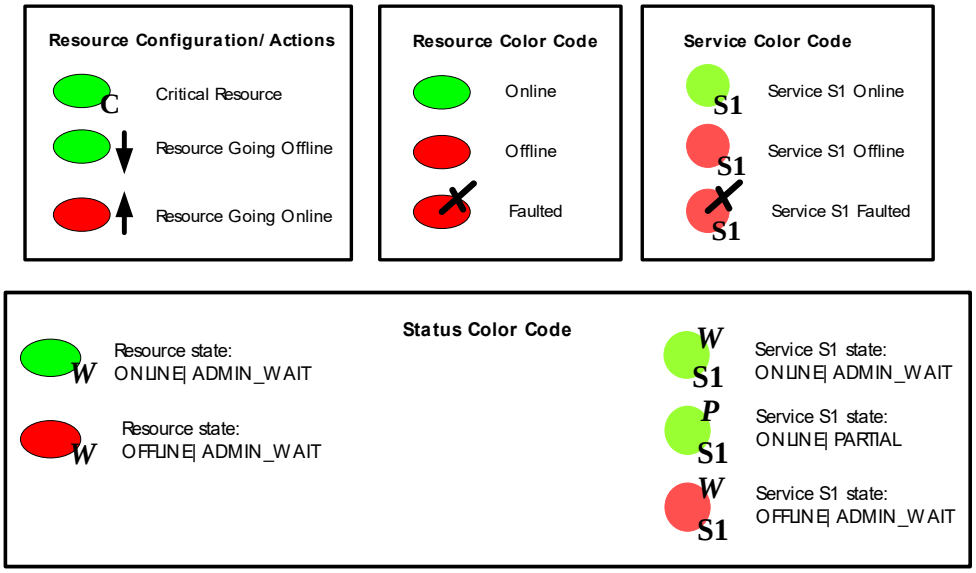
5.1.6 Service Group Failover

If the service priority of a FAULTED service is greater or equal than any online services of a service group, OSC takes the whole service group offline. This means that all services on the affected OSC node are shutdown. If the service group category is either **Failover** or **MultiInstance** and the **SrvGrpFailover** attribute (of the service group is TRUE), OSC takes the service group online on another OSC cluster member. Otherwise the service group is not failed over.

The failover target OSC system is selected based on the configured OSC failover policy defined.

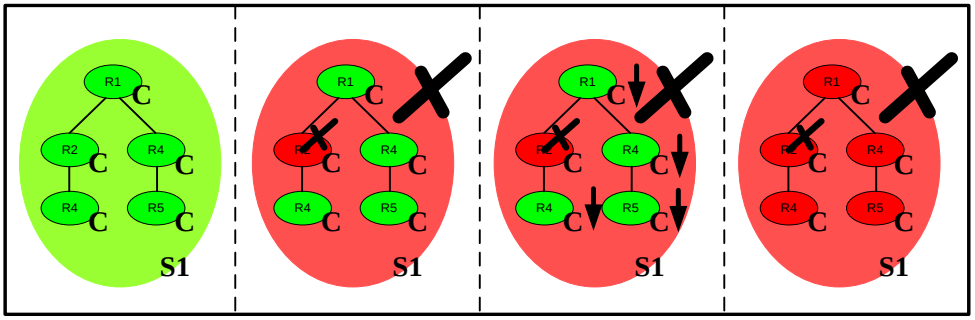
5.1.7 OSC Behavior Diagrams

This section describes the default functionality of OSC when resources fail. The following illustration displays the symbols used in this section.



5.1.7.1 Scenario: Critical resource fails

The service group in the following example contains one service S1. Service S1 contains five resources which are all configured as critical resources. Fault propagation is enabled for the service group.

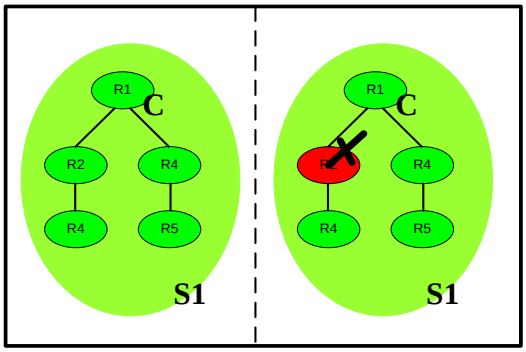


When resource R2 fails, OSC calls the clean action routine and the fault condition is propagated up the dependency tree to service S1. The fault bit in the status field of S1 is set and the fault condition is propagated to the service group. OSC detects that the service group has failed (the service group contains no other services) and shuts down the service

group. OSC takes all resources except the faulted resource offline in the order of their dependencies. The FAULTED resource R2 does not have to be shutdown since it has already been “cleaned up”. After taking resources R1, R3, R4, and R5 offline, OSC fails over the service group to another node if the service group is defined as a **Failover** or **MultiInstance** service group.

5.1.7.2 Scenario: Non-Critical resource fails

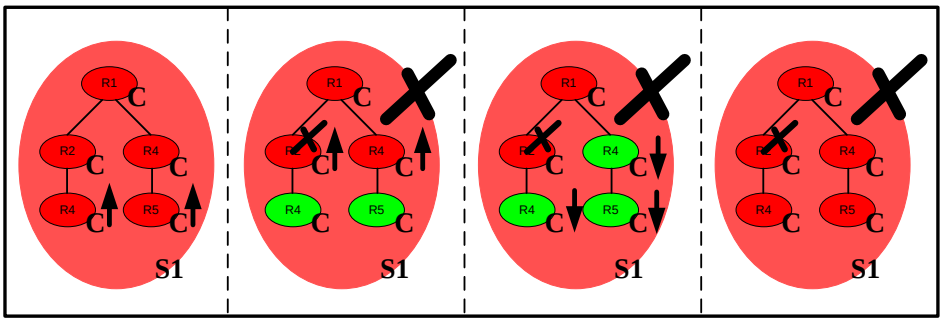
The service group in the following example contains one service S1. Only the resource R1 is defined as critical. Fault propagation is enabled for the service group.



When resource R2 fails, OSC calls the clean action routine. Since R2 is non-critical, fault propagation is not initiated and the service group is not marked faulted. Thus, the resource fault does not cause a service group shutdown and the resource tree is not taken offline.

5.1.7.3 Scenario: Critical resource fails to come online

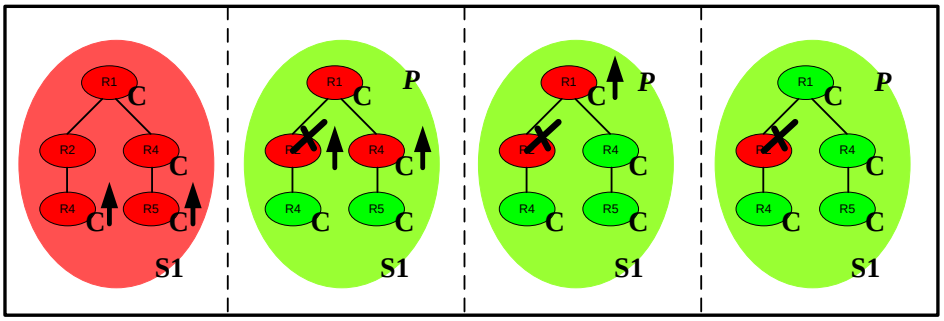
In the following example a service group is issued to go online. All resources of service S1 are critical. Resource R2 fails to come online. Fault propagation is enabled for the service group.



OSC calls the clean action routine for resource R2 and the fault condition is propagated up to the service and service group level. Resource R2 is a critical resource. Thus, the service group is taken offline and failed over to another node in the OSC cluster if the service group is defined as a **Failover** or **MultiInstance** service group.

5.1.7.4 Scenario: Non-Critical resource fails to come online

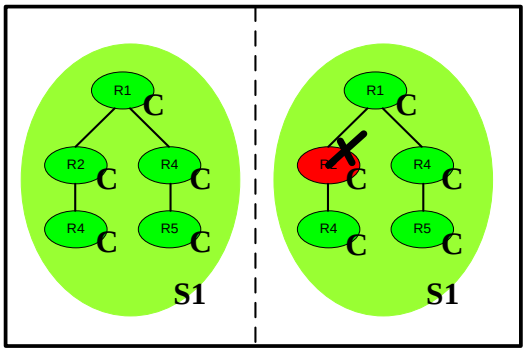
In the following example a service group is issued to go online. The child resource R2 of service S1 is configured as a non-critical resource and fails to come online. Fault propagation is enabled for the service group.



OSC calls the clean action routine for resource R2. Since this resource is non-critical, the fault condition is not propagated up to the service and service group level. Resource R2 is a non-critical resource. Thus, the ONLINE transaction continues to start the service group. After the transaction has completed the final status of service S1 and its parent service group will be ONLINE|PARTIAL.

5.1.7.5 Scenario: Critical resource fails and fault propagation is disabled for the service group

In this example all resources of the service group S1 are critical and fault propagation is disabled for the service group:

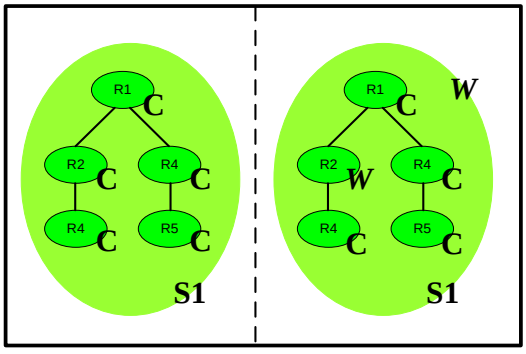


If resource R2 fails, OSC calls the clean action routine. R2 is a critical resource but fault propagation is disabled. Thus, the fault condition is not propagated up to the service and service group level, and the resource fault does not cause a service group shutdown and the resource tree is not taken offline.

5.1.7.6 Scenario: Critical resource fails and fault management is disabled for the service group

In this example all resources of the service group S1 are critical but fault management is disabled for the service group:

Since fault management is disabled the clean action routine is not called if R2 fails. Instead the status of R2, service S1 and the service group is set to ONLINE|ADMIN_WAIT. OSC does not take any other resources of service S1 offline.



5.2 Controlling OSC Failover Policy

Two different OSC failover policies can be configured:

- Static failover
- Load-balanced failover

The OSC cluster-wide failover policy is defined by the keyword assigned to the OSC master control attribute **OscCtrlFailoverPolicy**:

- STATIC
  Static failover policy is selected for the OSC cluster.
- LOAD-BALANCING
  Load balanced failover policy is selected for the OSC cluster

Please refer to the section A.6 OSC master control and service engine attributes for a complete list and description of all OSC master control attributes available.

5.1.7.1 STATIC Failover Policy

If the OSC failover policy is set to STATIC and a service group has to failover to another node or the service group has to be automatically started by OSC, the target node is selected from the execution node list assigned to the service group.

The service group execution node list is a service group attribute (**SrvGrpNodes** – see also 5.3.1 Controlling Service Group Execution Nodes) and is one of the key service group configuration items. It contains a comma separated list of OSC node configuration blocks. An OSC node configuration block contains the SCS node name of an OSC cluster member and the node priority to run the service group. These two attributes are separated by a colon.

The node priority is the selection criterion. OSC determines which OSC node of the service group execution node list the service group is allowed to be started on. Amongst these OSC nodes, the node with the highest node priority assigned is selected to start the affected service group.

A service group is allowed to be started on an OSC cluster member listed in the execution node list if:

- The OSC node is an active member of the OSC cluster

- None of the service groups listed in the service group attribute **SrvGrpDisAllow** are already running (online) or starting up on that OSC node (see also 5.3.2 Controlling Service Group Concurrent Execution).
- The affected service group state on the potential node is NOT one of the following:
    - FAULTED
    - ADMIN_WAIT
    - FROZEN
    - DISABLED

If the service group is not allowed to be started on any OSC cluster member listed in the node execution list (attribute **SrvGrpNodes**) due to the criteria described above, the service group will not be started as a result of an automatic failover or service group startup request.

However, if a user explicitly requests that a service group is started on a particular OSC node, then the service group will start on that OSC node regardless if any service group listed in the service group exclude list (attribute **SrvGrpDisAllow**) is already online on that OSC node or not (see also 7.3.3 How to online a Service Group). The only reason that would prevent a service group from starting on a particular OSC node due to an explicit user request, is if the service group is in one of the states listed below:

- FAULTED
- ADMIN_WAIT
- FROZEN
- DISABLED

5.1.7.2 LOAD-BALANCING Failover Policy

If the OSC failover policy is set to LOAD-BALANCING, OSC uses a load-balancing mechanism to determine which OSC system should host an application (service group) during startup, or after an application or server fault. This load-balancing algorithm requires that the user defines:

- The maximum workload capability of each OSC cluster member (= maximum workload a particular OSC cluster member is able to handle = user defined number).
- Workload values for each service group configured.

The maximum workload capability of all OSC cluster members is defined by the OSC master control attribute **OscCtrlNodeLoadCap**. This attribute contains a comma separated list of OSC workload capability configuration

blocks. An OSC workload capability configuration block contains the SCS node name of an OSC cluster member and the maximum system workload the node is able to handle.

Example:

VMSTM1:200,VMSTM2:200,VMSTM3:100

The OSC workload capability list in this example defines that the OSC cluster member VMSTM1 and VMSTM2 can run service groups with a maximum workload (= sum of the workload values assigned to the service groups) of 200. VMSTM3 can only run service groups up to a maximum workload of 100 since VMSTM3 has less capacity than VMSTM1 or VMSTM2.

The workload value of a specific service group is defined by the service group attribute **SrvGrpLoad**.

When a service group is failed over to another OSC cluster member or the service group has to be automatically started on a node by OSC, OSC determines which OSC node of the service group execution node list (attribute **SrvGrpNodes**) the service group is allowed to be started on and selects from the potential OSC nodes, the OSC nodes that have sufficient free workload capabilities to host the service group. From these OSC nodes the node with the actual largest free OSC workload capability is selected as the target OSC node to start the service group on.

A service group is allowed to be started on an OSC cluster member listed in the execution node list if:

- The OSC node is an active member of the OSC cluster
- None of the service groups listed in the service group attribute **SrvGrpDisAllow** are already running (online) or starting up on that OSC node (see also 5.3.2 Controlling Service Group Concurrent Execution).
- The service group state on the selected node is NOT one of the following:
  - FAULTED
  - ADMIN_WAIT
  - FROZEN
  - DISABLED

If the service group is not allowed to be started on any OSC cluster member listed in the node execution list (attribute **SrvGrpNodes**) due to the criteria described above, or if none of the OSC nodes have sufficient actual free workload capabilities to host the service group the service

group will not be started due to an automatic failover or service group startup request.

However if a user explicitly requests that a service group is started on a particular OSC node, the service group will start on that OSC node regardless if any service group listed in the service group exclude list (attribute **SrvGrpDisAllow**) is already online on that OSC node or not or if the OSC node has sufficient free load capabilities to host the service group (see also 7.3.3 How to online a Service Group). The only reason that would prevent a service group from starting on a particular OSC node due to an explicit user request, is if the service group is in one of the states listed below:

- FAULTED
- ADMIN_WAIT
- FROZEN
- DISABLED

5.3 Controlling OSC Behavior at the Service Group Level

This section provides a detailed description of how service group attributes affect OSC behavior.

Please refer to the section A.9 OSC service group attributes for a complete list and description of all service group attributes available.

5.3.1 Controlling Service Group Execution Nodes

The attribute **SrvGrpNodes** defines the service group execution node list. The service group execution node list defines the OSC cluster members the service group is allowed to be started on. It contains a comma separated list of OSC node configuration blocks. An OSC node configuration block contains the SCS node name of an OSC cluster member and the node priority to run the service group. These two attributes are separated by a colon.

Example: VMSTM1:2,VMSTM2:3,VMSTM3:1

The execution node list in the example above defines that a service group can be started on node VMSTM1, VMSTM2 and VMSTM3. VMSTM2 has the highest execution priority (3) followed by node VMSTM1 (2) and VMSTM3 (1).

If the OSC failover policy is set to STATIC and the service group is a **Failover** service group, OSC initially tries to start the service group on VMSTM2. VMSTM1 is the primary and VMSTM3 the secondary failover node. If the service group is configured as a **MultiInstance** service group, the service group is started on as many OSC nodes as defined by the service group attribute **SrvGrpMultInstance** according to the node priority of the OSC nodes. If the service group is a **Parallel** service group the service group is started on all the OSC members defined in the execution node list regardless of the node priority.

If the OSC failover policy is set to LOAD-BALANCING the node priorities of the OSC nodes are ignored. The OSC node selection algorithm to start **Failover** and **MultiInstance** service group is based on the OSC free workload capabilities of the OSC members listed in the node execution list and the service group workload value (see section 5.1.7.2 LOAD-BALANCING Failover Policy).

5.3.2 Controlling Service Group Concurrent Execution

The user can define that particular service groups are not allowed to run concurrently on the same OSC node. The service group is not allowed to run on a particular OSC node if any of the service groups defined by the **OscSrvDisAllow** attribute is already running or starting up on that OSC member (see also

5.2 Controlling OSC Failover Policy).
The service group exclude list has to be entered as a comma separated list. This attribute only applies to automatic failover and service group startup requests. The service group exclude list is ignored if a user explicitly requests that a service group is started on a particular OSC node (see 7.3.3 How to online a Service Group). In addition, OSC also does not relocate mutually exclusive service groups when OSC starts up and finds that mutually exclusive service groups are online on the same node (OSC implicitly assumes that this is a user defined setup).

Example:

If the **SrvGrpDisAllow** attribute of the service group XMLT contains "ORADWH, MYSQL" the service group XMLT will not be started on a particular OSC node due to an automatic startup or failover request if either of the service groups ORADWH and/or MYSQL is already running on that OSC node, regardless of the OSC failover policy defined (see

5.2 Controlling OSC Failover Policy).

5.3.3 Controlling Service Group Workload Value

The **SrvGrpLoad** attributes defines the workload value of the service group. This service group attribute is mandatory if the selected OSC failover policy is LOAD-BALANCING (see section 5.1.7.2 LOAD-BALANCING Failover Policy)

5.3.4 Controlling automatic Service Group startup

The **SrvGrpAutoStart** attribute defines whether or not OSC automatically starts the service group (if it is not already online – keep in mind a service group represents a particular application which can be started without using OSC) when OSC starts up.

5.3.5 Controlling Clean Behavior on Resource Faults

The **SrvGrpManageFault** attribute specifies whether or not OSC calls the clean action routine when a resource fails. Each service group can be configured individually of how it should react to resource failures.

- If the value of the **SrvGrpManageFault** attribute is TRUE, OSC calls the clean action routine when an **On-Off** resource fails. If an **On-Only** or **Persistent** resource fails the resource is marked FAULTED without calling a clean action routine.
- If the value of the **SrvGrpManageFault** attribute is FALSE, OSC takes no action on a resource fault. It "hangs" the resource and the associated service and service group until system management has intervened. OSC marks the state of the failed resource, the affected service and the affected service group as ADMIN_WAIT and removes the failed resource from status monitoring until system management has fixed the fault condition and the ADMIN_WAIT state has been manually cleared via the OSC$MGR utility.

Note

If a resource is a member of several service groups and only one of these service groups is disabled to manage faults, the clean action routine will not be called if that particular resource fails. All service groups the resource is a member of will be set to ADMIN_WAIT.

OSC triggers a fatal ADMIN_WAIT state change event message when a resource, service or service group enters the ADMIN_WAIT state. Any fatal OSC event message signals the need for immediate system management intervention, since OSC cannot automatically recover the resource fault in this state.

5.3.6 Controlling Fault Propagation

The **SrvGrpFaultProp** attribute defines, provided that fault management is enabled, whether a resource fault is propagated up the resource dependency tree to the service and service group level.

- If the value of the **SrvGrpFaultProp** attribute is set to TRUE (default), a resource fault is propagated up the resource dependency tree. If the faulted resource is **Critical**, the service group is taken offline and failed over, provided the **SrvGrpFailover** attribute is set to TRUE (default) and the service group is not a **Parallel** service group.
- If the **SrvGrpFaultProp** is set to FALSE, resource faults are contained at the resource level. OSC does not take the service group offline, thus preventing failover.

The **SrvGrpFaultProp** attribute is a configuration attribute and cannot be changed during run-time.

5.3.7 Controlling Failover on Service Group or System Faults

The **SrvGrpFailover** attribute defines the service group behavior in response to service group and system faults.

- If the **SrvGrpFailover** attribute is set to TRUE, the service group fails over when a system or a service group fails, provided that a suitable OSC system exists for failover, and the service group is not a **Parallel** service group.
- If the **SrvGrpFailover** attribute is set to FALSE the service group does not failover when a system or service group fails. If a fault occurs in a service group, the service group is taken offline. If an OSC system fails (crash), the service group is not started on another system.

The **SrvGrpFailover** attribute is a configuration attribute and cannot be modified during run-time.

5.3.8 Controlling Behavior on external Resource startup

The service group configuration attribute **SrvGrpCleanOnUnexpOn** specifies the behavior of OSC when OSC detects that an **On-Off** resource of the service group was started outside of OSC control:

- If the value of the **SrvGrpCleanOnUnexpOn** attribute is FALSE (default), OSC marks the resource, the affected service and the service group state as ADMIN_WAIT. Thus, the service group is removed from resource fault and failover handling until system management has cleaned up the situation and the ADMIN_WAIT state is manually cleared by use of the OSC$MGR utility. OSC triggers a fatal ADMIN_WAIT state change event message for immediate system management intervention.
- If the value of the **SrvGrpCleanOnUnexpOn** attribute is TRUE, OSC unconditionally takes this **On-Off** resource offline.

This attribute does not apply to **On-Only** or **Persistent** resources, since these resources are cluster aware and thus no concurrency violation risk exists even if such a resource was started outside of OSC control. Thus, OSC takes no action if an **On-Only** or **Persistent** resource unexpectedly changes state from offline to online – it just updates the status of the resource.

5.3.9 Controlling online Behavior of MultiInstance and Parallel Service Group containing cluster locked resources

The service group configuration attribute **SrvGrpOnlineLocked** controls the online behavior of **MultiInstance** and **Parallel** service groups if the service group contains cluster locked resources.

A cluster locked resource is a resource that will be started on only one OSC cluster member regardless if the service group that owns this resource is configured to run concurrently on different nodes (**MultiInstance** or **Parallel** service group – see also section 5.5.1 Resource Type Attributes).

If a service group starts up on any OSC node and detects that a cluster locked resource is already online on any other node the value of this Boolean attributes decides whether or not OSC starts the parent resources of this particular cluster locked resource or not.

If the value of this attribute is FALSE, OSC will not start the parent resource(s) of the cluster locked resource, if the cluster locked resource cannot be started due to the lock state (already online on another node) of the cluster locked resource. If the value of this attribute is TRUE, the parent resource(s) will be started.

The default value of the **SrvGrpOnlineLocked** attribute is TRUE.

5.3.10 Freezing Service Groups

Freezing a service group prevents OSC from taking any action when the service group or an OSC system fails. Freezing a service group prevents the service group from going offline due to a resource fault and thus the service group will not failover. It also prevents clean action routines from being called for faulted resources of the service group.

A service group can be 'frozen' on the whole OSC cluster or just on a particular OSC cluster member. If you freeze a service group on a particular OSC node and the actual state of the service group on that node is OFFLINE, the OSC node will then not be elected as the failover target for the service group. If you freeze a service group on an OSC node that is currently online on that node, this will prevent OSC to offline the service group and to subsequently failover the service group to another OSC cluster member.

Freeze a service group if you want to:

- Exclude a particular OSC node temporarily from becoming the potential failover target node for this particular service group.
- Temporarily prevent that a service group fails over to another node due to a resource failure reported by an OSC agent.

Note

The freeze flag attribute of a service group is not a configuration, but a run-time attribute. Thus, when you restart the OSC environment on a node, a service group that had previously been 'frozen' will now appear as 'unfreezed' and be subject of the OSC ruleset after the OSC restart.

Freezing a service group implicitely freezes all services and all **On-Off** resources that are resources that are members of that service group, but not **On-Only** or **Persistent Persistent** resources. To freeze **On-Only** or **Persistent** resources these resources have to

resources have to be explicitely 'frozen' with the FREEZE RESOURCE command (see section 5.5.1.16 ScriptExecUser Attribute

Starting with VSI OpenVMS ServiceControl V3.5 the resource attribute **ScriptExecUser** is available. This attribute defines the user account used to execute DCL action scripts at the resource level. If this attribute is defined for a particular resource the OSC agent executes the DCL action scripts in the context of the user defined in this attribute. If the attribute value defined is invalid (user defined is not found in SYSUAF), or this attribute is empty, the user context of the OSC agent is used (this is equivalent to the pre-V3.5 behavior) to run the DCL action scripts.

The great value of this attribute is that if different resources are to be started using the same DCL script (online action routine), but it is also a requirement that for each of these resources this start DCL script is executed in a different user context, they can all be handled by a single OSC agent. Prior to V3.5 different OSC agents had to be created for each of the resources to meet this requirement, since prior to V3.5 DCL action scripts were always executed in the user context of the OSC agent.

Note

This attribute has no effect on compiled action routines. Compiled action routines are started in the same user context as the OSC agent has been started with.

5.5.2 Freezing Resources).

To freeze a *service group*:

OSC$MGR> FREEZE SRVGRP *servicegroup-name* [/node=*node-name*]

To unfreeze a *service group*:

OSC$MGR> UNFREEZE SRVGRP *servicegroup-name* [/node=*node-name*]

For detailed information about the FREEZE and UNFREEZE commands please refer to the OSC$MGR online help.

5.3.11 Disabling Service Groups

The effect of disabling a service group is similar to freezing a service group. It prevents OSC from taking any action on disabled service groups. The main differences between freezing and disabling a service group are:

- When a service group is disabled all **On-Off** resources of its services are removed from status monitoring. Thus, once a service group is disabled the user will not be notified about state changes of all its **On-Off** resources. In contrast 'freezing' a service group does not stop status monitoring of its resources
- Disabling a service group sets the **SrvGrpDisabled** flag of the service group. The **SrvGrpDisabled** attribute is a configuration and run-time attribute. Thus, once a service group is disabled it remains disabled even when the whole OSC environment is restarted.

Disable a service group if you want to execute operator commands on its resources outside of OSC control, and you do not want to be notified about status changes of all the **On-Off** resources of that service group. For example, disable a database service group before you manually start and stop the database.

As with 'freezing' a service group, a service group can be disabled on the whole OSC cluster or on particular OSC nodes.

Note

The disable flag of a service group is a configuration attribute. Thus, once a service group is disabled it remains disabled even if the whole OSC environment is restarted.

Disabling a service group implicitely disables all services and all **On-Off** resources that are members of the service group, but not **On-Only** or **Persistent** resources. To disable **On-Only** or **Persistent** resources, these resources have to be explicitely disabled with the DISABLE RESOURCE command (see section 5.5.3 Disabling Resources).

To disable a *service group*:

OSC$MGR> DISABLE SRVGRP *servicegroup-name* [/node=*node-name*]

To enable a disabled *service group*:

OSC$MGR> ENABLE SRVGRP *servicegroup-name* [/node=*node-name*]

For detailed information about the DISABLE and ENABLE commands please refer to the OSC$MGR online help.

5.4 Controlling OSC Behavior at the Service Level

This section describes how service attributes effects OSC behavior.

Please refer to the section A.8 OSC service attributes for a complete list and description of all the service attributes available.

5.4.1 ServicePriority Attribute

The **ServicePriority** attribute defines the "importance" of a service within a service group.

A service is the management unit of a resource set that represents a dedicated application (service). The OSC failover unit is a service group. A service group may contain 1 to n services. Services may depend on each other within a service group. It is also possible that a service group contains services that do not depend on each other in terms of ordered online and offline processing. If a service group contains independent services, these can be started and stopped in parallel. Nevertheless since they are defined as part of the same service group they will always run together on the same OSC node. A typical example for grouping two independent services into a service group is when there are two applications that can run independently of each other, but exchange data via mailboxes if the partner application is present.

When a resource fails the fault condition is propagated up to the service and service group level and OSC initiates offline/failover processing. OSC first checks the values of the **ServicePriority** attribute of the top level services of the independent service trees configured within a service group. If the value of the **ServicePriority** attribute of an online service tree is greater than the value of the failed service tree, no service group failover is initiated. Otherwise all services are taken offline and the whole service group is failed over to another node in the OSC cluster as specified by the OSC configuration.

The propagation of a failed critical resource up to the service group level is influenced by the definition of this service attribute amongst the different services in the respective service group.

Unlike a resource, a service cannot be shared between different service groups. A service is service group specific.

5.4.2 Freezing Services

Services can be explicitely 'frozen'. Freezing a service implicitely freezes all of its **On-Off** resources. Freezing a resource prevents the OSC agent from launching any action routines except the monitor routine. Thus, if a service is 'frozen' neither the online, offline nor the clean action routines are called for any of its **On-Off** resources.

When a service is 'frozen' the service group the service is a member of is automatically marked 'frozen' too.

Thus, the effect of 'freezing' a service is comparable to 'freezing' a service group (see section 5.3.10 Freezing Service Groups). The main difference is that if you only 'freeze' a particular service the **On-Off** resources of this service are 'frozen'. Other resources that are members of the same service group, but are not members of the 'frozen' service are not affected.

You can 'freeze' a service on the whole OSC cluster or on particular OSC cluster members.

Freezing a service enables the user to modify and test online existing OSC agent Monitor routine (script) for the **On-Off** resources of the service that has been 'frozen' without the risk that OSC initiates any management operations (failover, offline) on the parent service groups (your applications), due to a bug in the monitor routine that provides wrong or incomplete resource status information.

Note

The freeze flag of a service is not a configuration but a run-time attribute. Thus, when you restart the OSC environment on an OSC node, a service that has previously been 'frozen' will now be 'unfreezed' when OSC restarts.

As has been previously stated, 'freezing' a service group implicitely 'freezes' all **On-Off** 'freezes' all **On-Off** resources that are members of the service, but not **On-Only** or **On-Only** or **Persistent** resources. To freeze **On-Only** or **Persistent** resources these resources these resources have to be explicitely 'frozen' with the FREEZE RESOURCE RESOURCE command (see section 5.5.1.16 ScriptExecUser Attribute

Starting with VSI OpenVMS ServiceControl V3.5 the resource attribute **ScriptExecUser** is available. This attribute defines the user account used to execute DCL action scripts at the resource level. If this attribute is defined for a particular resource the OSC agent executes the DCL action

scripts in the context of the user defined in this attribute. If the attribute value defined is invalid (user defined is not found in SYSUAF), or this attribute is empty, the user context of the OSC agent is used (this is equivalent to the pre-V3.5 behavior) to run the DCL action scripts.

The great value of this attribute is that if different resources are to be started using the same DCL script (online action routine), but it is also a requirement that for each of these resources this start DCL script is executed in a different user context, they can all be handled by a single OSC agent. Prior to V3.5 different OSC agents had to be created for each of the resources to meet this requirement, since prior to V3.5 DCL action scripts were always executed in the user context of the OSC agent.

Note

This attribute has no effect on compiled action routines. Compiled action routines are started in the same user context as the OSC agent has been started with.

5.5.2 Freezing Resources).

To freeze a Service

OSC$MGR> FREEZE SERVICE *service-name* [/node=*node-name*]

To unfreeze a Service

OSC$MGR> UNFREEZE SERVICE *service-name* [/node=*node-name*]

For detailed information about the FREEZE and UNFREEZE commands please refer to the OSC$MGR online help.

5.4.3 Disabling Services

Services can be explicitely disabled. Disabling a service removes its **On-Off** resources from status monitoring. When a service is disabled the current status of its **On-Off** resources remains unchanged until the service is enabled again.

When a service is disabled the service group, the service is member of, is automatically disabled too.

Thus, the effect of disabling a service is comparable to disabling a service group (see section 5.3.11 Disabling Service Groups). The main difference is that if you disable a particular service only the resources of that service are disabled. Other resources that are members of the same service group, but not members of the disabled service are unaffected.

Disable a service if you want to execute operator commands on **On-Off** resources of the service outside of OSC control and you do not want to be notified about the status change while you are working on these resources. For example disable a database resource before you manually start and stop the database and its listener processes.

As with 'freezing' a service, a service can be 'disabled' on the whole OSC cluster or on particular OSC nodes.

Note

The disable flag of a service is a configuration attribute. Thus, once a service is disabled it remains disabled even the whole OSC environment is restarted.

As has been previously stated, disabling a service implicitely disables all **On-Off** resources that are members of the service, but not **On-Only** or **Persistent** resources. To disable **On-Only** or **Persistent** resources, these resources have to be explicitely disabled with the DISABLE RESOURCE command (see section 5.5.3 Disabling Resources).

To disable a Service

OSC$MGR> DISABLE SERVICE *service-name* [/node=*node-name*]

To enable a disabled Service

OSC$MGR> ENABLE SERVICE *service-name* [/node=*node-name*]

For detailed information about the DISABLE and ENABLE commands please refer to the OSC$MGR online help.

5.5 Controlling OSC Behavior at the Resource Level

This section describes how OSC behavior can be controlled at the resource level. Note that a resource is not considered as faulted until the agent framework forwards the fault to the OSC service engine.

5.5.1 Resource Type Attributes

The following attributes affect how the OSC agent framework reacts to problems with individual resources before a resource is considered as faulted and the fault condition is forwarded to the OSC service engine.

Please refer to the section A.7 OSC resource attributes for a complete list and description of all resource attributes available.

5.5.1.1 ClusterLocked Attribute

The **ClusterLocked** attribute defines whether or not the resource is a cluster-locked resource. A cluster locked resource is a resource that will be started only on one node within the OSC cluster, regardless if the service group that this resource is a member of, is configured to run concurrently on different nodes (**MultiInstance** or **Parallel** service group).

MultiInstance and **Parallel** service groups are started on several or all OSC cluster members (see section 4.2.3.1 Service Group). Starting a service group on a node means starting all its respective services and resources. Thus, resources of a **MultiInstance** or **Parallel** service group can be active concurrently on several OSC nodes, except if a resource of such a service group is configured as a cluster locked resource. In this case the resource will be started on only one of the OSC nodes where the service group has been started.

Only **On-Off** resources can be defined as cluster locked resources. This attribute is ignored if the resource is either an **On-Only** or a Peristent resource.

If the value of this Boolean attribute is set to TRUE, OSC considers the resource a cluster locked resource.

When a **MultiInstance** or **Parallel** service group is started, OSC selects the nodes to start the service group from the service group node execution list (see attribute description of **SrvGrpNodes** in section 5.3.1 Controlling Service Group Execution Nodes). If all nodes defined in the service group node execution list are up and running and the service group is a **Parallel** service group all the nodes will be selected. If some of the nodes are down, or the service group is configured as a **MultiInstance** service group, a subset of the node execution list will be selected.

OSC selects the node to start a cluster locked resource only from this set of nodes. The primary node selection criterion is the node priority (see description of the **SrvGrpNodes** attribute in section 5.3.1 Controlling Service Group Execution Nodes). OSC searches the node set starting from the node with the highest priority down to the node with the lowest priority and checks if the cluster locked resource is allowed to be started. A cluster locked resource is allowed to be started on a node if no resource listed in the **CluLckResDisAllow** attribute (see next section) is already online on this node. OSC starts the cluster locked resource on the first node that provides the permissions to run the resource on, according to the criteria described above, but not on any other node of the service group node set. If OSC fails to find a node with the necessary permissions to run the cluster locked resource on, the resource will not be started.

If a service group fails for any reason on an OSC node where one of its resources is a cluster locked resource which is online, the cluster locked resource will be stopped on this node and started on another node according to the node selection criteria described above.

If a service group contains a cluster locked resource, the service group is considered ONLINE on all the respective OSC nodes, if the cluster locked resource is online on this one node.

Example:

The service group TEST contains the cluster locked resource PRC::TEST and is configured to run as a **Parallel** service group on node VMTM1, VMSTM2 and VMSTM4. Since PRC::TEST is a cluster locked resource it is started only on one of these nodes. In this example, we will assume that PRC::TEST is online on VMSTM1. If all resources except the cluster locked resource PRC::TEST of the service group TEST are online on VMSTM1, VMSTM2 and VMSTM4 the status of the service group TEST will be ONLINE on all three nodes.

The big advantage of cluster locked resources are that cluster aware and non-cluster aware resources can be bundled into **MultiInstance** and **Parallel** service groups.

Usage example:

You are running a RDB database on an OpenVMS cluster. A remote client application accesses this RDB database via a service IP address. It is a requirement that the RDB database is open for access cluster-wide, because other applications running on OpenVMS, scripts and interactive SQL sessions directly access the same database from any cluster member. To guarantee that the RDB database is opened cluster-wide by OSC, the RDB database has to be configured as a resource of a **Parallel** service group. From the client point of view the whole service consists of the availability of the RDB database via the service IP address. Thus, it makes sense to put both resources into one service group. The problem is that the service IP address is a non-cluster aware resource, because it can only be assigned to a network adapter of one of cluster nodes, at any one time

If the service IP address is not configured as a cluster locked resource it would have to be a member of a different **Failover** service group.

The disadvantage of configuring the IP address as a member of a **Failover** service group and the RDB database as a member of a different **Parallel** service group is that OSC handles all service groups independently from each other (see section 4.2.3 Service Groups). That would mean that the service IP address and the RDB database are handled independently of each other even though they depend on each other. This may cause several problems. For example, if the RDB database failed on the node where the service IP address was assigned, the service IP address would be failed over automatically without system management intervention. In addition, the information that these two resources depend on each other is lost, since OSC does not permit the configuration of dependencies between resources of different service groups (see also 8.2 OSC configuration rules)

If the service IP address is configured as a cluster locked resource, it can now be a member of the same **Parallel** service group that the RDB database resource is a member of. Putting both resources into one service group overcomes all the problems described above.

5.5.1.2 CluLckResDisAllow Attribute

The user can define a list of cluster locked resources that are mutually exclusive i.e. are not allowed to run on the same node. A cluster locked resource is not allowed to run on a particular node if any of the cluster locked resources defined by the **CluLckResDisAllow** attribute are already online or starting up on that OSC node. The resource exclude list has to be entered as a comma separated list. This attribute applies to cluster locked resources only. If the resource is not configured as a cluster locked resource (see the description of the **ClusterLocked** attribute in the previous section) this attribute is ignored.

The resources listed in the **CluLckResDisAllow** attribute must be cluster locked resources too. OSC will not start up if this attribute of a cluster locked resource contains non cluster locked resources.

The cluster locked resource exclude list only applies to automatic failover and service group startup requests. It is ignored if a user requests to stop a cluster locked resource on one node and to start this same cluster locked resource on another node. In this case the cluster locked resource will start, even if a resource listed in the cluster locked exclude list is already online on this particular node. In addition, OSC will not relocate mutually exclusive cluster locked resources when OSC starts up, and it finds that mutually exclusive cluster locked resources are online on the same node (OSC implicitly assumes that this is a user defined setup).

Usage example:

Let us expand on the usage example of section 5.5.1.1 ClusterLocked Attribute. Let us assume that an additional remote client access to the RDB database is available via a different service IP address. For load balancing reasons, these two remote clients shall not access the RDB database via the same OpenVMS cluster member. This is a typical example of when to configure a cluster locked exclusion list. Both service IP addresses have to be configured as cluster locked resources so that they can be added to the same **Parallel** service group that the RDB database is a member of. In addition these two service IP addresses must never be assigned to the network adapters of the same node. This functionality can be simply achieved, if one adds the resource name of the second service IP address to the **CluLckResDisAllow** attribute of the first service IP address resource and vice versa.

5.5.1.3 RestartLimit Attribute

The **RestartLimit** attribute defines the number of times OSC attempts to restart an **On-Off** resource due to an unexpected offline fault, before forwarding the fault condition to the OSC service engine.

If the value of the **RestartLimit** attribute of a resource is not zero, the OSC agent attempts to restart the resource as often as defined by this attribute before declaring the resource as faulted. When restarting a failed resource, the agent framework calls the clean action routine before calling the online action routine. However, if fault management is disabled the clean action routine is not called and hence the OSC agent will not retry to set this **On-Off** resource online.

This **RestartLimit** attribute does not apply to **On-Only** and **Persistent** resources since the prerequisite for a restart attempt is that the resource can be shutdown reliably in advance. OSC does not call a clean action routine for **On-Only** and **Persistent** resources and hence if these resources fail the status is undefined from the OSC perspective. Fault conditions of these resource types are always reported to the OSC service engine, without any attempts to recover this situation.

5.5.1.4 ToleranceLimit Attribute

The **ToleranceLimit** attribute defines the number of times the monitor routine has to consecutively return an offline status for a resource that is considered online, before OSC consideres the resource as unexpected offline. This attribute is typically used when a resource is monitored that monitors its own operational state and is capable of restarting itself if an invalid operational state is detected. Setting the attribute to a non-zero value instructs OSC to allow multiple offline response monitor cycles, with the expectation that the resource will get back into the online state within the tolerance limit. Setting a non-zero **ToleranceLimit** extends the time required for OSC to respond to an actual fault.

5.5.1.5 FaultOnMonitorTmo Attribute

The **FaultOnMonitorTmo** attribute defines whether or not OSC considers a monitor routine timeout as a resource fault.

If the value of this attribute is FALSE, OSC does not treat monitor timeouts as a resource fault. If the value of the attribute is TRUE, monitor routine timeouts are considered as a resource fault.

5.5.1.6 FaultOnMonitorTmoLimit Attribute

The **FaultOnMonitorTmoLimit** defines the number of times the monitor routine can consecutively time out, before the resource is considered faulted.

This attribute is typically used if the execution time of the monitor routine varies depending on the availability of system resources and/or the workload load of the monitored resource.

This attribute has no effect if the value of the attribute **FaultOnMonitorTmo** is set to FALSE.

5.5.1.7 OnlineRetryLimit Attribute

The **OnlineRetryLimit** attribute specifies the number of times the online action routine is retried if the initial attempt to bring a resource online fails.

When the value of the **OnlineRetryLimit** attribute of a resource is not zero, the agent framework calls the clean action routine before re-running the online action routine. However, if fault management is disabled at the service group level the clean action routine is not called and thus the OSC agent does not retry to set the resource online.

This attribute does not apply to **On-Only** and **Persistent** resources since the prerequisite for an online retry attempt is that the resource has been reliably shutdown in advance. OSC does not call a clean action routine for **On-Only** and **Persistent** resources and hence if these resources fail the status is undefined from the OSC perspective. Fault conditions of these resource types are always reported to the OSC service engine without any attempts to recover the situation.

5.5.1.8 OnlineWaitLimit Attribute

Whenever a resource is put online the online action routine is triggered. The return code of the Online routine signals the numbers of seconds OSC waits before it triggers the monitor routine to verify if the resource is actually online.

The **OnlineWaitLimit** attribute defines the number of the times the Monitor routines can consecutively return an offline status, after an attempt to set a resource online, before OSC considers the transaction to put the resource online as failed.

This attribute does not apply to **Persistent** resources since resources of this type cannot be started by OSC.

5.5.1.9 OnlineTmoWaitLimit Attribute

The **OnlineTmoWaitLimit** attribute defines the maximum number of online timeout intervals OSC waits for the completion of the online routine, before OSC triggers online timeout processing.

This attribute is typically used if the execution time of the online routine varies depending on the current state of the resource. For example, the startup time and thus the execution time of the online routine for an Oracle database depends whether or not a database recovery has to be performed.

This attribute does not apply to **Persistent** resources since resources of this type cannot be started by OSC.

5.5.1.10 OfflineWaitLimit Attribute

Whenever a resource is taken offline the offline action routine is triggered. The return code of the offline routine signals the numbers of seconds OSC waits before it triggers the monitor routine to verify if the resource is actually offline.

The **OfflineWaitLimit** attribute defines the number of times the monitor routines should return online status after an attempt to put a resource offline, before OSC considers the transaction to put the resource offline as failed.

This attribute does not apply to **On-Only** and **Persistent** resources since resources of these types cannot be stopped by OSC.

5.5.1.11 OfflineTmoWaitLimit Attribute

The **OfflineTmoWaitLimit** attribute defines the maximum number of offline timeout intervals OSC waits for the completion of the offline routine before OSC triggers offline timeout processing.

This attribute is typically used if the execution time of the offline routine varies depending on the current state of the resource.

This attribute does not apply to **On-Only** and **Persistent** resources since resources of these types cannot be stopped by OSC.

5.4.1.12 ConfInterval Attribute

The **ConfInterval** attribute defines how long a resource must remain online without encountering problems, before previous problem counters are cleared. This attribute controls when OSC clears the **RestartCnt**, **ToleranceCnt** and **CurrentMonitorTmoCnt** values.

5.5.1.13 CleanRetryLimit Attribute

If the clean routine signals to OSC that it has failed to "clean up" a resource the **CleanRetryLimit** attribute defines the number of times an OSC agent retries to execute the clean routine, before OSC accepts that the resource cannot be shutdown forcibly and that the subsequent resource status is set to ADMIN_WAIT.

The ADMIN_WAIT state is propagated up to the service and service group level and the affected service groups are removed from offline and failover processing. Fatal ADMIN_WAIT state change event messages are triggered for immediate system management intervention, since OSC is now unable to control the affected resources, services and service groups.

This attribute does not apply to **On-Only** and **Persistent** resources since OSC does not call clean action routines for resources of these types in response to a resource fault.

5.5.1.14 TimeOutRetryLimit Attribute

The **TimeOutRetryLimit** attribute defines the number of retries to

- Online
- Offline
- Clean

an **On-Off** resource, if the appropriate action routine has timed out. An action routine times out if the OSC agent receives no return code from the action routine within the time periods listed below:

- Online: **OnlineTmo** * (**OnlineTmoWaitLimit** + 1)
- Offline: **OfflineTmo** * (**OfflineTmoWaitLimit** + 1)
- Clean: **CleanTmo**

where:

- **OnlineTmo** Online routine timeout interval [s]
- **OfflineTmo** Offline routine timeout interval [s]
- **CleanTmo** Clean routine timeout interval [s]

If the retry attempts exceed the value of the **TimeOutRetryLimit** attribute for the online, offline or clean action routine the resource status is set to ADMIN_WAIT.

The ADMIN_WAIT resource state is propagated up to the service and service group level and the affected service groups are removed from offline and failover processing. Fatal ADMIN_WAIT state change event messages are triggered for immediate system management intervention, since OSC is now unable to control the affected resources, services and service groups.

This attribute does not apply to **On-Only** and **Persistent** resources. OSC does not call online, offline or clean action routines for **Persistent** resources. The online action routine is called for **On-Only** resources. OSC does not retry to bring an **On-Only** resource online when the initial online attempt to put a resource online times out. An online retry attempt requires the execution of the clean action routine that is not available for **On-Only** resources (see also the description of the **OnlineRetry** attribute in this section). Thus, if the online action routine times out for an **On-Only** resource, OSC starts ADMIN_WAIT processing as described above without any preceding timeout recovery actions.

5.5.1.15 DisableManageFault Attribute

If the **DisableManageFault** attribute is set to TRUE the clean action routine is not called for a particular resource when the resource fails, even if the value of the **SrvGrpManageFault** attribute of the service groups that own the resource is TRUE. The resource status is set to ADMIN_WAIT and the state is propagated up to service and service group level. Thus, the service groups of the resource are removed from fault and failover handling until the condition has been cleared by system management.

This attribute is typically used if the offline status of a resource indicates that it is very likely that the appropriate service group cannot be started on any other node in the OSC cluster. For example, if the OSC shadow set agent returns offline for a particular shadow set it is very likely that the shadow set is not accessible from any other OpenVMS cluster member. Thus, it might be senseless to failover the associated service groups since these may not be able to start up on the other OSC nodes.

5.5.1.16 ScriptExecUser Attribute

Starting with VSI OpenVMS ServiceControl V3.5 the resource attribute **ScriptExecUser** is available. This attribute defines the user account used to execute DCL action scripts at the resource level. If this attribute is defined for a particular resource the OSC agent executes the DCL action scripts in the context of the user defined in this attribute. If the attribute value defined is invalid (user defined is not found in SYSUAF), or this attribute is empty, the user context of the OSC agent is used (this is equivalent to the pre-V3.5 behavior) to run the DCL action scripts.

The great value of this attribute is that if different resources are to be started using the same DCL script (online action routine), but it is also a requirement that for each of these resources this start DCL script is executed in a different user context, they can all be handled by a single OSC agent. Prior to V3.5 different OSC agents had to be created for each of the resources to meet this requirement, since prior to V3.5 DCL action scripts were always executed in the user context of the OSC agent.

Note

This attribute has no effect on compiled action routines. Compiled action routines are started in the same user context as the OSC agent has been started with.

5.5.2 Freezing Resources

Freezing a resource prevents the OSC agent to perform any action routine except the monitor routine. Thus, if a resource is 'frozen' neither the online, offline nor the clean action routine is called.

When a resource is 'frozen' all services and service groups the resource is a member of are also automatically marked 'freezed'.

Thus, the effect of 'freezing' a resource is comparable to 'freezing' a service group or service (see section 5.3.10 Freezing Service Groups and 5.4.2 Freezing Services). The main difference is that if you freeze a particular service group (service) only this service group (service) is 'frozen' even though the service group (service) may contain resources that are members of other service groups (services) too, because only **On-Off** resources are implicitely 'frozen' when a service group or service is 'frozen'. With the FREEZE RESOURCE command any resource type (**On-Off**, **On-Only** and **Persistent**) can be 'frozen'.

You can 'freeze' a resource on the whole OSC cluster or on a particular OSC cluster member.

Freezing a resource enables the user to modify and test online an existing OSC agent Monitor routine (script) without the risk that OSC initiates any management operations (failover, offline) on the service groups (your applications) due to a bug in the monitor routine that provides wrong or incomplete resource status information.

Note

The freeze flag of a resource is not a configuration but a run-time attribute. Thus, when you restart the OSC environment on an OSC node, a resource that was previously set 'frozen' will now be 'unfreezed' after an OSC restart.

To freeze a Resource

OSC$MGR> FREEZE RESOURCE *resource-name* [/node=*node-name*]

To unfreeze a Resource

OSC$MGR> UNFREEZE RESOURCE *resource-name* [/node=*node-name*]

For detailed information about the FREEZE and UNFREEZE commands please refer to the OSC$MGR online help.

5.5.3 Disabling Resources

Disabling a resource removes the resource from status monitoring. When a resource is disabled the current status of the resource remains unchanged until the resource is enabled again since the OSC agent does execute any action routines for disabled resources.

When a resource is disabled all services and service groups the resource is a member of are automatically marked disabled too.

Thus, the effect of disabling a resource is comparable to disabling a service group or service (see section 5.3.11 Disabling Service Groups and 5.4.3 Disabling Services). The main difference is that if you disable a particular service group (service) only this service group (service) is disabled, even though the service group (service) may contain resources that are members of other service groups (services) too, because only **On-Off** resources are implicitely disabled when a service group or service is disabled. With the DISABLE RESOURCE command any resource type (**On-Off**, **On-Only** and **Persistent**) can be disabled.

You can disable a resource on the whole OSC cluster or on a particular OSC cluster member.

Disable a resource if you want to execute operator commands outside of OSC control and you do not want to be notified about the status change while you are working on that resource. For example disable a database resource before you manually start and stop the database.

As with 'freezing' a resource, resources can be disabled on the whole OSC cluster or on particular nodes.

Note

The disable flag of a resource is a configuration attribute. Thus, once a resource is disabled it remains disabled even if you restart the whole OSC environment.

To disable a Resource

OSC$MGR> DISABLE RESOURCE *resource-name* [/node=*node-name*]

To enable a disabled Resource

OSC$MGR> ENABLE RESOURCE *resource-name* [/node=*node-name*]

For detailed information about the DISABLE and ENABLE commands please refer to the OSC$MGR online help.

5.5.4 OSC agent monitor routine return codes

The return code of the monitor routine defines the current state of a resource. The monitor routine may return:

- OSC$_ONLINE (exit code 1) resource is online
- OSC$_OFFLINE (exit code 9) resource is offline
- OSC$_FAULTED (exit code 19) resource has faulted
- Any other valid OpenVMS failure code

Typically the monitor routine returns either OSC$_ONLINE or OSC$_OFFLINE for a dedicated resource. Based on the return code an OSC agent determines whether or not a fault condition occurred for the resource:

- Unexpected Online
  The resource status has changed from offline to online without being requested to go online by OSC. The resource has been started outside of OSC control
- Unexpected Offline
  The resource status has changed from online to offline without being requested to go offline by OSC.
- Online ineffective
  The resource status is offline although OSC requested the resource to go online
- Offline ineffective
  The resource status is online although OSC requested the resource to go offline

If OSC detects one of these fault conditions listed above the OSC agent initiates the fault handling based on the resource attributes described in the previous section.

The monitor return code OSC$_FAULTED signals to the OSC agent that the resource is in an unrecoverable fault state. Thus, all resource attributes that are related to automatic fault recovery on the resource level are

ignored. The resource is immediately declared faulted and the fault state of the resource is forwarded to the OSC service engine for failover processing. The following resource attributes are ignored:

- **RestartLimit**
- **ToleranceLimit**
- **OnlineRetryLimit**

Thus, the monitor routine code return OSC$_FAULTED reduces the time delay between the first occurrence of a fault condition and initiating failover processing.

If the monitor routine returns any OpenVMS error code the resource status is immediately set to ADMIN_WAIT, since an OpenVMS error code indicates that the monitor routine has failed to evaluate the state of the resource. Thus, from an OSC perspective the resource state is undefined..

The ADMIN_WAIT state is propagated up to the service and service group level and the affected service groups are removed from offline and failover processing. Fatal ADMIN_WAIT state change event messages are triggered for immediate system management intervention since OSC is now unable to control the affected resources, services and service groups.

5.6 How OSC Handles Resource Faults

This section describes the process OSC uses to determine the course of action when a resource fails.

5.6.1 OSC Behavior when an Online Resource faults

The OSC agent framework invokes the monitor action routine to determine the state of a resource. The exit codes of the monitor action routine signals the current state of a resource.

This section describes the actions performed by the OSC agent framework when it calls the monitor routine for a particular resource.

1. If the Monitor routine does not complete within the time defined by the resource attribute **MonitorTmo**, OSC checks the **FaultOnMonitorTmo** (FOMT) attribute of the resource.

- If **FaultOnMonitorTmo** =FALSE, monitor routine timeouts are not considered as fault conditions. OSC considers the resource online and continues to monitor the resource periodically, depending on the monitor interval.
- If **FaultOnMonitorTmo** = TRUE, OSC compares the **CurrentMonitorTmoCnt** (CMTC) with the **FaultOnMonitorTmoLimit** (FOMTL) value. If the monitor timeout count is less than the value of **FaultOnMonitorTmoLimit**, **CurrentMonitorTmoCnt** is incremented and OSC waits another monitor cycle for a response from the monitor routine.
- If **FaultOnMonitorTmoLimit** = **CurrentMonitorTmoCnt** the available monitor timeout count (credit) is exhausted and OSC must now take corrective action.
- If the resource is not an **On-Off** resource, OSC marks the resource as ONLINE|ADMIN_WAIT|TMO and triggers the ADMIN_WAIT state change event for immediate system management intervention.
- If the value of the **ManageFault** attribute is FALSE, OSC marks the resource as ONLINE|ADMIN_WAIT|TMO and triggers the ADMIN_WAIT state change event for immediate system management intervention.
  ManageFault = FALSE if either:
  - resource attribute **DisableManageFault** = TRUE
  - service group attribute **SrvGrpManageFault** = FALSE.
- Otherwise OSC invokes the clean routine with the reason code MONITOR HUNG.
- If the clean routine succeeds (exit code 0x1), OSC examines the value of the **RestartLimit** attribute. If the clean routine fails (any exit code except 0x1), the resource remains online with the state UNABLE TO OFFLINE. OSC fires a status change notification message and monitors the resource again. The resource enters a cycle of alternating monitor and clean routine calls until the clean action routine succeeds, the user intervenes or the **CleanRetryCount** (CRC) exceeds the value define by the **CleanRetryLimit** (CRL) attribute.
- If **CleanRetryCount** = **CleanRetryLimit**, OSC sets the ADMIN_WAIT bit in the status field of the resource and fires an ADMIN_WAIT state change event message for immediate system management intervention. The resource is removed from status monitoring until the ADMIN_WAIT state is cleared by system management.

2. If the monitor routine does not time out, the actions performed by the OSC agent depend on the return code of the monitor routine.

- If the monitor routine returns OSC$_ONLINE (exit code 1) the OSC agent takes no further actions. The **ToleranceCount** attribute is reset to 0 when the resource has remained online for the time period defined by the **ConfInterval** attribute.
- If the monitor routine returns OSC$_OFFLINE (exit code 9) the **ToleranceCnt** (TC) attribute is incremented. Afterwards the OSC agent checks whether the value of the **ToleranceCount** attribute is greater than the value defined by the **ToleranceLimit** (TL) attribute. If **ToleranceCount** > **ToleranceLimit** the OSC agent declares the resource as faulted.
- If the monitor routine returns OSC$_FAULTED (exit code 19) the OSC agent skips the **ToleranceCount** check.
- If the monitor routine returns neither OSC$_ONLINE, OSC$_OFFLINE or OSC$_FAULTED, the OSC agent sets the ADMIN_WAIT bit in the status field of the resource and fires an ADMIN_WAIT state change event message for immediate system management intervention. The resource is removed from status monitoring until the ADMIN_WAIT state is cleared by system management. No further action is taken.
- Once the OSC agent has declared a resource as faulted, OSC checks the **ManageFault** attribute. If **ManageFault**=FALSE, OSC marks the resource as OFFLINE|ADMIN_WAIT|UNEXP. OFFLINE and triggers ADMIN_WAIT notification for immediate system management intervention.
- If **ManageFault**=TRUE, and the resource is not an **On-Off** resource, OSC marks the resource as OFFLINE|FAULTED|UNEXP. OFFLINE and initiates service group fault processing (propagating the fault up the resource and service tree, service group offline and failover processing).
  ManageFault = TRUE if:
  - resource attribute **DisableManageFault** = FALSE and
  - service group attribute **SrvGrpManageFault** = TRUE.
- If the resource is an **On-Off** resource, OSC checks the **Freeze** attribute. If the resource is frozen, OSC marks the resource as OFFLINE|FAULTED|UNEXP. OFFLINE. No further action is taken – service group fault processing is not initiated on faults of frozen resources.
- If the resource is not frozen the clean routine is called.
- If the clean routine fails (any exit code except 0x1), the resource remains online with the state UNABLE TO OFFLINE. OSC fires a status change event and monitors the resource again. The resource enters a cycle of alternating monitor and clean action routine calls until the clean action routine succeeds, user

intervenes or the **CleanRetryCount** (CRC) exceeds the value define by the **CleanRetryLimit** (CRL) parameter.
If **CleanRetryCount** = **CleanRetryLimit**, OSC sets the ADMIN_WAIT bit in the status field of the resource and fires an ADMIN_WAIT state change event for immediate system management intervention. The resource is removed from status monitoring until the ADMIN_WAIT state is cleared by system management.

- If the clean routine succeeds (exit code 0x1), the next actions depend on the return status the OSC agent has received from the monitor routine:
    - OSC$_FAULTED
      OSC marks the resource as OFFLINE|FAULTED|UNEXP OFFLINE and initiates service group fault processing (propagating the fault up the resource and service tree, setting the service group offline and initiating failover processing).
    - OSC$_OFFLINE
      OSC tries to restart the resource depending on the values of the **RestartLimit** (RL) and **RestartCnt** (RC) attribute. If the value of the **RestartLimit** attribute is non-zero OSC increments the **RestartCnt** attribute and executes the online action routine.
    - If the online action routine succeeds to bring the resource online again OSC resumes periodic monitoring.
    - Otherwise OSC continues resource restart attempts until **RestartLimit** = **RestartCnt**.
    - If **RestartLimit** = **RestartCnt** OSC marks the resource as OFFLINE|FAULTED|UNEXP OFFLINE and initiates service group fault processing (propagating the fault up the resource and service tree, setting the service group offline and initiating failover processing).

5.6.1.1 OSC monitor flowchart for Online Resources

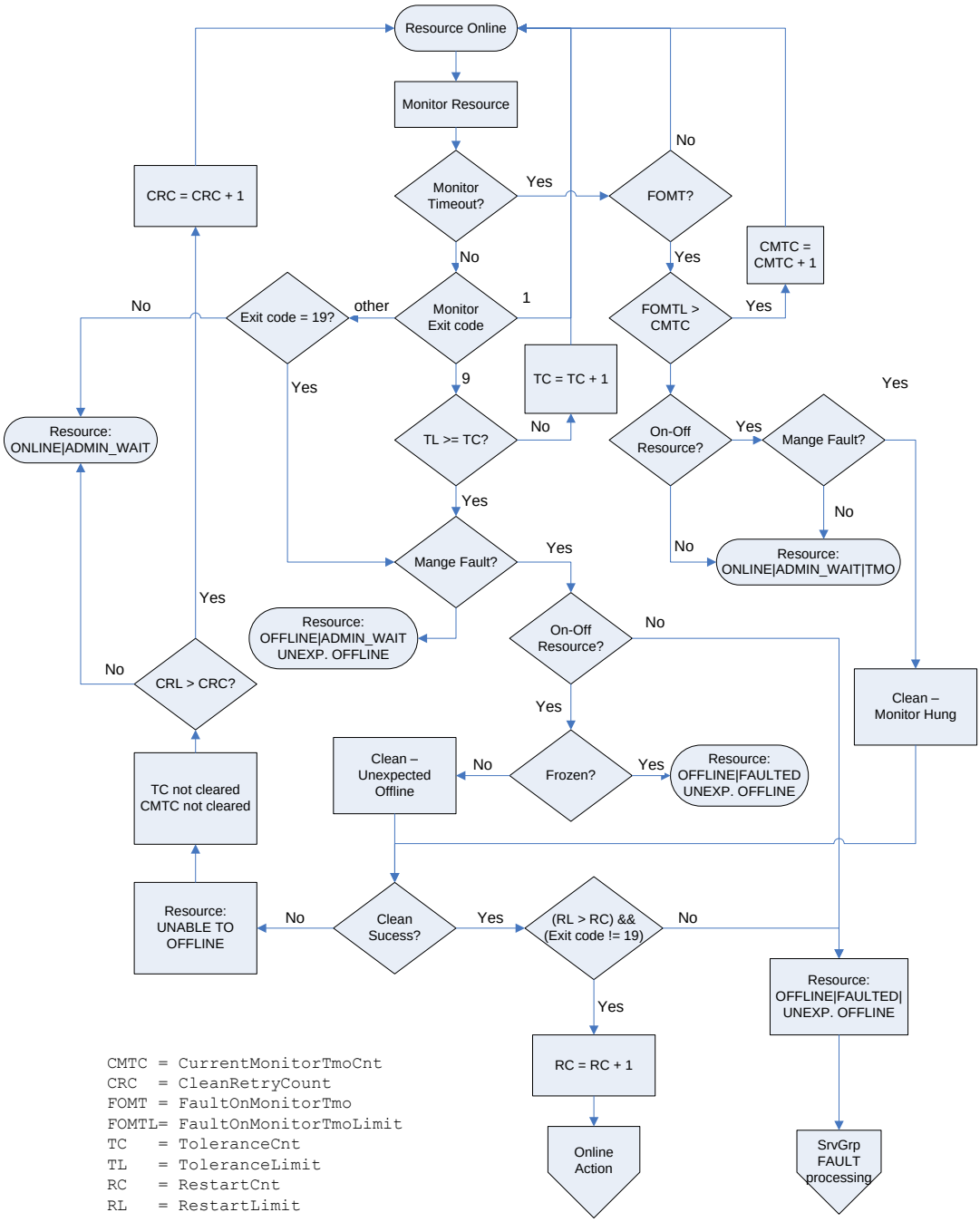


Fig. 5.1: Monitor decision flowchart of the OSC agent framework for online resources

5.6.2 OSC Behavior when a Resource fails to come Online

This section describes the actions performed by the OSC agent framework when the online action routine is called to bring an offline resource online.

- If the online action routine does not complete within the time defined by **OnlineTmo** attribute, OSC increments the **OnlineTmoWaitCnt** (OTWC) and checks if it is greater than the value of the **OnlineTmoWaitLimit** (OTWL). If **OnlineTmoWaitCnt** <= **OnlineTmoWaitLimit** the OSC agent fires a timeout notification event and continues waiting another online timeout period for the online action routine to complete. If **OnlineTmoWaitCnt** > **OnlineTmoWaitLimit** OSC accepts that the online routine has timed out.
- If the online action routine times out, OSC examines the value of the **ManageFault** attribute.
- If **ManageFault**=FALSE, OSC sets the ADMIN_WAIT bit in the status field of the resource and fires an ADMIN_WAIT state change event for immediate system management intervention. The resource is removed from status monitoring until the ADMIN_WAIT state is cleared by system management. No further action is taken.
- If **ManageFault**=TRUE, OSC calls the clean action routine with the clean reason set to ONLINE HUNG.
  ManageFault = TRUE if:
  - resource attribute **DisableManageFault** = FALSE and
  - service group attribute **SrvGrpManageFault** = TRUE.
- If the online action routine does not time out, OSC invokes the monitor routine. The exit code of the online action routine defines the time delay between online action routine completion and monitor routine execution.
- If the monitor routine returns OSC$_ONLINE (exit code 1) the resource is considered online and the OSC agent resumes periodic monitoring.
- If the monitor routine returns OSC$_FAULTED (exit code 19), OSC immediately declares the resource as faulted.
- If the monitor routine returns OSC$_OFFLINE (exit code 9), OSC examines the value of the **OnlineWaitLimit** (OWL) attribute. This attribute defines how many monitor cycles can return an offline status before the agent framework declares the resource as faulted. Each successive monitor cycle increments the **OnlineWaitCount** (OWC) attribute. When **OnlineWaitLimit** =

OnlineWaitCount (or if **OnlineWaitLimit** = 0), OSC considers the resource has faulted.

- After OSC has determined that the resource has failed, OSC examines the value of the **ManageFault** attribute. If the **ManageFault**=FALSE, the resource state changes to OFFLINE |ADMIN_WAIT.
- If **ManageFault**=TRUE and the resource is an **On-Only** resource, OSC marks the resource as OFFLINE|FAULTED and initiates service group fault processing (propagating the fault up the resource and service tree, service group offline and failover processing).
- If the resource is an **On-Off** resource, OSC calls the clean action routine with the clean reason set to ONLINE INEFFECTIVE.
- If the clean action routine fails (any exit code except 1), the resource remains online with the state UNABLE TO OFFLINE. OSC fires a status change event and monitors the resource again. The resource enters a cycle of alternating monitor and clean routine calls until the clean action routines succeeds, system management intervenes or the **CleanRetryCount** (CRC) exceeds the value define by the **CleanRetryLimit** (CRL) parameter.
  If **CleanRetryCount** = **CleanRetryLimit**, OSC sets the ADMIN_WAIT bit in the status field of the resource and fires an ADMIN_WAIT state change event for immediate system management intervention. The resource is removed from status monitoring until the ADMIN_WAIT state is cleared by system management.
- If the clean action routine succeeds to shutdown the resource (exit code 1), the OSC agent checks if the value of the **OnlineRetryLimit** (ORL) is non-zero. OSC increments the **OnlineRetryCount** (ORC) and invokes the online action routine again. This starts the cycle all over again. If **OnlineRetryLimit** = **OnlineRetryCount**, or if **OnlineRetryLimit** = 0, OSC assumes that the online operation has failed. OSC marks the resource as OFFLINE|FAULTED and initiates service group fault processing (propagating the fault up the resource and service tree, setting the service group offline and initiating failover processing).

5.6.2.1 OSC online decision flowchart

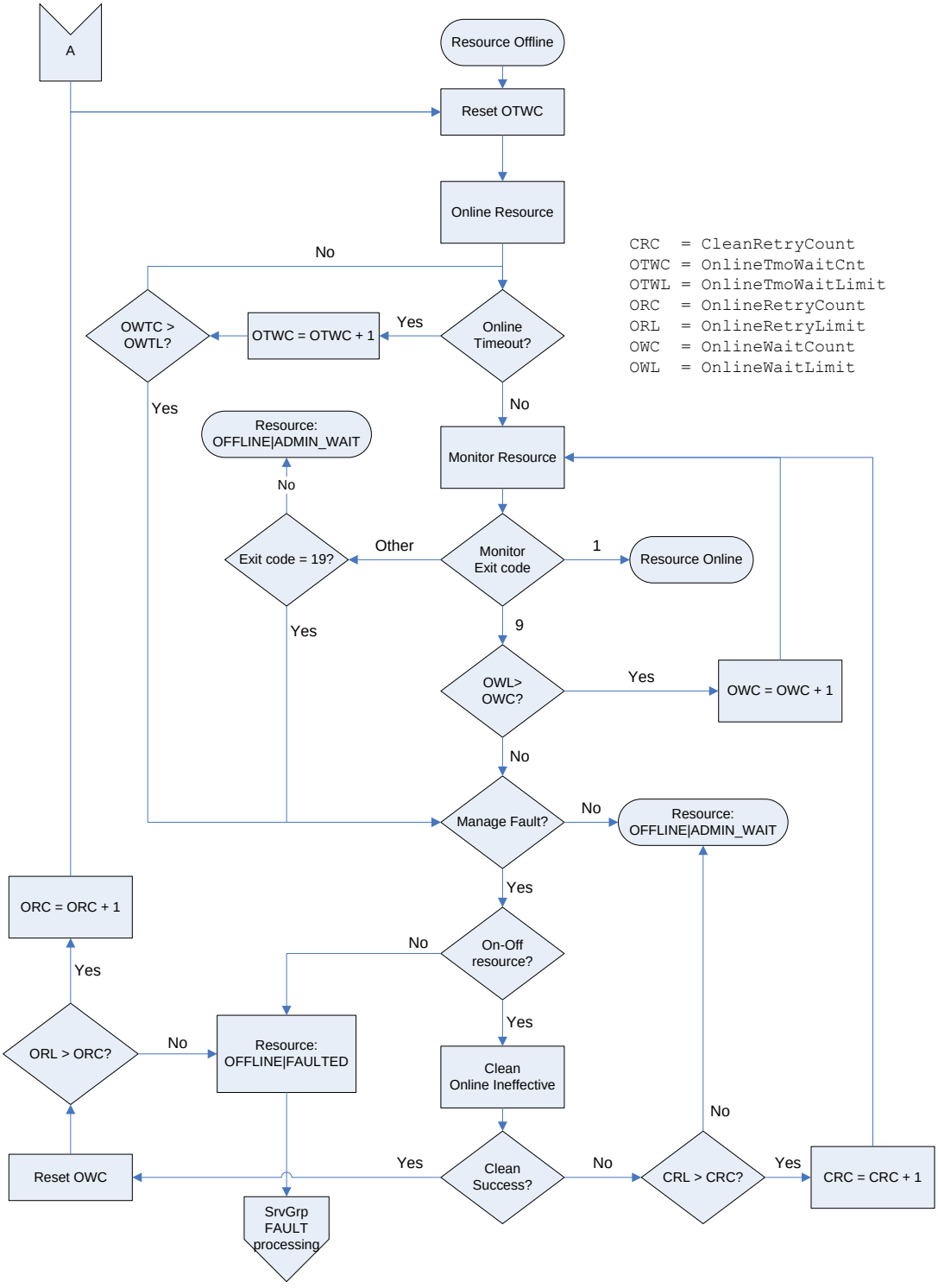


Fig. 5.2: Online decision flowchart of the OSC agent framework.

5.6.3 OSC Behavior after a Resource is declared faulted

After a resource is declared faulted, OSC triggers an error status event notification message and executes the tasks listed below:

- OSC examines if the faulted resource is critical. If the resource is not critical, the PARTIAL bit is set in the status field of the related services and service groups. No further actions are taken.
- If the resource is critical OSC examines the value of the service group attribute **SrvGrpFaultProp** of all service groups the resource is a member of.
- If **SrvGrpFaultProp**=FALSE, OSC does not propagate the fault condition up the resource tree to the service and service group level. The service and service group status does not change and thus, the service group does not failover.
  If **SrvGrpFaultProp**=TRUE, OSC propagates the fault condition up the resource tree to the service and service group level.
- OSC sets the FAULT bit in the status field of all services the resource is a member of and propagates the fault condition up to the service group level.
- OSC examines the values of the **ServicePriority** attribute of the top level services of all independent service trees configured within a service group.
- If OSC detects that the service group contains an online service tree with a service priority higher then the failed service tree, the service group is not failed over to another node – only the fault bit is set in the status field of the service group to indicate that some services of the service group have failed. No further actions are taken.
- If the service group contains no other service tree other than the failed one or the service priorities of the remaining online service trees are equal or less than the service priority of the failed service, OSC attempts to shutdown (offline) all service trees configured in the service group.
- If **SrvGrpFailover** = FALSE, the service group is not started on another service group execution node.
- If **SrvGrpFailover** = TRUE, OSC initiates failover processing for the service group according to the OSC failover policy configured (for detailed information about OSC failover policies please refer to section 5.2 Controlling OSC Failover Policy).
- If no valid failover target exists OSC fires a fatal control event message for immediate system management intervention.

5.7 Clearing OSC states

The FAULT and ADMIN_WAIT bits in the resource, service or service group status fields are not automatically cleared by OSC, since these flags indicate problems with the respective resource, service or service group. As long as either the FAULT or the ADMIN_WAIT bit is set for a resource, service or service group on a particular node this resource, service or service group is removed from the list of entities being managed by OSC. These states have to be manually cleared to signal OSC that the problems that caused OSC to set these flags have been resolved (=the resource, service or service group have returned to normal operational state) and that OSC can take over control of these entities again.

5.7.1 Clearing FAULT states

1. If the FAULT bit is set in the resource, service or service group status field, the respective resource service or service group has been offlined by forcible means and has been removed from being managed by OSC.
   The problem that caused a resource to fail outside the scope of OSC control must be resolved first.
2. Clear the FAULT bit
   The FAULT bit on a service group, service or on the resource level can be cleared by issuing the appropriate OSC$MGR commands listed below:

   OSC$MGR> CLEAR SRVGRP *servicegroup-name*/FAULTED[/NODE=*nodename*]
   OSC$MGR> CLEAR SERVICE *service-name*/FAULTED[/NODE=*nodename*]
   OSC$MGR> CLEAR RESOURCE *resource-name*/FAULTED[/NODE=*nodename*]

   If you clear the FAULT bit on a service group level, OSC automatically clears the FAULT bit in the status field of all its associated services and resources.

   If you clear the FAULT bit on a service level, OSC automatically clears the FAULT bit in the status field of all its associated resources.

   OSC immediately calls the Monitor routine for a resource after the FAULT bit has been cleared, to validate the operational state of that resource.

For detailed information about the OSC$MGR command syntax to clear faults please refer to the online help of the OSC$MGR utility.

5.7.2 Clearing ADMIN_WAIT states

1. OSC sets the ADMIN_WAIT bit in the resource, service or service group status field, if OSC encountered problems that OSC was not able to handle.
   These problems have to be resolved outside of the scope of OSC control.
2. Clear the ADMIN_WAIT bit
   The ADMIN_WAIT bit can be cleared at the service group, service, or on the resource level by issuing the appropriate OSC$MGR commands listed below:

   OSC$MGR> CLEAR SRVGRP *servicegroup-name*/ADMIN_WAIT
   [/NODE=*nodename*]
   OSC$MGR> CLEAR SERVICE *service-name*/ADMIN_WAIT
   [/NODE=*nodename*]
   OSC$MGR> CLEAR RESOURCE *resource-name*/ADMIN_WAIT
   [/NODE=*nodename*]

   If the ADMIN_WAIT bit is cleared on a service group level, OSC automatically clears the ADMIN_WAIT bit in the status field of all its associated services and resources.

   If the ADMIN_WAIT bit is cleared on a service level, OSC automatically clears the ADMIN_WAIT bit in the status field of all its associated resources.

   OSC immediately calls the monitor routine for a resource after the ADMIN_WAIT bit has been cleared to validate the operational state of that resource. If OSC re-encounters problems that cannot be handled by OSC the ADMIN_WAIT bit is reset.

   For detailed information about the OSC$MGR command syntax to clear ADMIN_WAIT states please refer to the online help of the OSC$MGR utility.

5.8 OSC adaptive startup behavior

OSC provides adaptive startup behavior. An execution node list has to be defined for each service group configured (see section5.3.1 Controlling Service Group Execution Nodes). The execution node list defines the OSC cluster members the service group is allowed to be started on and it defines the execution order of the OSC nodes. Adaptive startup means that OSC does not forcibly bring a service group online on the primary execution node if the service group is already online on any other node of the service group's execution node list when OSC is started cluster-wide.

When OSC is started cluster-wide (see section 7 Managing OSC) the active OSC master control process initiates OSC state transition. During the state transition the active OSC master control process checks if all OSC components are started (OSC service engines, OSC agents) and starts them if required. In addition, the OSC master control process requests the current state of all managed OSC items (resources, services and service groups) from the OSC agents and OSC service engines cluster-wide in order to guarantee that the OSC master control status information of all OSC items is accurate before it starts processing. The duration of the OSC state transition is defined by the OSC master control attribute **OscCtrlStartupWait** (see section 7 Managing OSC and A.6 OSC master control and service engine attributes of the appendix).

After the state transition timer **OscCtrlStartupWai** has expired the active OSC master control process checks the current state of all managed service groups.

If all the required instances of a service group are online on any OSC cluster members it is configured to run on, OSC does not initiate any management activities, even if the service group is not online on the primary execution node. The primary execution nodes are the preferred nodes the service group is configured to run on (see section 5.3.1 Controlling Service Group Execution Nodes).

If OSC detects that too few instances of a service group are online OSC starts the missing instances of the service group according to the priority of the service group's execution node list (see section 5.3.1 Controlling Service Group Execution Nodes).

If a service group is partially online on an OSC cluster node implying that not all of its resources are online, OSC automatically tries to start the offline resources on that OSC node.

If OSC detects during startup that too many instances of a service group are online within the OSC cluster, OSC sets the service group state to

ADMIN_WAIT on all the service group execution nodes even though the value of the service group attribute **SrvGrpCleanOnUnexpOn** is TRUE (when OSC starts up it is not evident to OSC which instances have to be to shutdown - see also section5.3.8 Controlling Behavior on external Resource startup) and triggers the fatal ADMIN_WAIT state change event message for immediate system management intervention.

6

OSC event notification

OSC provides event notification messages when:

- A resource, service or service group state changes
- The OSC cluster state changes
    - The operational state of any OSC component changes (OSC agents, OSC service engine, OSC master control process)
    - Any of the OSC cluster member state changes (OSC cluster member leaves or joins the OSC cluster)
- OSC executes automatic or user initiated administrative operations to:
    - Recover resource fault conditions
    - Online/offline/failover/switchover service groups
    - Re-establish OSC cluster integrity (restart of nonoperational OSC components, removing a member from the OSC cluster if a node fails …)

Nine OSC event classes exist:

- HEARTBEAT
  Heartbeat events are triggered whenever an OSC component receives a heartbeat message from one of its managed OSC components (see section 4.3 OSC software architecture).
- OSCAGT_CONTROL_EVT
  This OSC event class contains resource control messages. Resource control messages are triggered by OSC agents when they execute administrative operations (such as online, offline commands) either on request by the OSC service engine or automatically triggered to recover a resource fault condition.
- OSCAGT_STATE_EVT
  This OSC event class contains resource state change messages. Resource state change messages are sent by an OSC agent when it detects a resource status change.
- OSCSRV_CNXMAN_EVT
  This OSC event class contains all OSC connection management messages from the OSC service engine. The OSC service engine triggers connection management messages when:
    - It detects that the state of an OSC agent has changed from operational to nonoperational

    - It (re)starts OSC agents
    - It has lost/established connection to the OSC master control process
- OSCSRV_CONTROL_EVT
  This OSC event class contains all control messages from the OSC service engine. A control message is triggered when the OSC service engine executes administrative operations on service groups and services on request of the OSC master control process.
- OSCSRV_STATE_EVT
  This OSC event class contains all state change messages from the OSC service engine. A state change event is triggered if the state of one of the services and service groups managed by the OSC service engine changed its state.
- OSCCTRL_CNXMAN_EVT
  This OSC event class contains all OSC connection management messages from the OSC master control process. The OSC master control triggers connection management messages when:
    - The OSC cluster state changes. These occur when:
        - OSC cluster member leaves/joins the OSC cluster
        - The OSC master control process detects that the state of a managed OSC service engines changes from operational to non-operational
    - The OSC master control process (re)starts the OSC service engines
    - The OSC master control process initiates an OSC cluster state transition
    - The OSC master control process modifies the OSC quorum due to the OSC cluster reforming.
- OSCCTRL_CONTROL_EVT
  This OSC event class contains all control messages from the OSC master control process. The OSC master control process triggers control messages when it executes administrative operations on a service group (setting a service group: online, offline, switchover, failover, freezing a service group, disabling a service group …) either due to a service group fault condition or upon user request.
- OSCCTRL_STATE_EVT
  This OSC event class contains all state change messages of the OSC master control process. A state change message is triggered when the state of a service group on any OSC cluster member changes.

Possible OSC event message severity levels:

- Informational
  Informational messages indicate that the state of the managed items has not changed, or that a managed item changed state as expected. For example:
  - An informational event message is sent when a user connects to the console of the OSC master control process via the OSC$MGR utility.
  - An informational event message is sent if a user requests to set a service group offline and the service group state changed to offline as expected.
- Warning
  Warning messages are triggered when OSC detects a fault condition that does not cause OSC intervention. Typically warning messages are sent if a non-critical resource fails or OSC rejects a user command due to state or authorization conflicts. For example, the user issues a command to set a service group online and the service group is already online, or the user is not authorized to manage a particular service group.
- Error
  Error messages are triggered when OSC detects any fault condition that causes OSC to intervene automatically and OSC is able to recover from the fault condition. Error messages are typically triggered when a resource, service and service group is declared faulted, a node is removed from the OSC cluster, or OSC detects that one of its components (OSC agent, OSC service engine, OSC master control process) is non-operative and this component has had to be restarted by OSC.
- Fatal
  Fatal messages are sent when OSC detects a fault condition that cannot be managed by OSC and requires immediate system management intervention. For example, if a resource, service or service group state is set to ADMIN_WAIT a fatal event message is sent. A managed item is set to the ADMIN_WAIT state when OSC cannot recover the fault condition. For detailed information about OSC fault handling please refer to the section 5 Controlling OSC behavior and 5.6 How OSC Handles Resource Faults.

The OSC event notification service is able to:

- log events into the common OSC event message file
- provide OPCOM messages for OSC events
- forward events to all connected OSC consoles (for detailed information about OSC event monitoring consoles please refer to section 7.6)

- execute a user script for each OSC event for site specific event handling

When a user starts the OSC$MGR utility it connects automatically to one of the 64 OSC consoles managed by the active OSC master control process. For more information about OSC consoles please refer to section 7 Managing OSC.

6.1 Configuring OSC events

Based on the severity level the user can configure which events of an OSC event class are:

- logged into the common OSC event message file
- sent as an OPCOM message
- forwarded to the connected OSC consoles
- processed by a user script

The OSC event class definition is stored in the OSC configuration database. To modify the OSC event class definitions start the OSC$CFG utility, open the working OSC configuration database you wish to modify and issue the command:

OSC$CFG> MODIFY EVENT *event-class*

The *event-class* parameter is mandatory and defines the event class to be modified. Enter one of the event class keywords listed in the previous section (HEARTBEAT, OSCAGT_STATE_EVT, OSCAGT_CONTROL_EVT ...). This command starts the OSC event configuration wizard. It prompts you to enter the base severity level for each event notification method available for the event class. A particular event method is triggered only if the severity of the event is equal or greater than the base severity of the event method.

Severity level values:

- 1 Informational
- 2 Warning
- 3 Error
- 4 Fatal

Example:

In this example the OSCCTRL_STATE_EVT event class definition stored in the working OSC configuration database with version V1.0 is modified:

```
OSC$CFG> OPEN DATABASE / VERSION=1.0
OSC$CFG-I-SUCCESS, CFG database
'OSC$COMMON:[CFG]OSC$CONFIG_514.DAT;1' is the new working
area
OSC$CFG> MODIFY EVENT OSCCTRL_STATE_EVT


Welcome to the EVENT OSC configuration wizard
-------------------------------------------

Dscr: Event Log-File severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtLogPriority} [1]:
Dscr: Event OPCOM-Msg severity (0=none,1=Inf,2=Warn,3=Err,4=Fat)
Attr: {EvtOpcomPriority} [0]:
Dscr: Event forward to consoles (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtMgrFwd} [1]:
Dscr: OSC console severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtUserScriptPriority} [0]: 4
Dscr: Event User-Script filename
Attr: {EvtUserScript} []: OSC$CFG:OSC_SENDMAIL.COM
OSC$CFG-I-MODIFY, item 'OSCCTRL_STATE_EVT' has been updated
in the current working CFG database
```

In this example all events of the OSCCTRL_STATE_EVT event class are logged into the common event message file and forwarded to the connected consoles (attributes: **EvtLogPriority**, **EvtMgrFwd**). No OPCOM event message is sent (attribute **EvtOpcomPriority**). Fatal event messages of this event class causes the OSC event notification service to execute the user script OSC$CFG:OSC_SENDMAIL.COM.

6.2 OSC event message file

The OSC event notification service creates a daily common event message file. The common event message file is a text file that contains the events of all event classes sent by any OSC component within the OSC cluster ordered by their occurrence. The file name format of the OSC event file is:

- OSC$EVENT.LOG_*date*

The common event message files are located in the OSC$LOG directory. Common event messages files are never deleted by any OSC component. It is the system manager's task to clean up old event message files.

The format of the event message file can be configured using the OSC$CFG MODIFY EVTFMT command. For detailed information about the event message format configuration options please refer to the OSC$CFG online help.

The most recent OSC event file is always locked by the OSC event notification service. Thus, this OSC event file cannot be opened with any editor until the OSC event file is closed at midnight or after OSC has been shutdown.

To display the events of the most recent OSC event file or to extract the events into a new file execute the OSC$MGR command:

```
OSC$MGR> EXTRACT EVENTS [/SINCE][/BEFORE][/OUTPUT]
```

This command forces the OSC event notification service to flush its event buffers into the OSC event file (or another file) and extracts the events according to the qualifiers applied.

For a detailed description of the command qualifiers please refer to the OSC$MGR online help.

6.3 User event script

The OSC event notification service passes eight parameters to a DCL user event script:

- P1 Name of the OSC item
  This parameter contains the name of an OSC resource, service or service group.
  Please note that the OSC event classes listed below provide a zero length string in P1:
  - HEARTBEAT
  - OSCSRV_CLUSTER_EVT
  - OSCCTRL_CLUSTER_EVT
- P2 Severity keyword
  The available severity keyword strings passed in P2 are:
  - INFO
  - WARNING
  - ERROR
  - FATAL
- P3 OSC cluster node name that hosts the OSC process that triggered the event

- P4 OSC process that triggered the event (OSC$CTRL, OSC$SRV, OSC agent process name)
- P5 OSC event message type
  Available keywords passed in P5:
  - %CNXMN OSC connection manager message
  - %CONTROL OSC control message
  - %STATE OSC state change message
  - %OSCHBT OSC heartbeat message
- P6 Event message text
- P7 Activation number
  This is a unique process identification number (but not the process ID) of the process that is executing the DCL script.
- P8 OSC entity type
  This parameter contains the OSC entitytype in a state change event. Possible keywords passed in P8:
  - RESOURCE (resource state change)
  - SERVICE (service state change)
  - SRVGRP (service group state change)

  Please note that only the OSC state change event classes listed below pass either of these keywords to DCL user event scripts:
  - OSCAGT_STATE_EVT
  - OSCSRV_STATE_EVT
  - OSCCTRL_STATE_EVT

  All other event classes provide a zero length string in P8.

6.3.1 Example

The installation procedure of VSI OpenVMS ServiceControl provides an example of a user defined event script. OSC$CFG:OSC_SENDMAIL.COM sends a mail if the severity of the event is either ERROR or FATAL.

```
$! VSI OpenVMS ServiceControl Mail notification
$!
$! This is an example for an user notification script triggered by
$! VSI OpenVMS ServiceControl.
$!
$! This script checks the severity state of the message. If the severity
$! is:
$!     o      FATAL
$!     o      ERROR
$!
$! a mail is sent.
$!
$! Input parameter:
$!     P1 ...  virtual resource, service, service group name
```

```
$!     P2 ...  Severity level - possible values:
$!          o  FATAL   -       fatal events
$!                             = failure conditions that cannot be handled
$!                             by OSC - System management intervention
$!                             is required immediately
$!          o  ERROR   -       error message
$!                             = failure conditions that can be handled by OSC
$!                             System management intervention is not required
$!          o  WARNING -    warning event
$!                             = abnormal conditions that do not cause OSC
$!                             intervention
$!          o  INFO    -       info event
$!     P3 ...  Node name the message was created
$!     P4 ...  Process running on node P3 that created the message
$!     P5 ...  OSC message type:
$!          o      %STATE   -       state change of managed SrvGrps,
$!                                  Services and Resources
$!          o      %CONTROL -    OSC operation control messages
$!          o      %CNXMAN  -   OSC cluster connection manager messages
$!          o      %OSCHBT  -     OSC process heartbeat messages
$!     P6 ...  event message text
$!     P7 ...  Activation Number - identifies the sub-proccess that processed this script
$!
$ SET NOON
$!
$! IF ((P2 .NES. "ERROR") .AND. (P2 .NES. "FATAL")) THEN EXIT (1)
$!
$ DEFINE SYS$SCRATCH OSC$SCRATCH
$!
FILE_TO_OPEN = "OSC$LOG:OSCEVT_''P4'_''P7'.''P3'"
$!
$ OPEN/WRITE OUTFILE 'FILE_TO_OPEN'
$ WRITE OUTFILE "''F$TIME()': OSC ''P2' ''P5' event notification from ''P4'@''P3'"
$ WRITE OUTFILE ""
$ WRITE OUTFILE "''P6'"
$ CLOSE OUTFILE
$!



Help

$ MAIL/NOSELF/SUB="OSC ''P2' ''P5' event notification from ''P4'@''P3'" -
                        'FILE_TO_OPEN' SMTP%"support@vmssoftware.com"
$!
$ DELETE/NOCONF/NOLOG 'FILE_TO_OPEN';*
$!
$ EXIT
```

Please note at present P1 contains the name of the affected entity, which may be the name of a resource, a service or a service group. It is quite easy to determine, whether the name of the entity describes a resource as the resource name will always contain the resource type followed by the

"::" characters. However if the same name has been given to a service as well as the associated service group it will be impossible to differentiate between these two entity types. This is a point to consider during your configuration phase, if you plan to implement action routines dependent on the entity type

7

Managing OSC

This section provides a brief overview of how to manage VSI OpenVMS ServiceControl. OSC is managed using the OSC$MGR DCL command line utility or the graphical user interface OscMgrGUI available for Windows 2003, XP, V7 and V8. All management commands can be either applied using OSC$MGR or OscMgrGUI.

Using the graphical user interface to manage OSC will hopefully be self-explanatory. The aim of this section is to explain how to manage OSC using the OSC$MGR DCL command line utility.

To start the OSC$MGR DCL command line utility run the image:

- OSC$BIN:OSC$MGR.EXE

Once the command line utility is started it tries to establish an IP connection to the console port of the OSC master control process (OSC$CTRL). The OSC master control process can manage up to 64 console connections in parallel.

For a detailed description of the OSC$MGR command syntax please refer to the online help of OSC$MGR.

7.1 How to start OSC

VSI OpenVMS ServiceControl requires a valid default configuration database to run. For more detailed information about how to create a default configuration database please refer to section 8 OSC configuration guidelines and to the online help of the configuration utility OS$CFG.

The startup command script

- @SYS$STARTUP:OSC$STARTUP.COM

starts VSI OpenVMS ServiceControl cluster-wide if OSC is not running on any OSC cluster member. Otherwise the command script starts OSC on the node the command was issued on, to join the OSC cluster.

Alternatively the user can start OSC cluster-wide using the OSC$MGR command

- OSC$MGR> STARTUP/CLUSTER [/MODE=SIMULATION]

If you apply the /MODE=SIMULATION qualifier or the OSC master control attribute **OscCtrlSimulate** is set to TRUE (see A.6 OSC master control and service engine attributes), OSC is started in simulation mode cluster-wide. Starting the OSC environment in simulation mode is useful for:

- Training of how to manage OSC
- Testing a new OSC configuration
- Learning how OSC behaves in particular fault scenarios

without affecting the applications (service groups) already running on the OpenVMS cluster.

Please refer to section 11 OSC simulation for a more detailed description of the OSC simulation mode.

To start OSC on a particular node the user can also apply the OSC$MGR command:

- OSC$MGR> STARTUP/NODE=*node-name*

7.2 How to manage an OSC cluster

7.2.1 Required privileges to manage the OSC cluster

In order to execute the commands listed above, a user has to logged on to the system with the required privileges (see also section 4.4 OSC user privileges) or the account the user is logged into has been granted the following identifier:

- OSC$MANAGE_CLUSTER

Unprivileged users are only allowed to view the status of the OSC cluster.

7.2.2 Importance of OSC votes

An OSC cluster consists of all or a subset of the OpenVMS cluster members VSI OpenVMS ServiceControl is installed on. The OSC cluster members are defined when you configure the OSC cluster (**OscCtrlNode** attribute – see section A.6 OSC master control and service engine attributes). You have to assign votes to each OSC cluster member.

The OSC master control (OSC$CTRL) process is the software component that decides whether or not to offline or failover a service group in response to a resource failure. The OSC master control process is started on all OSC cluster members, but is only active on one OSC node at any one time. All other OSC master control processes are on standby waiting to take over activities if the active OSC master control process fails. The number of votes assigned to the individual OSC cluster member nodes determines the node the OSC master control process initially becomes active on, when VSI OpenVMS ServiceControl is started:

- The OSC master control process on the OSC cluster member node with the highest number of OSC votes assigned becomes the active OSC master.
- If all OSC cluster members have been assigned the same number of votes, the OSC cluster member node where the VSI OpenVMS ServiceControl start command script is executed on, becomes the active OSC master.

In addition OSC quorum is calculated based on the number of OSC votes assigned to each OSC cluster member. The algorithm used to calculate OSC quorum is identical to how OpenVMS calculates cluster quorum. For detailed information about how OpenVMS calculates cluster quorum please refer to the OpenVMS manual *OpenVMS cluster systems*.

If an OSC cluster member unexpectedly leaves the OpenVMS cluster (crash, SCS link lost), OSC checks whether the remaining OSC cluster members have OSC quorum. This means that OSC checks if the sum of votes of all remaining OSC cluster members is greater or equal to the value of OSC quorum.

If the OSC cluster has lost OSC quorum (sum of OSC member votes is less than OSC quorum) OSC continues monitoring all resources, services and service groups on the remaining OSC cluster members, but OSC stops performing automatic service group activities (taking service group offline, failing over service groups) in response to resource fault conditions. In addition OSC blocks any interactive service group, service and resource management commands except the SHOW commands.

When OSC is started, the OSC master control attribute **OscCtrlExpVotes** defines the initial OSC quorum in the same way the EXPECTED_VOTES SYSGEN parameter defines the initial OpenVMS cluster quorum.

The importance of the OSC quorum is to prevent starting all service groups on a subset of the OSC cluster members if the overall system

resources of the remaining OSC cluster members is not sufficient to run all service groups on these OSC nodes.

To regain OSC quorum either additional OSC cluster members have to join the OSC cluster, or the user has to manually adjust OSC quorum (see section 7.2.5 How to adjust OSC quorum).

However this default behavior can be overruled. If the OSC master control attribute **OscCtrlAutoAdjustQuorum** is set to TRUE (see also A.6 OSC master control and service engine attributes) OSC automatically adjusts OSC quorum whenever an OSC cluster member unexpectedly leaves the OpenVMS cluster (crash, SCS link lost). VSI recommends that this feature is only enabled if the OpenVMS cluster that OSC is installed on, contains a quorum disk instead of a quorum system. A quorum disk cannot be configured in OSC. Thus, in order to avoid that OSC blocks, even though the OpenVMS cluster is still valid, automatic quorum adjustment should be enabled.

7.2.3 OSC reconnection interval

When the OSC master control process detects that an OSC cluster member has left the OSC cluster (crash, SCS link lost) it waits the time of the OSC reconnection interval to see whether or not the OSC node rejoins the OSC cluster. After the OSC reconnection interval has expired and if the OSC node has not rejoined the OSC cluster the OSC node is declared as faulted and the active OSC master control process initiates service group failover processing. This mechanism is similar to the OpenVMS cluster reconnection interval.

The OSC reconnection interval is defined by the OSC master control attribute **OscCtrlReconnWait** (see section A.6 OSC master control and service engine attributes of the appendix).

7.2.4 OSC state transition

The OSC state transition phase is initiated whenever:

- OSC cluster is started
- An OSC cluster member joins the OSC cluster
- An OSC cluster member is removed from the OSC cluster
- The active OSC master control process moves from one OSC cluster member to another

During the state transition phase the OSC master control process requests the current state of all managed resources, services and service groups from all OSC agents and OSC service engines on the available OSC cluster members. This is done to ensure that the status information of the managed items maintained by the active OSC master control process is up to date before it starts service group processing. This mechanism avoids wrong service group offline/failover decisions based on incomplete and/or out of date status information.

The duration of the OSC state transition is defined by the OSC master control attribute **OscCtrlStartupWait** (see section A.6 OSC master control and service engine attributes of the appendix).

7.2.5 How to adjust OSC quorum

Unless the OSC master control attribute **OscCtrlAutoAdjustQuorum** (see also A.6 OSC master control and service engine attributes) has not been set to TRUE the user has to manually adjust OSC quorum if OSC quorum is lost due to a node crash and the user wants OSC to continue (see also 7.2.2 Importance of OSC votes).

To adjust OSC quorum start the OSC$MGR utility and execute the command:

```
OSC$MGR> ADJUST QUORUM
```

In order to execute this command you have to be logged in with the required privileges (see also section 4.4 OSC user privileges) or the account you are logged into has been granted the following identifier

- OSC$MANAGE_CLUSTER

7.2.6 How to display OSC cluster status

In order to display the current state of the OSC cluster execute the command:

```
OSC$MGR> SHOW CLUSTER
```

This command displays the status of all configured OSC cluster members and the status of the respective OSC master control processes on all the OSC cluster members.

For more detailed information about possible OSC cluster members and OSC master control process states please refer to the OSC$MGR online help or refer to section A.1 OSC cluster and system states).

7.2.7 How to switch OSC master control

The OSC master control process can be manually switched from one OSC node to another, regardless of the votes assigned to the target OSC cluster member.

To switch OSC master control start the OSC$MGR utility, and issue either of the commands listed below:

```
OSC$MGR> SWITCH CONTROL [/TARGET_NODE=nodename]
OSC$MGR> MOVE CONTROL [/TARGET_NODE=nodename]
```

If you omit the /TARGET_NODE qualifier the OSC master control process is switched to the highest rated OSC master control standby node. The OSC node rating is based on the votes assigned to the individual OSC cluster members. The higher the number of votes the higher the rating. If some or all OSC cluster members have the same number of votes assigned, they will be rated by the order they are listed in the OSC cluster definition section of the configuration database in use.

7.2.8 How to lock an OSC node

An OSC cluster member is called “user locked” if it has been removed from the expected active OSC node member set of the OSC cluster during run-time. Typically an OSC cluster member will be “user locked” if the OSC node has been shutdown or the OSC node has left the OpenVMS cluster unexpectedly (crash, SCS link lost) and this OSC node will not join the OSC cluster for a longer period of time (several days).

An OSC node can be “user locked”only if it has been previously shutdown or if it has previously left the OpenVMS cluster unexpectedly (OSC node state: BROKEN).

In order to lock an OSC cluster member execute the OSC$MGR command:

```
OSC$MGR> LOCK/NODE=nodename
```

The /NODE qualifier is mandatory. It defines the node to be “user locked”.

With the LOCK/NODE command the user can, as described above, remove an OSC node from the expected active OSC node member set of the OSC cluster, without removing the OSC node from the OSC cluster configuration. A "user locked" OSC node remains a member of the potential OSC node member set. A "user locked" OSC node will be marked as "USR_LCK" in the output of the OSC$MGR SHOW CLUSTER command.

Once an OSC cluster member is "user locked" the OSC environment prevents OSC from starting up on that OSC node (none of the OSC$MGR STARTUP commands will start OSC on a "user locked" node). The OSC node has to be unlocked before OSC can be started on that node again.

The LOCK/NODE command restarts OSC automatically on the remaining OSC nodes of the expected active OSC cluster member set.

7.2.9 How to unlock an OSC node

In order to unlock an OSC cluster member execute the OSC$MGR command:

```
OSC$MGR> UNLOCK/NODE=nodename
```

The /NODE qualifier is mandatory. It defines the node to be "user unlocked".

The UNLOCK/NODE command adds a previously "user locked" OSC node to the expected active OSC node member set of the OSC cluster.

The UNLOCK/NODE command restarts OSC automatically on all OSC nodes of the expected active OSC node member.

7.2.10 How to shutdown OSC

OSC can be shutdown on either a dedicated OSC cluster member or cluster-wide

7.2.10.1 OSC cluster member shutdown

To shutdown OSC on a dedicated OSC cluster member execute the OSC$MGR command:

```
OSC$MGR> SHUTDOWN/NODE=node_name/FORCE=keyword
```

The /NODE qualifier defines the OSC cluster member (SCS node name) to shutdown OSC.

The /FORCE qualifier is mandatory for shutting down OSC on a dedicated OSC cluster member. It defines how online service groups will be handled before the OSC components are stopped on the OSC cluster member. Valid keywords are:

- MIGRATE
  All online service groups are switched over to other OSC cluster members before OSC is shutdown.
- OFFLINE
  All online service groups are taken offline before OSC is shutdown. The online service groups are not switched over to another OSC node.

Whenever an OSC node is shutdown manually OSC quorum is automatically adjusted. Thus, OSC will never block service group management activities due to an OSC quorum loss.

7.2.10.2 OSC cluster shutdown

To shutdown VSI OpenVMS ServiceControl cluster-wide execute the OSC$MGR command:

```
OSC$MGR> SHUTDOWN/CLUSTER [/FORCE=OFFLINE]
```

If you omit the /FORCE qualifier all OSC components will be stopped on all OSC cluster members without preprocessing any managed items (resources, services and service groups). Thus, if you omit the /FORCE qualifier the applications represented by the service groups remain available.

If you apply the /FORCE=OFFLINE qualifier all the applications represented by the online service groups will be shutdown on all OSC cluster members before OSC is stopped. No other keyword can be assigned to the /FORCE qualifier.

7.2.11 How to manage OSC console connections

As described in the introduction of this section OSC is managed using the OSC$MGR DCL command line utility or the graphical user interface OscMgrGUI available for Windows 2003, XP, V7 and V8. Both management utilities establish IP connections to the console port of the OSC master control process (OSC$CTRL). The OSC master control process can manage up to 64 console connections in parallel.

To display the current console connections execute the OSC$MGR command:

```
OSC$MGR> SHOW CONSOLE
```

This command lists all console connections with their connection attributes:

- Console ID
- TCP socket (BG) device of the console connection allocated by the OSC$CTRL process
- Console connection type (DCL/GUI)
- State of the console connection
- Flag that indicates whether or not a particular console connection is the console connection of the OSC$MGR session
- User login name
- Process ID of the console client (management utility)
- Name or IP address of the node where the console client (management utility) associated with the console connection is running.
- Login time

For a detailed description of the SHOW CONSOLE command please refer to the OSC$MGR online help.

Privileged users (see section 4.4 OSC user privileges) or users who have the OSC$MANAGE_CLUSTER identifier assigned can disconnect console connections:

```
OSC$MGR> DISCONNECT CONSOLE/ID=console-ID
```

For a detailed description of the DISCONNECT CONSOLE command please refer to the OSC$MGR online help.

7.3 How to manage OSC service groups

7.3.1 Required privileges to manage the OSC Service Groups

In order to execute OSC service group management commands you have to be logged in with the required privileges (see also section 4.4 OSC user privileges) or the user has at least one of the following identifiers assigned:

- OSC$MANAGE_ALL
- OSC$MANAGE_*servicegroup-name*

The identifier OSC$_MANAGE_*servicegroup-name* defines a dedicated service group an unprivileged user is allowed to manage.

Unprivileged users are allowed to view the status of service groups.

7.3.2 How to display Service Group states

In order to display the status of service groups execute the OSC$MGR command:

```
OSC$MGR> SHOW SRVGRP servicegroup-name
```

The *servicegroup-name* parameter defines the service group for which status information will be displayed. Full wildcard support is provided for the *servicegroup-name* parameter. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within the parameter.

The displayed status content depends on the qualifiers applied. For detailed information about the available command qualifiers please refer to the OSC$MGR online help.

7.3.3 How to online a Service Group

In order to bring service groups that are offline online, execute the following OSC$MGR command:

```
OSC$MGR> ONLINE SRVGRP servicegroup-name [/NODE=nodename]
```

If you omit the /NODE qualifier, as many instances of the service group will be started as defined by the OSC failover policy (see

5.2 Controlling OSC Failover Policy) and dependent on the current state of the service group within the OSC cluster members.

The *servicegroup-name* list parameter defines the service groups to start. A comma separated list of service groups can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list. Thus, several service groups can be started in parallel with a single command.

A service group cannot be started on an OSC node if the status of the service group is:

- ADMIN_WAIT
- FAULTED
- FROZEN
- DISABLED

If the /NODE qualifier is applied the service group is started on the node specified by the /NODE qualifier even if:

- The OSC failover policy is set to LOAD_BALANCING and the OSC node has insufficient load capabilities to run the service group
- One of the service groups listed in the service group exclude list (attribute **SrvGrpDisAllow**) is already running (online) or starting up (see also 5.3.2 Controlling Service Group Concurrent Execution).

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.3.4 How to offline a Service Group

In order to bring service groups that are online offline execute the OSC$MGR command:

OSC$MGR> OFFLINE SRVGRP *servicegroup-name* [/NODE=nodename]

If you omit the /NODE qualifier all existing online OSC instances of the service group will be shutdown.

The *servicegroup-name* list parameter defines the service groups to stop. A comma separated list of service groups can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere

within each item of the comma separated parameter list. Thus, several service groups can be stopped in parallel with a single command.

A service group cannot be shutdown on an OSC node if the status of the service group is:

- FAULTED (service group is already offline)
- FROZEN
- DISABLED

If the /FORCE qualifier is applied, OSC tries to offline the service group even if the service group is in ADMIN_WAIT state. If the /FORCE qualifier is omitted the command fails if the service group is in ADMIN_WAIT state.

Note

If the /FORCE qualifier is applied and one of the service group's resources is in OFFLINE| ADMIN_WAIT state, the offline command is not executed on this resource since its state indicates that it is already OFFLINE. Thus, the resource state does not change and the ADMIN_WAIT state does not get cleared even if the OFFLINE command succeeds. In this case the ADMIN_WAIT state has to be explicitly cleared.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.3.5 How to switch a Service Group

Instead of shutting down a service group on one node and starting it on another one, you can directly switch-over the service group to another node. To switch a service group execute either of the OSC$MGR commands:

```
OSC$MGR> SWITCH SRVGRP servicegroup-name
         [/TARGET_NODE=node-name/SOURCE_NODE=node-name]

OSC$MGR> MOVE SRVGRP servicegroup-name
         [/TARGET_NODE=node-name/SOURCE_NODE=node-name]
```

The *servicegroup-name* parameter defines the service group that is to be switched over. The use of wildcard characters is not permitted. With the /TARGET_NODE and /SOURCE_NODE qualifier you can explicitly define the source and target node of the switch operation.

You cannot apply the command if the service group is a **Parallel** service group. A **Parallel** service group is online concurrently on all OSC cluster members.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.3.6 How to freeze a Service Group

In order to temporarily remove service groups from being failed-over or to temporarily prevent that a particular OSC node is selected to host the service group, the user can freeze service groups. For detailed information about freezing a service group, please refer to the section5.3 Controlling OSC Behavior at the Service Group Level.

To freeze a service group execute the OSC$MGR command

OSC$MGR> FREEZE SRVGRP *servicegroup-name* [/NODE=nodename]

This command implicitly freezes all services and all **On-Off** resources that are members of the service group, but not **On-Only** or **Persistent** resources. To freeze **On-Only** or **Persistent** resources these resources have to be explicitly disabled with the FREEZE RESOURCE command (see section 7.5.5 How to freeze a Resource).

The *servicegroup-name* list parameter defines the service groups to freeze. A comma separated list of service groups can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the service group is 'frozen' on all nodes it is configured to run on. Otherwise the service group will be 'frozen' only on the node defined by the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.3.7 How to unfreeze a Service Group

To 'unfreeze' service groups execute the OSC$MGR command

OSC$MGR> UNFREEZE SRVGRP *servicegroup-name* [/NODE=node-name]

The *servicegroup-name* list parameter defines the service groups to 'unfreeze'. A comma separated list of service groups can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.
If you omit the /NODE qualifier the service group will be 'unfreezed' on all OSC nodes it is configured to run. Otherwise the service group will be 'unfreezed' only on the OSC node defined by the /NODE qualifier.

Note

The UNFREEZE SRVGRP command implicitly 'unfreezes' all **On-Off** resources of a service group only. If the service also contains 'frozen' **On-Only** or **Persistent** resources these resources have to be explicitely 'unfreezed' with the UNFREEZE RESOURCE command.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.3.8 How to disable a Service Group

In order to remove a service group from being managed by OSC during normal OSC operation the service group has to be disabled. Disabling a service group is required whenever the user wants to execute operative commands outside of OSC control on **On-Off** resources of a service group to ensure that OSC does not intervene on resource fault conditions and that OSC state change events are not triggered. For detailed information about disabling a service group, please refer to the section 5.3 Controlling OSC Behavior at the Service Group Level.

To disable a service group execute the OSC$MGR command

OSC$MGR> DISABLE SRVGRP *servicegroup-name* [/NODE=node-name]

This command implicitly disables all services and all **On-Off** resources that are members of the service group, but not **On-Only** or **Persistent** resources. To disable **On-Only** or **Persistent** resources these resources have to be explicitly disabled with the DISABLE RESOURCE command (see section
7.5.7 How to disable a Resource).

The *servicegroup-name* list parameter defines the service groups to disable. A comma separated list of service groups can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the service group is disabled on all OSC nodes it is configured to run on. Otherwise the service group will be disabled only on the OSC node defined by the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.3.9 How to enable a Service Group

To enable a service group execute the OSC$MGR command

OSC$MGR> ENABLE SRVGRP *servicegroup-name* [/NODE=node-name]

The *servicegroup-name* list parameter defines the service groups to be enabled. A comma separated list of service groups can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the service group will be enabled on all OSC nodes it is configured to run on. Otherwise the service group will be enabled only on the OSC node defined by the /NODE qualifier.

Note

The ENABLE SRVGRP command implicitly enables all **On-Off** resources of a service group only. If the service also contains disabled **On-Only** or **Persistent** resources these resources have to be explicitly enabled with the ENABLE RESOURCE command.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.3.10 Clearing Service Group FAULT state

The fault state of a service group has to be manually reset in order to signal to OSC that the problem that caused the fault state has been resolved and that OSC can take over control again. For more detailed information about clearing service group fault states please refer to section

5.7 Clearing OSC states.

To clear the service group fault state execute the OSC$MGR command:

```
OSC$MGR> CLEAR SRVGRP servicegroup-name/FAULT
                                        [/NODE=node-name]
```

This command implicitly clears the fault state of all services and resources that are members of this service group.

The *servicegroup-name* list parameter defines the service groups for which the fault state will be cleared. A comma separated list of service groups can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you want to clear the fault state of a service group on a dedicated OSC node apply the /NODE qualifier. Otherwise omit the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.3.11 Clearing Service Group ADMIN_WAIT state

The ADMIN_WAIT state of a service group has to be manually reset in order to signal to OSC that the problem that caused the ADMIN_WAIT state has been resolved and that OSC can take over control again. For more detailed information about clearing service group ADMIN_WAIT state please refer to section

5.7 Clearing OSC states.

To clear the service group ADMIN_WAIT state, execute the OSC$MGR command:

```
OSC$MGR> CLEAR SRVGRP servicegroup-name/ADMIN_WAIT
                                        [/NODE=node-name]
```

This command implicitly clears the ADMIN_WAIT state of all services and resources that are members of this service group.

The *servicegroup-name* list parameter defines the service groups for which the ADMIN_WAIT state will be cleared. A comma separated list of service groups can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you want to clear the ADMIN_WAIT state of a service group on a dedicated OSC node apply the /NODE qualifier. Otherwise the ADMIN_WAIT state of a service group will be cleared on all OSC nodes where the referenced service group is in the ADMIN_WAIT state.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.4 How to manage OSC Services

7.4.1 Required privileges to manage the OSC Services

In order to execute OSC service management commands you have to be logged in with the required privileges (see also section 4.4 OSC user privileges) or the user has at least one of the following identifiers assigned:

- OSC$MANAGE_ALL
- OSC$MANAGE_*service-name*

The identifier OSC$_MANAGE_*service-name* defines a dedicated service an unprivileged user is allowed to manage.

Unprivileged users are only allowed to view the status of services.

7.4.2 How to display Service states

In order to display the status of OSC services execute the OSC$MGR command:

```
OSC$MGR> SHOW SERVICE service-name
```

The *service-name* parameter defines the service for which the status information will be displayed. Full wildcard support is provided for the *service-name* parameter. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within the parameter.

The displayed status content depends on the qualifiers applied. For detailed information about the available command qualifiers please refer to the OSC$MGR online help.

7.4.3 How to online a Service

The user can bring a particular service that is offline online without starting the whole service group the service is a member of.

To start a service execute the OSC$MGR command

```
OSC$MGR> ONLINE SERVICE service-name [/NODE=nodename]
```

The *service-name* list parameter defines the services to start. A comma separated list of services can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list. Thus, several services can be started in parallel with a single command.

If you omit the /NODE qualifier OSC checks the status of the parent service group and brings the service online on all the OSC nodes where the parent service group is already online. If the parent service group of the service is not online on any OSC cluster member the command fails if the /NODE qualifier is not applied.

A service cannot be started on an OSC node if the status of the service is:

- ADMIN_WAIT
- FAULTED
- FROZEN

- DISABLED

Note

The ONLINE SERVICE command only starts the service addressed by the *service-name* parameter, but not any child or parent service. Thus, if this command is being used to online a service that is part of a dependency tree, ensure that the services are started in the right order.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.4.4 How to offline a Service

The user can bring a particular service that is online offline without causing OSC to offline or failover the service group the service is a member of.

To stop a service execute the OSC$MGR command

OSC$MGR> OFFLINE SERVICE *service-name* [/NODE=nodename]/FORCE

This command is useful if one has to stop a service (i.e. maintenance reasons) without relocating or stopping the whole parent service group.

The *service-name* list parameter defines the services to stop. A comma separated list of services can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list. Thus, several services can be stopped in parallel with a single command.

If you omit the /NODE qualifier the service will be stopped on all OSC nodes the service is online on. The use of wildcard characters is not permitted for this qualifier.

A service cannot be shutdown on an OSC node if the status of the service is:

- FAULTED (resource is already offline)
- FROZEN
- DISABLED

Note

The OFFLINE SERVICE command only stops the service addressed by the *service-name* parameter, but not any child or parent service.Thus, if this command is being used to offline a service that is part of a dependency tree, ensure that the services are stopped in the right order.

If the /FORCE qualifier is applied, OSC tries to offline the service even if the service is in ADMIN_WAIT state. If the /FORCE qualifier is omitted the command fails if the service is in ADMIN_WAIT state.

Note

If the /FORCE qualifier is applied and one of service's resources is in OFFLINE| ADMIN_WAIT state, the offline command is not executed on this resource since its state indicates that it is already OFFLINE. Thus, the resource state does not change and the ADMIN_WAIT state does not get cleared, even if the OFFLINE command succeeds. In this case the ADMIN_WAIT state has to be cleared explicitly.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.4.5 How to freeze a Service

Freezing a service freezes all its **On-Off** resources. Freezing a resource prevents the OSC agent from launching any action routines except the monitor routine. Thus, if a service is 'frozen' neither the online, offline nor the clean action routine is called for any of its **On-Off** resources.

When a service is 'frozen' the service group the service is a member of, is automatically marked 'frozen' too.

Thus, the effect of freezing a service is comparable to freezing a service group. The main difference is that if you freeze a particular service, only the **On-Off** resources of this service are 'frozen'. Other resources that are members of the same service group, but not members of the 'frozen' service are unaffected.

To freeze a service execute the OSC$MGR command

OSC$MGR> FREEZE SERVICE *service-name* [/NODE=node-name]

As stated previously, this command implicitly freezes all **On-Off** resources that are members of the service, but not **On-Only** or **Persistent** resources. To freeze **On-Only** or **Persistent** resources these resources have to be explicitly disabled with the FREEZE RESOURCE command (see section 7.5.5 How to freeze a Resource).

The *service-name* list parameter defines the services to freeze. A comma separated list of services can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the service is 'frozen' on all OSC cluster members. Otherwise the service will be 'frozen' only on the node defined by the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.4.6 How to unfreeze a Service

To 'unfreeze' a service execute the OSC$MGR command

OSC$MGR> UNFREEZE SERVICE *service-name* [/NODE=node-name]

The *service-name* list parameter defines the services to 'unfreeze'. A comma separated list of services can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the service is 'unfreezed' on all OSC cluster members. Otherwise the service will be 'unfreezed' only on the OSC node defined by the /NODE qualifier.

Note

The UNFREEZE SERVICE command implicitly 'unfreezes' all **On-Off** resources of a service only. If the service contains also 'frozen' **On-Only** or **Persistent** resources these resources have to be explicitly 'unfreezed' with the UNFREEZE RESOURCE command.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.4.7 How to disable a Service

Disabling a service removes its **On-Off** resources from status monitoring. When a service is disabled the current status of its **On-Off** resources remains unchanged until the service is enabled again.

When a service is disabled the service group the service is a member of is automatically disabled too.

Thus, the effect of disabling a service is comparable to disabling a service group. The main difference is that if you disable a particular service only the resources of that service are disabled. Other resources that are members of the same service group, but not of the disabled service are unaffected.

To disable a service execute the OSC$MGR command

OSC$MGR> DISABLE SERVICE *service-name* [/NODE=node-name]

As stated previously, this command implicitly disables all **On-Off** resources that are members of the service, but not **On-Only** or **Persistent** resources. To disable **On-Only** or **Persistent** resources these resources have to be explicitly disabled with the DISABLE RESOURCE command (see section
7.5.7 How to disable a Resource).

The *service-name* list parameter defines the services to disable. A comma separated list of services can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit /NODE qualifier the service is disabled on all OSC nodes it is configured to run. Otherwise the service will be disabled only on the node defined by the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.4.8 How to enable a Service

To enable a service execute the OSC$MGR command

OSC$MGR> ENABLE SERVICE *service-name* [/NODE=node-name]

The *service-name* list parameter defines the services to be enabled. A comma separated list of services can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the service is enabled on all OSC cluster members. Otherwise the service will be enabled only on the OSC node defined by the /NODE qualifier.

Note

The ENABLE SERVICE command implicitly enables all **On-Off** resources of a service only. If the service also contains disabled **On-Only** or **Persistent** resources these resources have to be explicitly enabled with the ENABLE RESOURCE command.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.4.9 Clearing Service FAULT state

The fault state of services has to be manually reset in order to signal to OSC that the problem that caused the fault state has been resolved and that OSC can take over control again. The fault state of a service can be cleared either implicitly using the CLEAR SRVGRP command or explicitly. For more detailed information about clearing a resource fault state please refer to the section

5.7 Clearing OSC states.

To clear the service fault state explicitly execute the OSC$MGR command:

```
OSC$MGR> CLEAR SERVICE service-name /FAULT
                                    [/NODE=node-name]
```

This command implicitly clears the fault state of all resources that are members of this service.

The *service-name* list parameter defines the services for which the fault state will be cleared. A comma separated list of services can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you want to clear the fault state of a service on a dedicated OSC node apply the /NODE qualifier. Otherwise the fault state of a service will be cleared on all OSC nodes where the addressed service is in FAULT state.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.4.10 Clearing Service ADMIN_WAIT state

The ADMIN_WAIT state of service has to be manually reset in order to signal to OSC that the problem that caused the ADMIN_WAIT state has been resolved and that OSC can take over control again. The ADMIN_WAIT state of a service can be cleared either implicitly using the CLEAR SRVGRP command or explicitly. For more detailed information about clearing service ADMIN_WAIT state please refer to section

5.7 Clearing OSC states.

To clear the service ADMIN_WAIT state explicitly execute the OSC$MGR command:

```
OSC$MGR> CLEAR SERVICE service-name /ADMIN_WAIT
                                    [/NODE=node-name]
```

This command implicitly clears the ADMIN_WAIT state of all resources that are members of this service.

The *service-name* list parameter defines the services for which the ADMIN_WAIT state will be cleared. A comma separated list of services can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you want to clear the ADMIN_WAIT state of a service on a dedicated OSC node apply the /NODE qualifier. Otherwise the ADMIN_WAIT state of a service will be cleared on all OSC nodes where the addressed service is in ADMIN_WAIT state.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.5 How to manage OSC resources

7.5.1 Required privileges to manage the OSC resources

In order to execute OSC service management commands you have to be logged in with the required privileges (see also section 4.4 OSC user privileges) or the user has to have at least one of the following identifiers assigned:

- OSC$MANAGE_ALL
- OSC$MANAGE_*resource-type*
    - OSC$RES_ALL
    - OSC$RES_*resource-name*

An unprivileged user is allowed to manage all resources of a particular type if the identifiers OSC$MANAGE_*resource-type* and OSC$RES_ALL

have been assigned to the user. An unprivileged user can be restricted to manage only dedicated resources. In this case the identifiers OSC$MANAGE_*resource-type* and OSC$RES_*resource-name* have been assigned to the user (see section 4.4.2.5 OSC$MANAGE_resource-type for detailed description).

Unprivileged users are only allowed to view the status of resources.

7.5.2 How to display resource states

In order to display the status of resources execute the OSC$MGR command:

```
OSC$MGR> SHOW RESOURCE resource-name
```

The *resource-name* parameter defines the resource for which status information will be displayed. Full wildcard support is provided for the *resource-name* parameter. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within the parameter.

The displayed status content depends on the qualifiers applied. For detailed information about the available command qualifiers please refer to the OSC$MGR online help.

7.5.3 How to online a Resource

The user can bring a particular resource that is offline online without starting the service group(s) the resource is a member of.

To start a resource execute the OSC$MGR command

```
OSC$MGR> ONLINE RESOURCE resource-name [/NODE=nodename]
```

The *resource-name* list parameter defines the resources to start. A comma separated list of resources can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list. Thus, several resources can be started in parallel with a single command

If you omit the /NODE qualifier OSC checks the status of the resource parent service groups (a resource can be a member of several services and service groups) and brings the resource online on all these OSC nodes where the parent service groups are already online. If the parent service groups of the resource are not online on any OSC cluster member the command fails, if the /NODE qualifier is not applied.

A resource cannot be started on an OSC node if the status of the resource is:

- ADMIN_WAIT
- FAULTED
- FROZEN
- DISABLED

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.5.4 How to offline a Resource

The user can bring a particular resource that is online offline without causing OSC to failover the service group(s) the resource is a member of.

To stop a resource execute the OSC$MGR command

OSC$MGR> OFFLINE RESOURCE *resource-name* [/NODE=nodename]/FORCE

This command is useful if one has to stop a resource (i.e. maintenance reasons) without relocating or stopping the parent service group(s).

The *resource-name* list parameter defines the resources to stop. A comma separated list of resources can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list. Thus, several resources can be stopped in parallel with a single command

If you omit the /NODE qualifier the resource is stopped on all OSC nodes where the resource is online. The use of wildcard characters is not permitted for this qualifier.

A resource cannot be shutdown on an OSC node if the status of the resource is:

- FAULTED (resource is already offline)

- FROZEN
- DISABLED

If the /FORCE qualifier is applied, OSC tries to offline the resource even if the resource is in ADMIN_WAIT state. If the /FORCE qualifier is omitted the command fails if the resource is in ADMIN_WAIT state.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.5.5 How to freeze a Resource

Freezing a resource prevents the OSC agent from launching any action routines except the monitor routine. Thus, if a resource is 'frozen' neither the online, offline nor the clean action routine is called.

When a resource is 'frozen' all services and service groups the resource is a member of are automatically marked 'frozen' too.

Thus, the effect of freezing a resource is comparable to freezing a service group or service. The main difference is that if you freeze a particular service group (service), only this service group (service) is 'frozen' even though the service group (service) may contain resources that are members of other service groups (services) too, because only **On-Off** resources are implicitly 'frozen' when a service group or service is 'frozen'. With the FREEZE RESOURCE command any resource type (**On-Off**, **On-Only** and **Persistent**) can be 'frozen'. For detailed information about freezing a resource, please refer to section 5.5 Controlling OSC Behavior at the Resource Level.

To freeze a resource execute the OSC$MGR command

OSC$MGR> FREEZE RESOURCE *resource-name* [/NODE=node-name]

The *resource-name* list parameter defines the resources to freeze. A comma separated list of resources can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the resource is 'frozen' on all OSC cluster members. Otherwise the resource will be 'frozen' only on the node defined by the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.5.6 How to unfreeze a Resource

To 'unfreeze' a resource execute the OSC$MGR command

OSC$MGR> UNFREEZE RESOURCE *resource-name* [/NODE=node-name]

The *resource-name* list parameter defines the resources to 'unfreeze'. A comma separated list of resources can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the resource is 'unfrozen' on all OSC cluster members. Otherwise the resource will be 'unfreezed'only on the OSC node defined by the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.5.7 How to disable a Resource

Disabling a resource removes the resource from status monitoring. When a resource is disabled the current status of the resource remains unchanged until the resource is enabled again, since the OSC agent does not execute any action routines for disabled resources.

When a resource is disabled all services and service groups the resource is a member of are automatically disabled too.

Thus, the effect of disabling a resource is comparable to disabling a service group or service. The main difference is that if you disable a particular service group (service) only this service group (service) is disabled even though the service group (service) may contain resources that are members of other service groups (services) too, because only **On-Off** resources are implicitly disabled when a service group or service is disabled. With the DISABLE RESOURCE command any resource type (**On-Off**, **On-Only** and **Persitent**) can be disabled. For detailed information about disabling a resource, please refer to section 5.5 Controlling OSC Behavior at the Resource Level.

To disable a resource execute the OSC$MGR command

OSC$MGR> DISABLE RESOURCE *resource-name* [/NODE=node-name]

The *resource-name* list parameter defines the resources to disable. A comma separated list of resources can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit /NODE qualifier the resource is disabled on all OSC nodes it is configured to run on. Otherwise the resource will be disabled only on the node defined by the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.5.8 How to enable a Resource

To enable a resource execute the OSC$MGR command

OSC$MGR> ENABLE RESOURCE *resource-name* [/NODE=node-name]

The *resource-name* list parameter defines the resources to be enabled. A comma separated list of resources can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you omit the /NODE qualifier the resource is enabled on all OSC cluster members. Otherwise the resource will be enabled only on the OSC node defined by the /NODE qualifier.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.5.9 Clearing Resource FAULT state

The fault state of a resource has to be manually reset in order to signal to OSC that the problem that caused the fault state has been resolved and that OSC can take over control again. The fault state of a resource can be cleared either implicitly using the CLEAR SRVGRP or the CLEAR SERVICE

command or explicitly. For more detailed information about clearing a resource fault state please refer to the section

5.7 Clearing OSC states.

To clear the resource fault state explicitly execute the OSC$MGR command:

```
OSC$MGR> CLEAR RESOURCE resource-name /FAULT
                                    [/NODE=node-name]
```

The *resource-name* list parameter defines the resources for which the fault state will be cleared. A comma separated list of resources can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you want to clear the fault state of a resource on a dedicated OSC node apply the /NODE qualifier. Otherwise the fault state of a resource will be cleared on all OSC nodes where the addressed resource is in FAULT state.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.5.10 Clearing Resource ADMIN_WAIT state

The ADMIN_WAIT state of a resource has to be manually reset in order to signal to OSC that the problem that caused the ADMIN_WAIT state has been resolved and that OSC can take over control again. The ADMIN_WAIT state of a resource can be cleared either implicitly using the CLEAR SRVGRP or the CLEAR SERVICE command or explicitly. For more detailed information about clearing resource ADMIN_WAIT state please refer to section

5.7 Clearing OSC states.

To clear the resource ADMIN_WAIT state explicitely execute the OSC$MGR command:

OSC$MGR> CLEAR RESOURCE *resource-name* /ADMIN_WAIT
[/NODE=node-name]

The *resource-name* list parameter defines the resources for which the ADMIN_WAIT state will be cleared. A comma separated list of resources can be applied. Full wildcard support is provided for each element of the parameter input list. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within each item of the comma separated parameter list.

If you want to clear the ADMIN_WAIT state of a resource on a dedicated OSC node apply the /NODE qualifier. Otherwise the ADMIN_WAIT state of a resource will be cleared on all OSC nodes where the addressed resource is in ADMIN_WAIT state.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.6 How to manage OSC transactions

The user has full control of all transactions currently in progress on the OSC cluster. The SHOW TRANSACTION command provides an overview of all ongoing transactions and the CANCEL TRANSACTION command can be utilized to cancel either a specific or all transactions in progress.

7.6.1 Required privileges to manage OSC transactions

In order to cancel an OSC transaction the user is either logged in as a privileged OSC user (see also section 4.4 OSC user privileges) or the user has the following identifier assigned:

- OSC$MANAGE_ALL

Unprivileged users are only allowed to view the status of OSC transactions.

7.6.2 How to display OSC transactions

The SHOW TRANSACTION command provides an overview of all ongoing transactions in the OSC cluster.

```
OSC$MGR> SHOW TRANSACTION
```

This command displays the transaction state (GOING ONLINE, GOING OFFLINE etc.), the transaction ID, the name of the OSC entity (Service Group, Service or Resource) the transaction is acting upon, the start time of the transaction and whether the transaction was started by a user or by the OSC master control.

If no qualifier or the /ALL qualifier is applied, all transactions are displayed ordered by Service Group, Service and Resources.

If the /SRVGRP qualifier is applied only active Service Group transactions are displayed.

If the /SERVICE qualifier is applied only active Service transactions are displayed.

If the /RESOURCE qualifier is applied only active resource transactions are displayed.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.6.3 How to cancel OSC transactions

The CANCEL TRANSACTION command cancels either a single or all transactions currently in progress.

```
OSC$MGR> CANCEL TRANSACTION [/ID /ALL /NODE]
```

If the /ID qualifier is applied the transaction associated with the transaction ID assigned to the qualifier is canceled. To display the transaction IDs of all ongoing transactions execute the SHOW TRANSACTION command.

If the /ALL qualifier is applied all transactions are canceled OSC cluster-wide.

If the /NODE qualifier in combination with /ALL qualifier is applied all transactions currently executing on the specified node are canceled.

The /ID and /NODE qualifiers, as well the combination of /ID and /ALL qualifiers are mutually exclusive.

If neither the /ID not the /ALL qualifier is applied the command fails.

If one starts a transaction in synchronous mode[1] and this transaction is canceled, SS$_CANCEL is returned to the caller of this transaction signaling, that the transaction has been canceled by another user.

If a transaction is canceled all associated resource action routines currently executing (except the monitor action routines) are immediately stopped by OSC. These resources and their respective parent services and service groups are marked as CANCELED. Since the action routines are stopped immediately by OSC, these routines will not have responded with a valid exit code and so from the OSC perspective these resources are deemed to be in a limbo state. Thus, the ADMIN_WAIT bit is set in the state fields of these resources and their respective parent services and service groups.

Note

Thus, if a resource is marked as CANCELED and ADMIN_WAIT, verify manually whether or not the system resource(s) referenced by the OSC resource is in a valid state before the ADMIN_WAIT state is cleared.

For detailed information about the command syntax please refer to the OSC$MGR online help.

7.7 OSC event console

Any OSC event class can be configured to forward OSC events to the connected event monitoring consoles. To define your OSC$MGR session as an event monitoring console, execute the OSC$MGR command:

```
OSC$MGR> MONITOR EVENTS
```

[1] An OSC command is started in synchronous mode if the command is called from the DCL command line prompt or if the DEFINE MODE/SYNC was executed prior to the command. For more information about the DEFINE MODE command please refer to the OSC$MGR command line help.

An OSC$MGR session remains an event monitoring console as long as the connection to the active OSC master control process is not disconnected. If an OSC$MGR session gets disconnected from the active OSC master control process and re-establishes the connection (CONNECT command) the MONITOR EVENT command has to be reapplied to define the current OSC$MGR session as an event monitoring console. An OSC$MGR session will be disconnected if the active OSC master control switches over to another OSC node (either due to a user command or automatically due to system failure).

The messages of the OSC event classes listed below are sent to all connected consoles regardless if they have been defined as monitoring event consoles or not:

- OSCCTRL_CLUSTER_EVT
- OSCCTRL_CONTROL_EVET
- OSCCTRL_STATE_EVT

For detailed information about OSC event classes please refer to section 6 OSC event notification.

7.8 SYSTEM Startup and Shutdown

When using OSC to manage you applications, time should be spent considering how these applications should be started, whether independent of OSC control or whether OSC should manage the complete startup. Both approaches are possible, but in both cases appropriate configuration descisions have to be made and implemented.

Similarly appropriate configuration decisions have to be made and implemented to cover planned shutdowns of an OSC Node to avoid unexpected failover scenarios.

Consider using a specific OSC entry in the system shutdown procedure SYS$STARTUP:SHUTDWN.COM that initiates an:

```
OSC SHUTDOWN/NODE/FORCE=MIGRATE
```

to ensure a controlled failover of all active service groups on the node to be shutdown to the remaining OSC members

7.9 Managing OSC using DCL scripts

Any OSC$MGR management command except the SHOW CLUSTER and the MONITOR EVENT commands can be directly called from the DCL command line prompt. Thus, OSC$MGR commands can be called from DCL command scripts.

In order to execute a management command directly from the DCL prompt a foreign command symbol that refers to the OSC$MGR image must be defined. Start the OS$MGR utility using the foreign command symbol and apply the OSC$MGR command to be executed.

Example:

```
$ OSC$MGR :== $OSC$BIN:OSC$MGR
$ OSC$MGR SWITCH SRVGRP PERFDAT
```

If an OSC$MGR command is directly called from the DCL command line as shown above, the command is executed in synchronous mode. This means that OSC$MGR does not return control to DCL until the OSC master control process signals that the transaction triggered by the OSC$MGR management command has completed or timed out. OSC$MGR assigns the final transaction status to the status symbol $STATUS.

In addition, if a SHOW command is executed, OSC$MGR assigns to the global symbol OSC$NODE all OSC node names, that matches the SHOW command filter criteria, as a comma separated list.

Example:

```
$ OSC$MGR :== $OSC$BIN:OSC$MGR
$ OSC$MGR SHOW SRVGRP PERFDAT/STATUS=ONLINE
$ SHOW SYMBOL OSC$NODE
OSC$NODE == "VMSTM1,VMSTM4"
```

The SHOW SRVGRP PERFDAT/STATUS=ONLINE command displays all online instances of the service group PERFDAT. Since the service group PERFDAT is online on node VMSTM1 and VMSTM4 the symbol OSC$NODE contains the value “VMSTM1, VMSTM4”.

7.9.1 Return codes

The OSC$MGR utility provides the status codes listed below when an OSC$MGR command is directly called from the DCL command line prompt

Status Code	Status Value	Description
SS$_NORMAL	%X1	Transaction completed successfully
SS$_INVARG	%XFCA	Invalid command syntax
SS$_TIMEOUT	%X22C	Transaction timed out
SS$_CANCEL	%X830	At least one of the started transactions has been canceled by another user
SS$_LINKDISCON	%X20EC	OSC$MGR was unable to establish a link to the active OSC master control process or the OSC master control process disconnected the logical link during command execution
SS$_REJECT	%x294	Command execution was denied by the OSC master control process
SS$_OPINCOMPL	%X2D4	At least one of the started transactions has failed

7.9.2 DCL managing script example

This DCL script switches the service group PERFDAT to a dedicated node. The target node is passed in P1 to the DCL script.

```
$! P1 - Target node to run PERFDAT
$!
$ OSC$MGR :== $OSC$BIN:OSC$MGR
$ RETRY_CNT = 0
$!
$ SWICH_LOOP:
$!
$! Check if SrvGrp PERFADT is already ONLINE on node P1
$!
$ OSC$MGR SHOW SRVGRP PERFDAT/NODE='P1/STATUS=ONLINE
$ IF $STATUS .NE. 1
$ THEN
```

```
$!      Send alert
$ ...
$!      and exit
$       EXIT
$ ENDIF
$!
$ IF OSC$NODE .EQS. ""
$ THEN
$!
$!      SrvGRP PERFDAT is not ONLINE on the node
$!      Switch SrvGrp PERFDAT
$!
$       OSC$MGR SWITCH SRVGRP PERFDAT/TARGET='P1
$       SWITCH_STATUS = $STATUS
$       IF SWITCH_STATUS .NE. 1
$       THEN
$!          Failure -> send alert
$           ...
$!          and exit
$           EXIT
$       ENDIF
$       IF SWITCH_STATUS .EQ. %X2D4
$       THEN
$!
$!          Command was rejected by OSC$CTRL
$!          It is very likely that the same
$!          transaction is already in progress
$!          Thus, wait for a while and restart.
$!
$           RETRY_CNT = RETRY_CNT + 1
$           IF (RETRY_CNT .LT. 5) THEN GOTO SWITCH_LOOP
$       ENDIF
$ ENDIF
$!
$ EXIT
```

8

OSC configuration guidelines

This section provides information about the OSC configuration rules and guidelines of how to plan and create a valid OSC configuration. It does not provide a detailed description of the available OSC$CFG commands. For a detailed description of all the available OSC$CFG commands syntax please refer to the OSC$CFG online help.

8.1 Steps to create a valid OSC configuration

1. Carefully read the OSC configuration rules described in the next section
2. Plan you configuration according to the configuration rules
3. Create a new OSC configuration database and configure all items according to you configuration plan
4. Verify the configuration
5. Define the new OSC configuration database as the default OSC configuration

8.2 OSC configuration rules

When planning a new OSC configuration you have to be aware of the following configuration rules:

1. The OSC cluster definition section of an OSC configuration database has to contain at least one member of the OpenVMS cluster you want to run VSI OpenVMS ServiceControl (OSC) on.

2. The OSC failover policy has to be defined in the OSC cluster definition section.

3. If the selected OSC failover policy is LOAD-BALANCING, OSC uses a load-balancing mechanism to determine which OSC system hosts an application (service group) during startup, or after an application or server fault. It is mandatory to define the following attributes (for detailed information about OSC failover policies please refer to

section

5.2 Controlling OSC Failover Policy):

- **OscCtrlNodeLoadCap** in the OSC cluster definition section
- **SrvGrpLoad** for each service group

4. OSC service group, service and resource names have to be unique within the configuration database

5. All OSC service groups have to contain a valid service group execution node list. For detailed information about the execution node list (attribute **SrvGrpNodes**) of a service group please refer to section

5.2 Controlling OSC Failover Policy and 5.3.1 Controlling Service Group Execution Nodes.

6. An OSC service group has to contain 1..n OSC services

7. An OSC service can be assigned only to one OSC service group

8. An **On-Off** resource can be assigned to only one OSC service. Thus, an **On-Off** resource can be a member of only one OSC service group.

9. **On-Only** and **Persistent** resources can be members of different OSC services and thus implicitly, can be members of different OSC service groups.

10. **On-Off** resources can depend on other:
    - **On-Off** resources
    - **On-Only** resources
    - **Persistent** resources

11. **On-Only** resources can depend on other:
    - **On-Only** resources
    - **Persistent** resources

    but not on:
    - **On-Off** resources

12. **Persistent** resources can depend on other:
    - **Persistent** resources

    but not on:
    - **On-Off** resources
    - **On-Only** resources

8.3 OSC configuration planning

Configuring OSC requires detailed planning. The check lists provided in this section lists the most important planning issues one has to consider before configuring OSC.

8.3.1 Service Group, Service and Resource configuration check list

This section provides a check list for resource, service and service group configuration planning.

1. Define all the services (your applications) that should be managed by OSC and assign unique names for all these OSC services.

   Defining a service means determining which resources are required to run that service.

   Use OSC service names that can be easily associated with the managed application. For example, if one of your services is an Oracle database named ORADWH use the same name for the OSC service.

2. Create resource dependency diagrams for all OSC services.

   A resource dependency diagram contains all the resources required to run your application (service) and illustrates their interdependencies (see Fig. 8.2 and Fig. 8.3. or Fig. 4.5 of section 4.2.1.2 Resource Dependencies)

3. Check if all OSC agents required to manage the different OSC resource type exist.

   If a resource exists that cannot be managed by one of the OSC agents bundled with VSI OpenVMS ServiceControl (see section 10 Bundled OSC agents), develop an OSC agent that manages this resource type as described in section 9 Developing new OSC agents.

4. Check if the OSC resources listed in the resource dependency diagram can be managed independently.

   This means, that a child resource is not implicitly shutdown when a parent resource is taken offline, and that a parent resource is not

automatically started when one of its child resources is started. Otherwise the overall OSC behavior will be unpredictable

Whether or not resources of different types are independent of each other within this context depends on the online and offline action routines of the appropriate OSC agents. All OSC agents bundled with VSI OpenVMS service control (see section 10 Bundled OSC agents) fulfill this condition as long as the action routines called by the OSC agents are not re-defined at the resource level.

5. Assign unique OSC resources names.

   Use resource names that can be easily associated with the managed resource. An OSC resource has to be defined according to the formatting rule shown below:

   Resource-type::resource-name[@node-name]

   The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon "::". To define a resource to be node specific, the node name has to be applied to the resource name together with the "@" prefix.

6. Check if all resources fulfill the resource type specific prerequisites so that they can be managed by the appropriate OSC agents.

   For example, an Oracle 10 database cannot be managed by the OSC Oracle 10 agent (OscAgtORA) if the database is configured to start on a dedicated node. Thus, check the NODE property in the Oracle 10 database property file. For detailed information about the resource type specific prerequisites of the OSC agents bundled with VSI OpenVMS ServiceControl please refer to section 10 Bundled OSC agents.

7. Define the resource type specific attributes for all resources.

   Resource type specific attributes define the set of parameters passed to the action routines to online, offline, monitor or clean up a resource. Thus, these attributes are the "resource identifiers" (see also 9.3.1 Resource type specific attributes). For detailed information about the resource type specific attributes of the OSC agents bundled with VSI OpenVMS ServiceControl refer to section 10 Bundled OSC agents.

8. Mark all OSC node specific resources in the resource dependency diagrams.

   OSC resources have to be configured node specific if they reference node specific system resources. For example, one of the OSC resources in the resource dependency diagrams refers to the system disk. If no cluster common system disk exists, the OSC system disk resource has to be defined node specific if the OpenVMS cluster members use different system disk devices (i.e. system disk of member 1 is 1DKB100, system disk of member 2 is 1DKA0).

9. Check if the resources in the resource dependency diagrams are defined according to the resource related OSC configuration rules listed in the previous section.

   a. An **On-Only** resource must not depend on **On-Off** resources.
   b. A **Persistent** resource must not depend on **On-Off** or **On-Only** resources
   c. An **On-Off** resource cannot be member of different OSC services

10. Check the resource type of the resources in the resource dependency diagrams.

    The resource type of a resource is typically defined by the managing OSC agent. However, the resource type can be re-defined at the resource level. If the resource type has to be re-defined, mark the appropriate resource in the resource dependency diagrams.

    For example one of your resources is an RDB database that is opened cluster-wide. In this case define the RDB database as an **On-Only** resource since the database will not be closed on an OSC node if the service group the RDB database is a member of is switched to another OSC cluster member. In this case the resource type has to be defined at the resource level, since the OSC RDB default agent is defined to manage **On-Off** resource type.

    Not all OSC agents are capable of managing all resource types:
    - An OSC agent capable of managing **On-Off** resources is also capable of managing **On-Only** and **Persistent** resources. Thus, a resource managed by an **On-Off** OSC agent can be redefined to be an **On-Only** or **Persistent** resource type.

- An OSC agent capable of managing **On-Only** resources is also capable of managing **Persistent** resources. Thus, a resource managed by an **On-Only** OSC agent can de redefined to be a **Persistent** resource type.
- An OSC agent capable of managing **Persistent** resources can only manage **Persistent** resources. Thus, a resource managed by a **Persistent** OSC agent has to be a **Persistent** resource type.

11. Check if the managing OSC agent will call resource specific action routines for particular resources.

    For example, if your resource dependency diagram contains resources managed by the OSC process agent (see 10.4 OscAgtPRC - OSC process agent) the online and offline action routine have to be defined resource specific. The OSC process agent can handle **On-Off** and **On-Only** resources only if the online and offline action routines are defined at the resource level.

    There may be other reasons for modifying the default action routines of the managing OSC agent at the resource level.

12. Evaluate the online, offline and clean action routine processing times under worst case conditions.

    This is important if the startup and shutdown time of a resource significantly depends on the status of the environment.

    For example, the startup time of a database depends whether or not the database has to trigger a database recovery operation. Thus, if an Oracle database is failed-over by OSC due to a system failure (node crashed), the database recovery operation of the Oracle database can increases the startup time significantly.

    The timeout attributes (see 5.5.1 Resource Type Attributes) of a resource define how long OSC waits for the appropriate action routine to complete. Ensure that the values of the resource timeout attributes are sufficient so that the action routine processing time under worst condition is not considered as an action routine timeout, in order to avoid OSC "phantom" fault processing.

    Create a list that contains all resources that require specific timeout configuration (that differ from the default values defined by the resource type template – see section 10 Bundled OSC agents for

detailed information about the resource type attribute defaults of the bundled OSC agents).

13. Define the service groups according to the OSC configuration rules.

    Service groups are the OSC failover entities. OSC considers service groups as isolated management instances that do not affect each other and thus can be failed-over independently. Bundle all services (your applications) into one service group if they are required to run together on the same node. If services depend on each other create service dependency diagrams.

    Service group names have to be unique throughout the whole OSC configuration database. It is best practice to use a service group name that can be easily associated with the services of the service group.

    If a service group contains independent services decide whether or not the whole service group should failover if one of these services fails. If the service group should only failover if a particular service fails, assign the highest service priority to this service (attribute **ServicePriority** – see section 5.4.1 ServicePriority Attribute).

14. Define the service group category (**Failover**, **MultiInstance**, **Parallel**) and the execution node list for all service groups.

    The service group execution node list defines the OSC cluster members the service group is allowed to be started on. For more information about execution node lists and the service group categories please refer to 4.2.3.1 Service Group and 5.3.1 Controlling Service Group Execution Nodes.

15. If the selected OSC failover policy is LOAD-BALANCING define the workload values (attribute **SrvGrpLoad**) of all service groups.

16. Specify all service groups that a particular service group is not allowed to run with on the same node (attribute **SrvGrpDisAllow**).

    For detailed information how to prevent particular service groups from running on the same node please refer to section

5.2 Controlling OSC Failover Policy and 5.3.2 Controlling Service Group Concurrent Execution.

17. Define the OSC event classes

    For detailed information about OSC event classes please refer to section 6 OSC event notification.

8.3.2 OSC cluster configuration check list

1. OSC cluster name – attribute **OscCtrlClusterName**?

   Define the name of the OSC cluster.

2. OSC cluster members and votes – attribute **OscCtrlNodes**?

   Although an OSC cluster may consist of a subset of OpenVMS cluster members, it is recommended for simplicity sake, to define all OpenVMS cluster members as OSC cluster members.

   It is best practice to assign the same number of votes to the OSC cluster members as defined by the OpenVMS VOTES system parameter (OpenVMS votes), except if you have to consider special system resource restrictions to run your applications (see section 7.2 How to manage an OSC cluster in the previous chapter).

3. OSC failover policy – attribute **OscCtrlFailoverPolicy**?

   Two different service group OSC failover policies can be configured:
   - Static failover
   - Load-balanced failover

   The OSC cluster-wide failover policy is defined by the value assigned to the OSC master control attribute **OscCtrlFailoverPolicy**. Two keywords can be assigned to this attribute:
   - STATIC
     Static failover policy is selected for the OSC cluster.
   - LOAD-BALANCING
     Load balanced failover policy is selected for the OSC cluster

4. Expected votes – attribute **OscCtrlExpVotes**?

It is best practice to assign the same value to the OscCtrlExpVotes attribute as defined by the OpenVMS EXPECTED_VOTES system parameter, except when the minimum number of nodes required to start all services managed by OSC is greater than the minimum number of nodes to form an OpenVMS cluster.

Example:

You run a 5 node OpenVMS cluster. Each cluster member contributes one vote and the OpenVMS EXPECTED_VOTES system parameter is 5. Thus, when you (re) boot the OpenVMS cluster it starts processing if at least 3 OpenVMS cluster members are up, i.e. when quorum is achieved. However, the overall system resources of 3 nodes are not sufficient to start all the applications managed by OSC – 4 nodes are required. In this case assign the value 6 or 7 to the **OscCtrlExpVotes** attribute.

5. Startup-wait and state transition-wait – attribute **OscCtrlStartupWait**?

   The OSC state transition phase is initiated whenever:
   - OSC cluster is started
   - An OSC cluster member joins the OSC cluster
   - An OSC cluster member is removed from the OSC cluster
   - The active OSC master control process moves from one OSC cluster member to another

   During the state transition phase the OSC master control process requests the current state of all managed resources, services and service groups from all OSC agents and OSC service engines on the available OSC cluster members. This is done to ensure that the status information of the managed items maintained by the active OSC master control process is up to date before it starts service group processing. This mechanism avoids wrong service group offline/failover decisions based on incomplete and/or out of date status information.

   The duration of the state transition is defined by the OSC master control attribute **OscCtrlStartupWait**.

   Thus, the state transition wait time must be greater than the time required to start all OSC components within the cluster plus the maximum initial resource monitor processing time of all OSC agents in use.

How to evaluate the initial resource monitor processing time of an OSC agent:

- Test the average resource monitor processing time T_{mon} of all OSC agents in use. The average resource monitor processing time is the time the monitor action routine of an OSC agent typically takes to evaluate the status of a single resource.
- An OSC agent can handle resources in parallel. The maximum number of resources that can be processed in parallel is defined by the OSC agent's attribute **AgentMaxAction**. Thus, the minimum time T_{int} an OSC agent takes for initial status evaluation of all its managed resources (R_{total}) is:

  $T_{int} = T_{mon} * (R_{total} / AgentMaxAction + 1)$
- Use the maximum value of T_{int} of all OSC agents to calculate the minimum value of **OscCtrlStartupWait**.

It is best practice to calculate the **OscCtrlStartupWait** value according to the equation:

OscCtrlStartupWait = MAX (MAX ($T_{int,i}$) * 2, 30)

i … all OSC agents

6. Reconnection interval – attribute **OscCtrlReconnInterval**?

   When the OSC master control process detects that an OSC cluster member has left the OSC cluster (crash, SCS link lost) it waits OSC reconnection interval to see whether or not the OSC node re-joins the OSC cluster. If after the OSC reconnection interval has expired and the OSC node has not re-joined the OSC cluster, the absent OSC node is declared faulted and the OSC master control process initiates service group failover processing. This mechanism is similar to the one used with the cluster reconnection interval of OpenVMS.

   It is recommended that the default value (30 seconds) is not modified.

7. Automatic Quorum Adjustment – attribute **OscCtrlAutoAdjustQuorum**?

   Define whether or not OSC quorum will be automatically adjusted if an OSC cluster member unexpectedly leaves the OSC/OpenVMS cluster (crash, SCS link lost). VSI recommends using this feature only if the OpenVMS cluster OSC is installed on, contains a quorum disk instead of a quorum system. A quorum disk cannot be configured in

OSC. Thus, in order to avoid that OSC activity is blocked, although the OpenVMS cluster is still valid, enable this feature.

8.4 From the configuration plan to the OSC configuration

Once you have completed the OSC configuration planning and your OSC configuration plan has passed all checks listed in the checklists described in the previous section you can start to create a new OSC configuration using the OSC$CFG utility.

The aim of this section is to provide you with a best practice OSC$CFG configuration workflow guideline, but not to describe the command syntax of the OSC$CFG commands. Please refer to the OSC$CFG utility online help for the detailed command syntax description.

Note

Use quotation marks for case sensitive string inputs. If you omit quotation marks string inputs will be converted to upper case.

---

1. Create a new working configuration database

   OSC$CFG> CREATE DATABASE/VERSION=major-ID.minor-ID

   Each working OSC configuration database requires a unique version number. If you do not define the version number using the /VERSION qualifier the OSC$CFG utility automatically assigns a unique version number to the new working OSC configuration database.

2. Configure the OSC cluster

   OSC$CFG> MODIFY CLUSTER

   When a new OSC configuration database is created the OSC cluster configuration data structures are automatically added to the OSC configuration database with default values. Thus, the MODIFY CLUSTER command has to be used to define the OSC cluster attributes according to your configuration plan. The MODIFY CLUSTER command invokes the OSC cluster configuration wizard.

3. Define all OSC service groups according to your configuration plan

   OSC$CFG> ADD SRVGRP *service-group-name* [/ADVANCED]

   The ADD SRVGRP starts the service group configuration wizard that prompts you define the service group attributes. Omit the /ADVANCED qualifier if you do not want to modify the default values

of the optional service group attributes. For detailed information about the optional service group attributes please refer to A.9.2 Optional OSC service group attributes.

It is best practice not to configure the service group exclusion list (**SrvGrpDisAllow** attribute) at this point, but to use the DEFINE SRVGRP command after all resources, services and service groups have been added to the OSC configuration database (see 10).

4. Define all OSC services according to your configuration plan.

   OSC$CFG> ADD SERVICE *service-name*

   The ADD SERVICE starts the service configuration wizard that prompts you define the service attributes. It is best practice not to configure the service group membership (**ServiceGrpMember** attribute) and the service dependency (**ServiceDependency** attribute) at this point, but to use the DEFINE SERVICE command after all resources, services and service groups have been added to the OSC configuration database (see 7 and 8).

5. Define all OSC resources according to your configuration plan

   OSC$CFG> ADD RESOURCE *resource-type::resource-name[@node-name]* [/ADVANCED]

   The ADD RESOURCE starts the resource configuration wizard that prompts you define the resource type specific and common resource attributes. It is best practice not to configure the service membership (**ServiceMember** attribute) and the resource dependency (**ResourceDependency** attribute) at this point, but to use the DEFINE RESOURCE command after all OSC resources, OSC services and OSC service groups have been added to the OSC configuration database (see 7 and 8).

   To configure a node specific OSC resource enter the OSC node name when the resource configuration wizard prompts you to define the value of the **ResourceNode** attribute. In this case the resource definition is only valid for the nodes defined by the **ResourceNode** attribute. In order to configure the same resource for another OSC cluster member, recall the ADD RESOURCE command (use the same resource name) proceed as described above. Repeat this configuration loop for a node specific resource until the resource has been configured for all the planned OSC nodes.

The node specific resource definition is valid for and can be directly addressed from the command line by adding the node name with the @ prefix to the resource name parameter.

If the resource definition is valid for all OSC cluster members, enter the asterisk (*) character when the resource configuration wizard prompts you to define the value of the **ResourceNode** attribute. Do not add the string '@*' to the resource name parameter at the command line if you configure a resource valid for all OSC cluster members.

Apply the /ADVANCED qualifier if either the resource type (**On-Off**, **On-Only**, **Persistent**) of a resource differs from the resource type defined by the managing agent or the action routines and/or the optional timeout and fault management attributes have to be re-defined resource specific.

6. Define the OSC resource dependencies

   OSC$CFG> DEFINE RESOURCE *resource-type::resource-name*
   /CHILD=(*child-resource-list)*

   Define the resource dependencies as defined by your resource dependency diagrams. The *resource-type::resource-name* parameter contains all resources as a comma separated list that depends on the child resources defined by the /CHILD qualifier. Assign the child resources as a comma separated list to the /CHILD qualifier.

   Full wildcard support is provided for the *resource-type::resource-name* parameter list and the child resource list. You can place the asterisk (*) and percentage (%) wildcard character anywhere within each item of these resource lists.

7. Define the OSC service dependencies

   OSC$CFG> DEFINE SERVICE *service*/CHILD=(*service-list)*

   Define the service dependencies (if any) as defined in your service dependency diagrams. The *service-name* parameter contains all services as a comma separated list that depends on the child services defined by the /CHILD qualifier. Assign the child services as a comma separated list to the /CHILD qualifier. Full wildcard support is provided for the *service-name* parameter list and the child service list. You can

place the asterisk (*) and percentage (%) wildcard character anywhere within each item of these service lists.

8. Add all top level resources of your resource trees to the appropriate services as defined by your configuration plan

   OSC$CFG> DEFINE RESOURCE *resource-type::resource-name*
   /MEMBERSHIP = *(service-list)*

   The DEFINE RESOURCE/MEMBERSHIP command assigns the resources defined by the *resource-type::resource-name* parameter and all its child resources to a service. Thus, if the *resource-type::resource-name* parameter contains all top level resources of a service the whole resource tree is automatically assigned to the service defined by the /MEMBERSHIP qualifier.

9. Add all top level services of your service trees to the appropriate service groups as defined by your configuration plan.

   OSC$CFG> DEFINE SERVICE *service-name* /MEMBERSHIP=*(service-group-list)*

   The DEFINE SERVICE/MEMBERSHIP command assigns the services defined by the *service-name* parameter and all its child services to a service group. Thus, if the *service-name* parameter contains all top level services of a service group the whole service tree is automatically assigned to the service group defined by the /MEMBERSHIP qualifier.

10. Define the service groups that are not allowed to run on the same OSC node (service group exclusion list – see also 5.3.2 Controlling Service Group Concurrent Execution)

    OSC$CFG> DEFINE SRVGRP *srvgrp-name*/DISALLOW=*(service-group-list)*

    The DEFINE SRVGRP/DISALLOW command defines the service groups that are not allowed to run on the same OSC node. The /DISALLOW qualifier specifies the service groups as a comma separated list that are not allowed to run concurrently on the same OSC nodes as the service groups defined by the srvgrp-name parameter (also assigned as a comma separated list).

11. Verify the configuration

    OSC$CFG> VERIFY DATABASE

This commands checks if the current working OSC configuration database contains a valid configuration according to the OSC configuration rules. If the database verification check detects any configuration issue, a description of the problem is displayed on the screen. Three configuration severities exist:

- Minor
  These issues can be handled by OSC
- Major
  These issues can also be handled by OSC, but it is very likely that the configuration does not reflect the intention of the user (i.e. a service group is configured to run on a particular node, but some of its resources are not valid for this node).
- Error
  These issues cannot be handled by OSC. The OSC database is inconsistent. An inconsistent database cannot be activated as the default OSC configuration database.

If the database verification check detects no configuration issues, use the SHOW SRVGRP/LAYOUT command to verify that the service and resource dependencies configured match the resource and service dependency diagrams.

12. Configure the OSC event notification service

    Use the MODIFY EVENT command of the OSC$CFG utility to modify the OSC event classes so that they meet your requirements. For detailed information about OSC event classes please refer to section 6 OSC event notification.

13. Define the working OSC database as the default OSC database

    If the database verification and dependency check was successful you can now define the new OSC configuration database as the default configuration database. All OSC components fetch the OSC configuration from the default configuration database. To define a working OSC configuration database as the default configuration database, OSC has to be shutdown cluster-wide (see section 7.2.10.2 OSC cluster shutdown). Afterwards use the OSC$CFG command

    ```
    OSC$CFG> ACTIVATE DEFAULT
    ```

    to activate the working OSC configuration database as the new default configuration database and restart OSC (see section 7.1 How to start OSC).

8.5 Configuration example

This section provides a configuration example of how to configure OSC to manage two Oracle 10 databases – ORA1 and ORA2 - on a three node cluster.

As shown in Fig. 8.1 the cluster consists of the cluster members VMSS1, VMSS2 and VMSQ. VMSQ is the quorum system of the OpenVMS cluster.

Oracle 10 is installed on the cluster common shadow set DSA10 (Label ORA_SYSDSK). The database and redo files of ORA1 are located on DSA20 (ORA1_DSK) and DSA21 (ORA1_REDO). The database and redo files of ORA2 are located on DS30 (ORA2_DSK) and DSA31 (ORA2_REDO). Tables 8.3 and 8.4 list the shadow set assignment of the Oracle databases in detail.

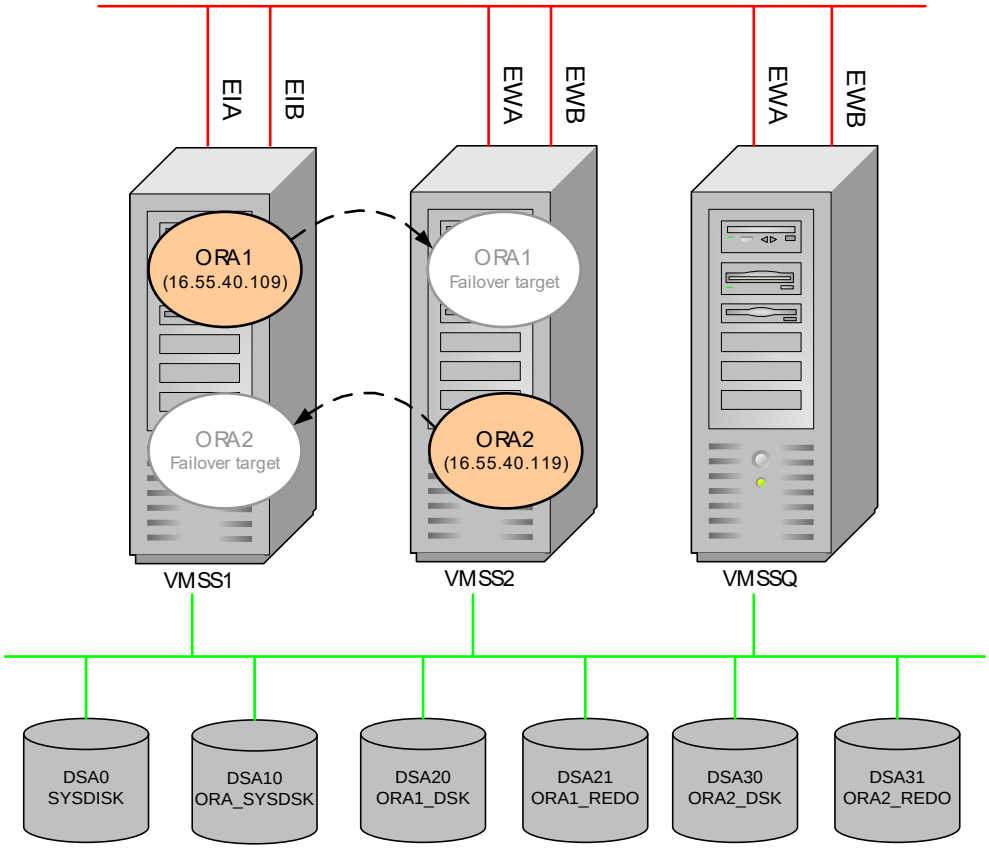


Fig. 8.1: Setup of a three node OpenVMS cluster consisting of node VMSS1, VMSS2 and VMSQ. OSC will be used to manage the Oracle 10 databases ORA and ORA2.

Primary execution node of ORA1 is VMSS1 and VMSS2 is the failover node.
Primary execution node of ORA2 is VMSS2 and VMSS1 is the failover target.

The values of the VOTES and EXPECTED_VOTES system parameters are listed in Table 8.1.

Tab. 8.1 VOTES and EXPECTED_VOTES system parameters of the cluster members

Node	VOTES	EXPECTED_VOTES
VMSS1	1	3
VMSS2	1	3
VMSQ	1	3

Each of the OSC cluster members has 2 network adapters installed, but they are different on the cluster members (see Tab. 8.1).

Tab. 8.2 Network cards installed on the cluster members

Node	NIC	Device name	IP interface
VMSS1	DE602	EIA	IE0
	DE602	EIB	IE1
VMSS2	DE500	EWA	WE0
	DE500	EWB	WE1
VMSQ	DE500	EWA	WE0
	DE500	EWB	WE1

Tab. 8.3 Shadow sets accessed by ORA1

Shadow Set	Label	Member 1	Member 2
DSA10	ORA_SYSDSK	1DGA110	1DGA210
DSA20	ORA1_DSK	1DGA120	1DGA220
DSA21	ORA1_REDO	1DGA121	1DGA221

Tab. 8.4 Shadow sets accessed by ORA2

Shadow Set	Label	Member 1	Member 2
DSA10	ORA_SYSDSK	1DGA110	1DGA210
DSA30	ORA2_DSK	1DGA130	1DGA230
DSA31	ORA2_REDO	1DGA131	1DGA231

As shown in Fig. 8.1 the preferred execution node of ORA1 is VMSS1 and the preferred execution node of ORA2 is VMSS2. The failover execution node or ORA1 is VMSS2 and the failover execution node for ORA2 is VMSS1.

Each of the execution nodes – VMSS1 and VMSS2 – provide sufficient system resources to run both Oracle databases.

The service IP address to access ORA1 and ORA2 are listed in Tab. 8.5.

Tab. 8.5 Service IP addresses to access ORA1 and ORA2

Oracle DB	IP address	FQDN
ORA1	10.10.40.109	ORA1.AUT.VSI.COM
ORA2	10.10.40.119	ORA2.AUT.VSI.COM

The failSAFE IP service of HP TCP/IP Services is enabled on all cluster members. It will be used to manage service IP address failover if one of the installed network adapters fails.

8.5.1 OSC configuration planning

Now we can start planning the OSC configuration according to the check list described in the previous sections.

8.5.1.1 Service Group, Service and Resource planning

1. Define all services (your applications) that will be managed by OSC and assign unique names for all these OSC services.

OSC will manage two Oracle database services – ORA1 and ORA2. Table 8.6 lists the resources required to run these services.

Tab. 8.6 ORA1 und ORA2 resource list

OSC service	Required resources
ORA1	ORA1 listener process ORA1 database open DSA10 DSA20 DSA21 Service IP address: 10.10.40.109 failSAFE IP service of TCP/IP
ORA2	ORA2 listener process ORA2 database open DSA10 DSA30 DSA31 Service IP address: 10.10.40.119 failSAFE IP service of TCPIP

2. Create resource dependency diagrams for all OSC services.

   The Oracle databases can be started if the shadow sets are mounted. Thus, the Oracle database resource depends on the shadow sets, but not on the listener process, nor on the availability of the service IP address. The listener process depends on the availability of the database and the service IP address. To assign a service IP address to the IP interfaces the failSAFE IP service of TCP/IP has to be started (we want to use failSAFE IP for interface failover management on a node).

   Fig. 8.2 and Fig. 8.3 show the service dependency diagrams of the OSC services ORA1and ORA2.

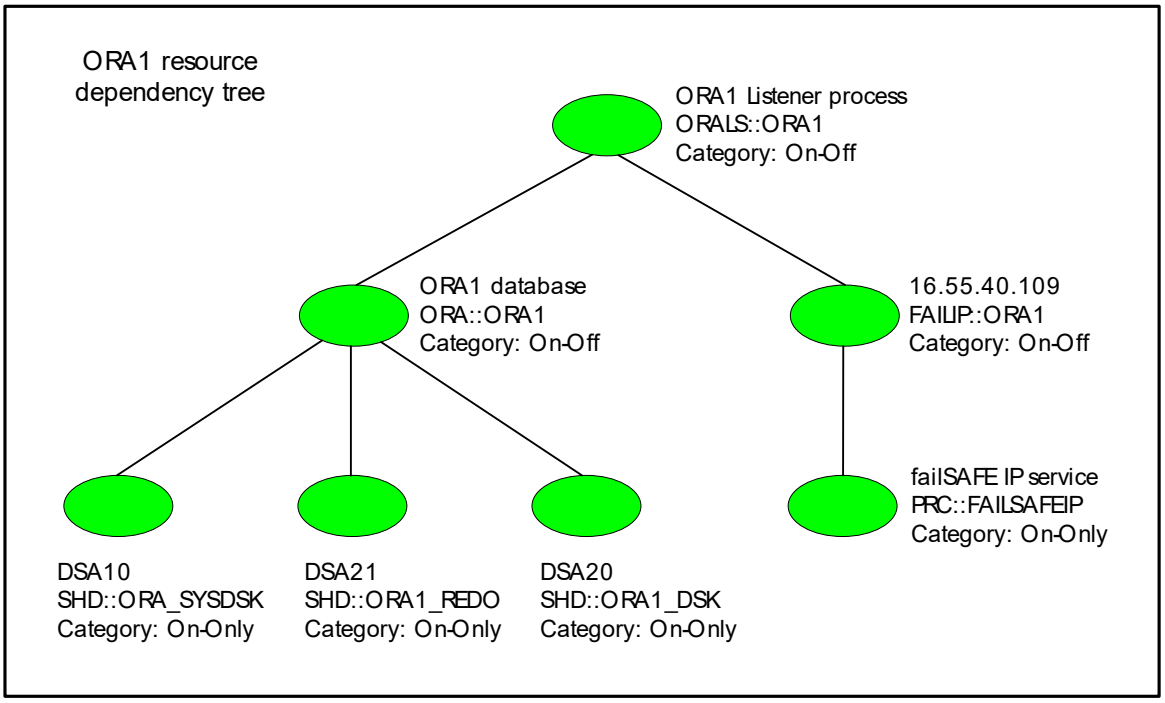


Fig. 8.2: Resource dependency tree of the Oracle database service ORA1.

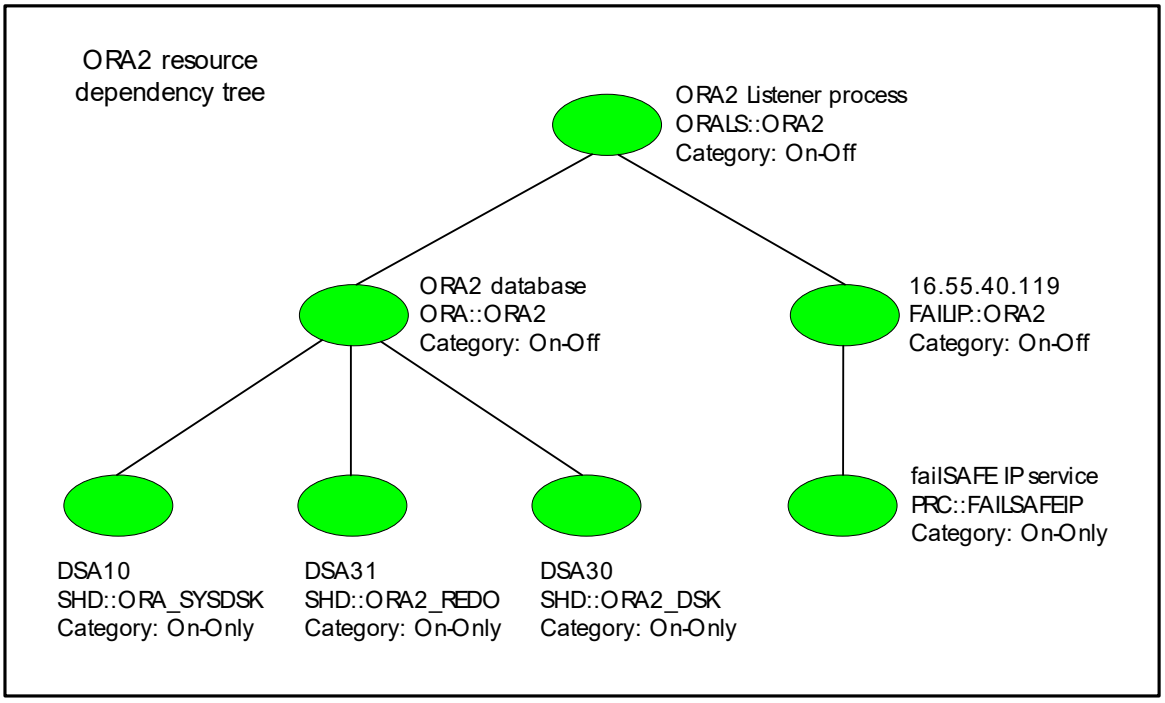


Fig. 8.3: Resource dependency tree of the Oracle database service ORA2.

3. Check if all OSC agents required to manage the different OSC resource types exist.

All resources can be managed by OSC agents bundled with VSI OpenVMS ServiceControl (see section 10 Bundled OSC agents)

4. Check if the OSC resources listed in the resource dependency diagram can be managed independently.

   Only OSC agents bundled with VSI OpenVMS service control are used. These agents fulfill this condition.

5. Assign unique OSC resources names.

   An OSC resource has to be defined according to the formatting rule shown below:

   Resource-type::resource-name[@node-name]

   The resource type is predefined by the OSC agent that manages a resource. The following resource names will be used.

Resource	OSC agent	OSC resource name
ORA1 listener process	OscAgtORALS	ORALS::ORA1
OR2 listener process	OscAgtORALS	ORALS::ORA2
ORA1 database	OscAgtORA	ORA::ORA1
ORA2 database	OscAgtORA	ORA::ORA2
10.10.40.109	OscAgtFailIP	FAILIP::ORA1
10.10.40.119	OscAgtFailIP	FAILIP::ORA1
failSAFE IP service	OscAgtPRC	PRC::FAILSAFEIP
DSA10	OscAgtSHD	SHD::ORA_SYSDSK
DSA20	OscAgtSHD	SHD::ORA1_DSK
DSA21	OscAgtSHD	SHD::ORA1_REDO
DSA30	OscAgtSHD	SHD::ORA2_DSK
DSA31	OscAgtSHD	SHD::ORA2_REDO

6. Check if all resources fulfill the resource type specific prerequisites so that they can be managed by the appropriate OSC agents.

   The OSC Oracle 10 agent (OscAgtORA) cannot manage databases that are configured to start on a dedicated node. Thus, we need to comment out the NODE property in the property files or ORA1 and ORA2.

   $ TYPE ORA_SYSDSK:[ORACLE10.DBS]sid_ORA1.properties

```
#-------------------------------------------------------
#  Database properties for instance SID=ORA1
#  Add any additional logical names you want defined
#  when executing: @<ORACLE_HOME>orauser <sid>
#  DO NOT USE QUOTES!
#-------------------------------------------------------
#NODE                = VMSS1
ORA_LOCAL_DATABASE = ORA1
ORA_DB              = ORA_SYSDSK:[oracle10.oradata.ORA1]
ORA_INSTANCE        = ORA_SYSDSK:[oracle10.oradata.ORA1]
ORA_DUMP            = ORA_SYSDSK:[oracle10.admin.ORA1.udump]
ORA_SNAP_CONTROL    = ORA_SYSDSK:[oracle10.oradata.ORA1]SNAPCF_ORA1.F
ORA_ARCHIVE         = ORA_SYSDSK:[oracle10.oradata.ORA1]
# ORA_CONTROL1      = ORA_SYSDSK:[oracle10.oradata.ORA1]CONTROL01.CTL
# ORA_CONTROL2      = ORA_SYSDSK:[oracle10.oradata.ORA1]CONTROL02.CTL
```

   $ TYPE ORA_SYSDSK:[ORACLE10.DBS]sid_ORA2.properties

```
#-------------------------------------------------------
#  Database properties for instance SID=ORA2
#  Add any additional logical names you want defined
#  when executing: @<ORACLE_HOME>orauser <sid>
#  DO NOT USE QUOTES!
#-------------------------------------------------------
#NODE                = VMSS2
ORA_LOCAL_DATABASE = ORA2
ORA_DB              = ORA_SYSDSK:[oracle10.oradata.ORA2]
ORA_INSTANCE        = ORA_SYSDSK:[oracle10.oradata.ORA2]
ORA_DUMP            = ORA_SYSDSK:[oracle10.admin.ORA2.udump]
ORA_SNAP_CONTROL    = ORA_SYSDSK:[oracle10.oradata.ORA2]SNAPCF_ORA2.F
ORA_ARCHIVE         = ORA_SYSDSK:[oracle10.oradata.ORA2]
# ORA_CONTROL1      = ORA_SYSDSK:[oracle10.oradata.ORA2]CONTROL01.CTL
# ORA_CONTROL2      = ORA_SYSDSK:[oracle10.oradata.ORA2]CONTROL02.CTL
```

   The NODE property attribute is not set. Thus, both Oracle databases can be managed by the OSC Oracle 10 agent.

   ORA1 and ORA2 will be accessed via the service IP addresses defined in Tab. 8.5. Thus, we need to check the HOST parameters in the listener.ora file.

```
$ TYPE ORA_SYSDSK:[ORACLE10.NETWORK.ADMIN]listener.ora

# listener.ora Network Configuration File:
# DSA10:[oracle10.network.admin]/listener.ora
# Generated by Oracle configuration tools.

LSNR_ORA1 =
 (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = ora1.aut.vsi.com)(PORT = 1531))
 )

SID_LIST_LSNR_ORA1 =
 (SID_DESC =
  (SID_NAME = ORA1)
  (PROGRAM = ORA_SYSDSK:[oracle10.network.admin]ORASRV_NETV2_ORA1.COM)
  (ORACLE_HOME = dsa10:[oracle10])
 )

LSNR_ORA2 =
 (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = ora2.aut.vsi.com)(PORT = 1532))
 )

SID_LIST_LSNR_ORA2 =
 (SID_DESC =
  (SID_NAME = ORA2)
  (PROGRAM = ORA_SYSDSK:[oracle10.network.admin]ORASRV_NETV2_ORA2.COM)
  (ORACLE_HOME = dsa10:[oracle10])
 )
```

The HOST parameters refer to the service IP fully qualified host names (where `ora1.aut.vsi.com = 10.10.40.109` and `ora2.aut.vsi.com = 10.10.40.119`). Thus, the listener.ora file does not have to be modified.

For detailed information about the prerequisites to use the OSC agents bundled with VSI OpenVMS ServiceControl please refer to section 10 Bundled OSC agents.

7. Define the resource type specific attributes for all resources.

   Tab. 8.7 lists the resource type specific attributes of all resources.

Tab. 8.7 Resource type specific attributes of the resources in this example

Resource	Resource type specific attributes
ORALS::ORA1	OracleLSNRPrc: LSNR_ORA1* OracleLSNRPrcCnt: [*unchanged default*] OracleLSNRSid: ORA1 OracleHomeDir: ORASYS_DSK:[ORACLE10] MonitorDelayTime: [*unchanged default*]

ORALS::ORA2	OracleLSNRPrc: LSNR_ORA2* OracleLSNRPrcCnt: [*unchanged default*] OracleLSNRSid: ORA2 OracleHomeDir: ORASYS_DSK:[ORACLE10] MonitorDelayTime: [*unchanged default*]
ORA::ORA1	OracleSID: ORA1 OracleRootDir: ORASYS_DSK:[ORACLE10] OracleParaFile: [*blank – no P-file used*] OnlineMonitorDelay: [*unchanged default*] ScriptDebug: [*unchanged default*]
ORA::ORA1	OracleSID: ORA2 OracleRootDir: ORASYS_DSK:[ORACLE10] OracleParaFile: [*blank – no P-file used*] OnlineMonitorDelay: [*unchanged default*] ScriptDebug: [*unchanged default*]
FAILIP:ORA1@VMSS1	IpAddress: 10.10.40.109 IpInterfaces: IE0, IE1 IpNetMask: 255.255.255.0 IpBroadCast: 10.10.40.255
FAILIP:ORA1@VMSS2	IpAddress: 10.10.40.109 IpInterfaces: WE0, WE1 IpNetMask: 255.255.255.0 IpBroadCast: 10.10.40.255
FAILIP:ORA2@VMSS1	IpAddress: 10.10.40.119 IpInterfaces: IE0, IE1 IpNetMask: 255.255.255.0 IpBroadCast: 10.10.40.255
FAILIP:ORA2@VMSS2	IpAddress: 10.10.40.119 IpInterfaces: WE0, WE1 IpNetMask: 255.255.255.0 IpBroadCast: 10.10.40.255
PRC:FAILSAFEIP	ProcessList: TCPIP$FAILSAF* ProcessCount: [*unchanged default*]
SHD::ORA_SYSDSK	ShadName: DSA10 ShadMembers: 1DGA110, 1DGA210 VolumeLabel: ORA_SYSDSK FullMbrOnMount: [*unchanged default*] FullMbrOnMonitor: [*unchanged default*]
SHD::ORA1_DSK	ShadName: DSA20 ShadMembers: 1DGA120, 1DGA220

	VolumeLabel: ORA1_DSK FullMbrOnMount: [*unchanged default*] FullMbrOnMonitor: [*unchanged default*]
SHD::ORA1_REDO	ShadName: DSA21 ShadMembers: 1DGA121, 1DGA221 VolumeLabel: ORA1_REDO FullMbrOnMount: [*unchanged default*] FullMbrOnMonitor: [*unchanged default*]
SHD::ORA2_DSK	ShadName: DSA30 ShadMembers: 1DGA130, 1DGA230 VolumeLabel: ORA2_DSK FullMbrOnMount: [*unchanged default*] FullMbrOnMonitor: [*unchanged default*]
SHD::ORA2_REDO	ShadName: DSA21 ShadMembers: 1DGA131, 1DGA231 VolumeLabel: ORA2_REDO FullMbrOnMount: [*unchanged default*] FullMbrOnMonitor: [*unchanged default*]

8. Mark all node specific resources in the resource dependency diagram.

   The resource type specific attributes of FAILIP resources are the IP interfaces and the service IP address that will be assigned (see section 10.7 OscAgtFailIP – OSC failSAFE IP agent). The OSC cluster members have different network adapters installed, the IP interfaces are different (see Tab. 8.2).

9. Check if the resources in the resource dependency diagrams are defined according to the resource related OSC configuration rules listed in the previous section.

   a. An **On-Only** resource must not depend on **On-Off** resources.
   b. A **Persistent** resource must not depend on **On-Off** or **On-Only** resources
   c. **On-Off** resources must not be members of different OSC services

   All these conditions are fulfilled.

10. Check the resource categories of the resources in the resource dependency diagrams.

The resource category of a resource is typically defined by the managing OSC agent. However, the resource category can be re-defined at the resource level. If the resource category has to be re-defined, mark the appropriate resources in the resource dependency diagrams.

The TCP/IP failSAFE IP service will be managed by the OSC process agent. This agent is defined to manage **On-Off** resources (see 10.4 OscAgtPRC - OSC process agent). The TCP/IP failSAFE IP service TCP/IP will be started by OSC, but will not be stopped if one of the services (ORA1 or ORA2) fails over to another OSC cluster member. Thus, the resource PRC::FAILSAFEIP must be defined as an **On-Only** resource. The resource type of this resource has to be re-defined at the resource level.

11. Check if the managing agent should call resource specific action routines for particular resources.

    The resource PRC::FAILSAFEIP is defined as an **On-Only** resource managed by the OSC process agent (see 10.4 OscAgtPRC - OSC process agent). Since this agent provides no default online action routine the online action routine has to be defined at the resource level – in this case SYS$STARtUP:TCPIP$FAILSAFE_STARTUP.COM.

12. Evaluate the online, offline and clean action routine processing times under worst case conditions.

    From disaster tests in the past we know that the startup of ORA1 database may last up to 10 minutes due to database recovery operations and the startup of ORA2 may last up to 5 minutes. Thus, we use the following online timeout settings for the resources ORA::ORA1 and ORA::ORA2.

OSC resource	Online timeout settings
ORA::ORA1	OnlineTmo: 240 sec. OnlineTmoWaitLimit: 5
ORA::ORA2	OnlineTmo: 120 sec. OnlineTmoWaitLimit: 5

    Using these settings, OSC waits up to 20 minutes for the online action routine of ORA::ORA1 and up to 10 minutes for the online action routine of ORA::ORA2 to complete (for a detailed description of the

resource attributes listed above please refer to section 5.6 How OSC Handles Resource Faults and A.7 OSC resource attributes).

13. Define the service groups according to the OSC configuration rules.

    The services ORA1 and ORA2 can be managed independently. Thus, the services can be members of different service groups. We define the service groups ORA1 and ORA2. Service group ORA1 contains the service ORA1 and service group ORA2 contains the service ORA2.

14. Define the service group categories (**Failover**, **MultiInstance**, **Parallel**) and the execution node list for all service groups.

    The services of the service group are not cluster aware. Thus, both service groups are defined as **Failover** service groups.

    In this example the OSC failover policy will be defined as STATIC. LOAD_BALANCING OSC failover policy makes no sense if we are running only two service groups on a two OSC node cluster.

    The preferred execution node of ORA1 is VMSS1 and VMSS2 is the failover node. The preferred execution node of ORA2 is VMSS2 and VMSS1 is the failover node. Thus, both nodes are members of the execution node list of the service groups ORA1 and ORA2. Since OSC STATIC Failover policy will be defined the node selection criterion is the node priority assigned to OSC nodes in the execution node list for each service group. VMSS1 will be the preferred execution node for ORA1. Thus, the execution priority of VMSS1 has to be higher than the execution priority for VMSS2. For the service group ORA2 the execution priority for VMSS2 has to be higher than the execution priority for VMSS1.

OSC service group	Execution node list
ORA1	VMSS1:2,VMSS2:1
ORA2	VMSS1:1,VMSS2:2

15. The attribute **SrvGrpLoad** does not have to be defined since the OSC failover policy STATIC will be used.

16. Both service groups are allowed to run on the same node (the preferred OSC node for ORA1 is the failover node of ORA2 and vice versa).

17. Define the OSC event classes

    All events except the heartbeat event will be logged in the common OSC event message file and forwarded to all connected OSC consoles. Fatal and error event messages of the OSC event classes:
    - OSCCTRL_CNXMAN_EVT
    - OSCCTRL_STATE_EVT
    - OSCSRV_CNXMAN_EVT
    - OSCSRV_STATE_EVT
    - OSCAGT_STATE_EVT

    will also trigger the execution of a user defined script that sends an alert e-mail to system management. The OSC installation procedure provides the event script OSC$CFG:OSC_SENDMAIL.COM. This script sends mail if the severity of an event passed to the script is either ERROR or FATAL. Thus, this script will be used.

    For detailed information about OSC event notification, OSC event classes and user event scripts please refer to section 6 OSC event notification.

8.5.1.1 OSC cluster

1. OSC cluster name – attribute **OscCtrlClusterName**?

   We select "OSC-Test" for the OSC cluster name

2. OSC cluster members and votes – attribute **OscCtrlNodes**?

   We include all OpenVMS cluster member as being OSC cluster members and assign the same number of OSC votes to the OSC cluster member as defined by the VOTES system parameter (see Tab. 8.1)

3. OSC failover policy – attribute **OscCtrlFailoverPolicy**?

   As mentioned in the previous section the STATIC OSC failover policy will be configured.

4. Expected votes – attribute **OscCtrlExpVotes**?

In this scenario there is no reason to define a value that differs from the value already defined by the EXPECTED_VOTES system parameter.

5. Startup wait and state transition wait – attribute **OscCtrlStartupWait**?

   The state transition wait time must be greater than the time required to start all OSC components within the cluster plus the maximum initial resource monitor processing time of all OSC agents in use. The default value of this attribute (60 seconds) fulfills this condition.

6. Reconnection interval – attribute **OscCtrlReconnInterval**?

   We use the default value.

7. Automatic quorum adjustment – attribute **OscCtrlAutoAdjustQuorum**?

   We use the default value = No.

Tab. 8.8 summarizes the values of the OSC cluster attributes.

Tab. 8.8 OSC cluster attributes used for this configuration

OSC cluster attribute	Value
OscCtrlClusterName	"OSC-Test"
OscCtrlNode	VMSS1:1, VMSS2:1, VMSQ:1
OscCtrlFailoverPolicy	STATIC
OscCtrlProcPriority	[Unchanged default]
OscCtrlReconnInterval	[Unchanged default]
OscCtrlStartupWait	[Unchanged default]
OscCtrlStartupWait	[Unchanged default]
OscCtrlExpVote	3
OscCtrlAutoAdjustQuorum	No

8.5.2 OSC configuration

During the planning phase all the required information was gathered and we have checked if all the prerequisites to manage the Oracle 10 databases ORA1 and ORA2 services by OSC are fulfilled. Now we can start to configure OSC using the OSC$CFG utility.

$ RUN OSC$BIN:OSC$CFG

1. We create a new working OSC configuration database

   OSC$CFG> CREATE DATABASE/V=1.0
   OSC$CFG-I-SUCCESS, CFG database 'OSC$COMMON:[CFG]OSC$CONFIG_64.DAT;1'
   Version: V1.0 is the new working CFG database

2. We configure the OSC cluster

   OSC$CFG> MODIFY CLUSTER

```
Welcome to the OSC cluster configuration wizard
-----------------------------------------------
Dscr: OscCtrl OSC cluster name
Attr: {OscCtrlClusterName} []: "OSC-Test"
Dscr: OscCtrl node list
Attr: {OscCtrlNode} []: VMSS1:1,VMSS2:1,VMSQ:1
Dscr: OscCtrl SrvGrp failover policy (Static | Load-Balancing)
Attr: {OscCtrlFailoverPolicy} [Static]:
Dscr: OscCtrl process priority
Attr: {OscCtrlProcPriority} [10]:
Dscr: OscCtrl reconnect interval
Attr: {OscCtrlReconnInterval} [30 sec]:
Dscr: OscCtrl time to wait for primary
Attr: {OscCtrlStartupWait} [60 sec]:
Dscr: OscCtrl expected votes
Attr: {OscCtrlExpVotes} [1]: 3
Dscr: OscCtrl auto-adjust quorum when a node is removed from OSC
Attr: {OscCtrlAutoAjustQuorum} [No]:
Dscr: OscCtrl Simulation Mode
Attr: {OscCtrlSimulate} [No]:
```

   OSC$CFG-I-MODIFY, OSC cluster definitions have been updated in the current working CFG Database

3. We configure the OSC service groups ORA1 and ORA2

   OSC$CFG> ADD SRVGRP ORA1

```
Welcome to the SRVGROUP OSC configuration wizard
------------------------------------------------
Dscr: Service Group Category (Failover|MultiInstance|Parallel)
Attr: {SrvGrpType} [Failover]:
Dscr: Service Group description
```

```
Attr: {SrvGrpDescription} []: "ORA1 Oracle database SrvGrp"
Dscr: Service Group Node List
Attr: {SrvGrpNodes} []: VMSS1:2,VMSS2:1
```
OSC$CFG-I-ADD, item 'ORA1' has been added to the working CFG database

OSC$CFG> ADD SRVGRP ORA2

```
Welcome to the SRVGROUP OSC configuration wizard
------------------------------------------------
Dscr: Service Group Category (Failover|MultiInstance|Parallel)
Attr: {SrvGrpType} [Failover]:
Dscr: Service Group description
Attr: {SrvGrpDescription} []: "ORA2 Oracle database SrvGrp"
Dscr: Service Group Node List
Attr: {SrvGrpNodes} []: VMSS1:1,VMSS2:2
```

OSC$CFG-I-ADD, item 'ORA2' has been added to the working CFG database

4. We configure the OSC services ORA1 and ORA2

OSC$CFG> ADD SERVICE ORA1

```
Welcome to the SERVICE OSC configuration wizard
-----------------------------------------------
Dscr: Service description
Attr: {ServiceDescription} []: "ORA1 Oracle database service"
Dscr: Service is member of Service Group
Attr: {ServiceGrpMember} []:
Dscr: Service depends on
Attr: {ServiceDependency} []:
Dscr: Service Priority
Attr: {ServicePriority} [1]:
```

OSC$CFG-I-ADD, item 'ORA1' has been added to the working CFG database

OSC$CFG> ADD SERVICE ORA2

```
Welcome to the SERVICE OSC configuration wizard
-----------------------------------------------
Dscr: Service description
Attr: {ServiceDescription} []: "ORA2 Oracle database service"
Dscr: Service is member of Service Group
Attr: {ServiceGrpMember} []:
Dscr: Service depends on
Attr: {ServiceDependency} []:
Dscr: Service Priority
Attr: {ServicePriority} [1]:
```

OSC$CFG-I-ADD, item 'ORA2' has been added to the working CFG database

5. We configure all the required OSC resources

First we configure all the node independent resources. These are:

- SHD::ORA_SYSDSK
- SHD::ORA1_DSK
- SHD::ORA1_REDO
- SHD::ORA2_DSK
- SHD::ORA2_REDO
- ORALS::ORA1
- ORALS::ORA2
- ORA::ORA1
- ORA::ORA2
- PRC::FAILSAFEIP

We use the AUTOCONFIG SHD command to configure the shadow sets instead of configuring each shadow set manually. This command saves time and eliminates the risk of configuration errors. For more detailed information about the AUTOCONFIG SHD command and the command qualifiers please refer to the OSC$CFG online help.

The default online action routine timeout resource attributes have to be modified for:

- ORA::ORA1
- ORA::ORA2.

The resource type and the online action routine resource attributes have to be defined resource specifically for:

- PRC::FAILSAFEIP

These resource attributes are optional resource attributes (see A.7.2 Optional common OSC resource attributes). Thus, we have to apply the /ADVANCED qualifier when we configure the resources:

- ORA::ORA1
- ORA::ORA2.
- PRC::FAILSAFEIP

```
OSC$CFG> AUTOCONFIG SHD
                    /SELECT=(DSA10,DSA20,DSA21,DSA30,DSA31
                    )
                    /NOCFG_DISK
OSC$CFG-I-AUTOCFG, successfully configured OSC resource SHD::ORA_SYSDSK for
shadow set DSA10
            Shadow Members: $1$DGA110
            Shadow Members: $1$DGA210
OSC$CFG-I-AUTOCFG, successfully configured OSC resource SHD::ORA1_DSK for
shadow set DSA20
            Shadow Members: $1$DGA120
            Shadow Members: $1$DGA220
OSC$CFG-I-AUTOCFG, successfully configured OSC resource SHD::ORA1_REDO for
shadow set DSA21
```

Shadow Members: 1DGA121
Shadow Members: 1DGA221
SHD::ORA2_DSK for shadow set DSA30
Shadow Members: 1DGA130
Shadow Members: 1DGA230
OSC$CFG-I-AUTOCFG, successfully configured OSC resource SHD::ORA2_REDO for shadow set DSA31
Shadow Members: 1DGA131
Shadow Members: 1DGA231

OSC$CFG> ADD RESOURCE ORALS::ORA1

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [*]:
Dscr: Resource Description
Attr: {ResourceDescription} []: "ORA1 listener resource"
Dscr: Oracle Listener Process Name
Attr: {OracleLSNRPrc} []: LSNR_ORA1*
Dscr: Number of active Oracle Listener processes
Attr: {OracleLSNRPrcCnt} [1]:
Dscr: Oracle SID
Attr: {OracleLSNRSid} []: ORA1
Dscr: Oracle Home directory
Attr: {OracleHomeDir} []:ORA_SYSDSK:[ORACLE10]
Dscr: Delay in seconds before MONITOR is called
                                        after OFFLINE-ONLINE
Attr: {MonitorDelayTime} [9]:
Dscr: Resource is member of service (s)
Attr: {ServiceMember} []:
Dscr: Resource dependency
Attr: {ResourceDependency} []:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList}      [OracleLSNRPrc,OracleLSNRPrcCnt,
                     OracleLSNRSid,OracleHomeDir,
                     MonitorDelayTime]:
```

OSC$CFG-I-ADD, RESOURCE 'ORALS::ORA1' has been added to the working CFG database.

OSC$CFG> ADD RESOURCE ORALS::ORA2

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [*]:
Dscr: Resource Description
Attr: {ResourceDescription} []: "ORA2 listener resource"
Dscr: Oracle Listener Process Name
Attr: {OracleLSNRPrc} []: LSNR_ORA2*
Dscr: Number of active Oracle Listener processes
Attr: {OracleLSNRPrcCnt} [1]:
Dscr: Oracle SID
Attr: {OracleLSNRSid} []: ORA2
Dscr: Oracle Home directory
```

```
Attr: {OracleHomeDir} []:ORA_SYSDSK:[ORACLE10]
Dscr: Delay in seconds before MONITOR is called
                                        after OFFLINE-ONLINE
Attr: {MonitorDelayTime} [9]:
Dscr: Resource is member of service (s)
Attr: {ServiceMember} []:
Dscr: Resource dependency
Attr: {ResourceDependency} []:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList}      [OracleLSNRPrc,OracleLSNRPrcCnt,
                     OracleLSNRSid,OracleHomeDir,
                     MonitorDelayTime]:
```

OSC$CFG-I-ADD, RESOURCE 'ORALS::ORA2' has been added to the working CFG database.

OSC$CFG> ADD RESOURCE ORA::ORA1/ADVANCED

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [*]:
Dscr: Resource Description
Attr: {ResourceDescription} []: "ORA1 database resource"
Dscr: Resource Category (Persistent | On-Only| On-Off)
Attr: {ResourceCategory} []:
Dscr: Oracle System ID
Attr: {OracleSID} []: ORA1
Dscr: Oracle root directory
Attr: {OracleRootDir} []:ORA_SYSDSK:[ORACLE10]
Dscr: Orcale startup parameter file
Attr: {OracleParaFile} []:
Dscr: Delay to start MONITOR script after ONLINE script
      completed
Attr: {OnlineMonitorDelay} [5]:
Dscr: Call action scripts in debug mode
Attr: {ScriptDebug} [No]:
Dscr: Resource is member of service (s)
Attr: {ServiceMember} []:
Dscr: Resource dependency
Attr: {ResourceDependency} []:
Dscr: Resource is critical
Attr: {Critical} [Yes]:
Dscr: Resource Enabled
Attr: {Enabled} [Yes]:
Dscr: Online Monitor interval
Attr: {OnlineMonitorInterval} [60 sec]:
Dscr: Offline Monitor interval
Attr: {OfflineMonitorInterval} [120 sec]:
Dscr: Monitor OFFLINE tolerance limit
Attr: {ToleranceLimit} [0]:
Dscr: Resource is a fault candidate if the monitor
      routine times out
Attr: {FaultOnMonitorTmo} [Yes]:
Dscr: Number of monitor timeouts to fault resource
Attr: {FaultOnMonitorTmoLimit} [4]:
Dscr: Do not manage faults even if SrvGrp is set to
```

```
      manage faults
Attr: {DisableMangeFault} [No]:
Dscr: Online retry limit
Attr: {OnlineRetryLimit} [2]:
Dscr: Online wait limit
Attr: {OnlineWaitLimit} [2]:
Dscr: Online timeout wait limit
Attr: {OnlineTmoWaitLimit} [2]: 5
Dscr: Offline wait limit
Attr: {OfflineWaitLimit} [2]:
Dscr: Offline timeout wait limit
Attr: {OfflineTmoWaitLimit} [2]:
Dscr: Resource restart limit
Attr: {RestartLimit} [1]:
Dscr: Clean entry retry limit
Attr: {CleanRetryLimit} [5]:
Dscr: Timeout retry limit for action entries
Attr: {TimeOutRetryLimit} [5]:
Dscr: Confidential Limit
Attr: {ConfLimit} [600]:
Dscr: Monitor Script
Attr: {MonitorScript} [OSC$COMMON:[CFG.ORA]ORACLE_MONITOR.COM]:
Dscr: Monitor Script timeout
Attr: {MonitorTmo} [45]:
Dscr: Online Script
Attr: {OnlineScript} [OSC$COMMON:[CFG.ORA]ORACLE_ONLINE.COM]:
Dscr: Online Script timeout
Attr: {OnlineTmo} [180]: 240
Dscr: Offline Script
Attr: {OfflineScript} [OSC$COMMON:[CFG.ORA]ORACLE_OFFLINE.COM]:
Dscr: Offline Script timeout
Attr: {OfflineTmo} [180]:
Dscr: Cleanup Script
Attr: {CleanScript} [OSC$COMMON:[CFG.ORA]ORACLE_CLEAN.COM]:
Dscr: Clean Script timeout
Attr: {CleanTmo} [180]:
Dscr: Open Script
Attr: {OpenScript} []:
Dscr: Open Script timeout
Attr: {OpenTmo} [60]:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList}      [OracleSID,OracleRootDir,
                     OracleParaFile,OnlineMonitorDelay,
                     ScriptDebug]:
Dscr: Pass function code in P1 to action scripts
Attr: {PassFuncCode} [No]:
```

OSC$CFG-I-ADD, RESOURCE 'ORA::ORA1' has been added to the working CFG database.

OSC$CFG> ADD RESOURCE ORA::ORA2/ADVANCED

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [*]:
```

```
Dscr: Resource Description
Attr: {ResourceDescription} []: "ORA2 database resource"
Dscr: Resource Category (Persistent | On-Only| On-Off)
Attr: {ResourceCategory} []:
Dscr: Oracle System ID
Attr: {OracleSID} []: ORA2
Dscr: Oracle root directory
Attr: {OracleRootDir} []:ORA_SYSDSK:[ORACLE10]
Dscr: Orcale startup parameter file
Attr: {OracleParaFile} []:
Dscr: Delay to start MONITOR script after ONLINE script
      completed
Attr: {OnlineMonitorDelay} [5]:
Dscr: Call action scripts in debug mode
Attr: {ScriptDebug} [No]:
Dscr: Resource is member of service (s)
Attr: {ServiceMember} []:
Dscr: Resource dependency
Attr: {ResourceDependency} []:
Dscr: Resource is critical
Attr: {Critical} [Yes]:
Dscr: Resource Enabled
Attr: {Enabled} [Yes]:
Dscr: Online Monitor interval
Attr: {OnlineMonitorInterval} [60 sec]:
Dscr: Offline Monitor interval
Attr: {OfflineMonitorInterval} [120 sec]:
Dscr: Monitor OFFLINE tolerance limit
Attr: {ToleranceLimit} [0]:
Dscr: Resource is a fault candidate if the monitor
      routine times out
Attr: {FaultOnMonitorTmo} [Yes]:
Dscr: Number of monitor timeouts to fault resource
Attr: {FaultOnMonitorTmoLimit} [4]:
Dscr: Do not manage faults even if SrvGrp is set to
      manage faults
Attr: {DisableMangeFault} [No]:
Dscr: Online retry limit
Attr: {OnlineRetryLimit} [2]:
Dscr: Online wait limit
Attr: {OnlineWaitLimit} [2]:
Dscr: Online timeout wait limit
Attr: {OnlineTmoWaitLimit} [2]: 5
Dscr: Offline wait limit
Attr: {OfflineWaitLimit} [2]:
Dscr: Offline timeout wait limit
Attr: {OfflineTmoWaitLimit} [2]:
Dscr: Resource restart limit
Attr: {RestartLimit} [1]:
Dscr: Clean entry retry limit
Attr: {CleanRetryLimit} [5]:
Dscr: Timeout retry limit for action entries
Attr: {TimeOutRetryLimit} [5]:
Dscr: Confidential Limit
Attr: {ConfLimit} [600]:
Dscr: Monitor Script
Attr: {MonitorScript} [OSC$COMMON:[CFG.ORA]ORACLE_MONITOR.COM]:
```

```
Dscr: Monitor Script timeout
Attr: {MonitorTmo} [45]:
Dscr: Online Script
Attr: {OnlineScript} [OSC$COMMON:[CFG.ORA]ORACLE_ONLINE.COM]:
Dscr: Online Script timeout
Attr: {OnlineTmo} [180]: 120
Dscr: Offline Script
Attr: {OfflineScript} [OSC$COMMON:[CFG.ORA]ORACLE_OFFLINE.COM]:
Dscr: Offline Script timeout
Attr: {OfflineTmo} [180]:
Dscr: Cleanup Script
Attr: {CleanScript} [OSC$COMMON:[CFG.ORA]ORACLE_CLEAN.COM]:
Dscr: Clean Script timeout
Attr: {CleanTmo} [180]:
Dscr: Open Script
Attr: {OpenScript} []:
Dscr: Open Script timeout
Attr: {OpenTmo} [60]:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList}      [OracleSID,OracleRootDir,
                     OracleParaFile,OnlineMonitorDelay,
                     ScriptDebug]:
Dscr: Pass function code in P1 to action scripts
Attr: {PassFuncCode} [No]:
```

OSC$CFG-I-ADD, RESOURCE 'ORA::ORA2' has been added to the working CFG database.

OSC$CFG> ADD RESOURCE PRC::FAILSAFEIP/ADVANCED

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [*]:
Dscr: Resource Description
Attr: {ResourceDescription} []: "fialSAFE IP service process"
Dscr: Resource Category (Persistent | On-Only| On-Off)
Attr: {ResourceCategory} []: On-Only
Dscr: Process list to monitor
Attr: {ProcessList} []: TCPIP$FAILSAF*
Dscr: Number of processes that have to exist
Attr: {ProcessCount} [1]:
Dscr: Resource is member of service (s)
Attr: {ServiceMember} []:
Dscr: Resource dependency
Attr: {ResourceDependency} []:
Dscr: Resource is critical
Attr: {Critical} [Yes]:
Dscr: Resource Enabled
Attr: {Enabled} [Yes]:
Dscr: Online Monitor interval
Attr: {OnlineMonitorInterval} [30 sec]:
Dscr: Offline Monitor interval
Attr: {OfflineMonitorInterval} [30 sec]:
Dscr: Monitor OFFLINE tolerance limit
Attr: {ToleranceLimit} [2]:
Dscr: Resource is a fault candidate if the monitor
```

```
      routine times out
Attr: {FaultOnMonitorTmo} [Yes]:
Dscr: Number of monitor timeouts to fault resource
Attr: {FaultOnMonitorTmoLimit} [4]:
Dscr: Do not manage faults even if SrvGrp is set to manage
      faults
Attr: {DisableMangeFault} [No]:
Dscr: Online retry limit
Attr: {OnlineRetryLimit} [0]:
Dscr: Online wait limit
Attr: {OnlineWaitLimit} [2]:
Dscr: Online timeout wait limit
Attr: {OnlineTmoWaitLimit} [2]:
Dscr: Offline wait limit
Attr: {OfflineWaitLimit} [2]:
Dscr: Offline timeout wait limit
Attr: {OfflineTmoWaitLimit} [2]:
Dscr: Resource restart limit
Attr: {RestartLimit} [0]:
Dscr: Clean entry retry limit
Attr: {CleanRetryLimit} [5]:
Dscr: Timeout retry limit for action entries
Attr: {TimeOutRetryLimit} [5]:
Dscr: Confidential Limit
Attr: {ConfLimit} [600]:
Dscr: Monitor Script
Attr: {MonitorScript} []:
Dscr: Monitor Script timeout
Attr: {MonitorTmo} [25]:
Dscr: Online Script
Attr: {OnlineScript} []: SYS$STARTUP:TCPIP$FAILSAFE_STARTUP.COM
Dscr: Online Script timeout
Attr: {OnlineTmo} [300]:
Dscr: Offline Script
Attr: {OfflineScript} []:
Dscr: Offline Script timeout
Attr: {OfflineTmo} [300]:
Dscr: Cleanup Script
Attr: {CleanScript} []:
Dscr: Clean Script timeout
Attr: {CleanTmo} [60]:
Dscr: Open Script
Attr: {OpenScript} []:
Dscr: Open Script timeout
Attr: {OpenTmo} [60]:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList} [ProcessList, ProcessCount]:
Dscr: Pass function code in P1 to action scripts
Attr: {PassFuncCode} [No]:
```

OSC$CFG-I-ADD, RESOURCE 'PRC::FAILSAFEIP' has been added to the working CFG database.

Finally we configure the node dependent resources:

- FAILIP::ORA1

- FAILIP::ORA2

In this example both methods to configure node specific resources are demonstrated – using the wizard (FAILIP::ORA1) and addressing the node from the command line (FAILIP::ORA2@*node-name*).

OSC$CFG> ADD RESOURCE FAILIP::ORA1

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [*]: VMSS1
Dscr: Resource Description
Attr: {ResourceDescription} []: "failSAFE IP service
                                 address for ORA1 on VMSS1"
Dscr: IP address to monitor
Attr: {IpAddress} []: 10.10.40.109
Dscr: FailSafe IP interfaces
Attr: {IpInterfaces} []: IE0,IE1
Dscr: IP Network mask
Attr: {IpNetMask} []: 255.255.255.0
Dscr: IP Broadcast address
Attr: {IpBroadCast} []: 10.10.40.255
Dscr: Resource is member of service (s)
Attr: {ServiceMember} []:
Dscr: Resource dependency
Attr: {ResourceDependency} []:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList} [IpAddress, IpInterfaces, IpNetMask,
IpBroadCast]:
```

OSC$CFG-I-ADD, RESOURCE 'FAILIP::ORA1@VMSS1' has been added to the working CFG database

OSC$CFG> ADD RESOURCE FAILIP::ORA1

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [*]: VMSS2
Dscr: Resource Description
Attr: {ResourceDescription} []: "failSAFE IP service
                                 address for ORA1 on VMSS2"
Dscr: IP address to monitor
Attr: {IpAddress} []: 10.10.40.109
Dscr: FailSafe IP interfaces
Attr: {IpInterfaces} []: WE0,WE1
Dscr: IP Network mask
Attr: {IpNetMask} []: 255.255.255.0
Dscr: IP Broadcast address
Attr: {IpBroadCast} []: 10.10.40.255
Dscr: Resource is member of service (s)
Attr: {ServiceMember} []:
Dscr: Resource dependency
```

```
Attr: {ResourceDependency} []:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList} [IpAddress, IpInterfaces, IpNetMask,
IpBroadCast]:
```

OSC$CFG-I-ADD, RESOURCE 'FAILIP::ORA1@VMSS2' has been added to the working CFG database

OSC$CFG> ADD RESOURCE FAILIP::ORA2@VMSS1

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [VMSS1]:
Dscr: Resource Description
Attr: {ResourceDescription} []: "failSAFE IP service
                                 address for ORA2 on VMSS1"
Dscr: IP address to monitor
Attr: {IpAddress} []: 10.10.40.119
Dscr: FailSafe IP interfaces
Attr: {IpInterfaces} []: IE0,IE1
Dscr: IP Network mask
Attr: {IpNetMask} []:255.255.255.0
Dscr: IP Broadcast address
Dscr: IP Broadcast address
Attr: {IpBroadCast} []: 10.10.40.255
Dscr: Resource is member of service (s)
Attr: {ServiceMember} []:
Dscr: Resource dependency
Attr: {ResourceDependency} []:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList} [IpAddress, IpInterfaces, IpNetMask,
IpBroadCast]:
```

OSC$CFG-I-ADD, RESOURCE 'FAILIP::ORA2@VMSS1@VMSS1' has been added to the working CFG database.

OSC$CFG> ADD RESOURCE FAILIP::ORA2@VMSS2

```
Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------
Dscr: Resource is valid for node
Attr: {ResourceNode} [VMSS2]:
Dscr: Resource Description
Attr: {ResourceDescription} []: "failSAFE IP service
                                 address for ORA2 on VMSS1"
Dscr: IP address to monitor
Attr: {IpAddress} []: 10.10.40.119
Dscr: FailSafe IP interfaces
Attr: {IpInterfaces} []: WE0,WE1
Dscr: IP Network mask
Attr: {IpNetMask} []: 255.255.255.0
Dscr: IP Broadcast address
Attr: {IpBroadCast} []: 10.10.40.255
Dscr: Resource is member of service (s)
```

```
Attr: {ServiceMember} []:
Dscr: Resource dependency
Attr: {ResourceDependency} []:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList} [IpAddress, IpInterfaces, IpNetMask,
IpBroadCast]:
```

OSC$CFG-I-ADD, RESOURCE 'FAILIP::ORA2@VMSS2@VMSS2' has been added to the working CFG database.

6. We configure all the resource dependencies according to the resource dependency diagrams (Fig. 8.2 and Fig. 8.3).

OSC$CFG> DEFINE RESOURCE ORA::ORA1
/CHILD=(SHD::ORA_SYSDSK,SHD::ORA1_*)
OSC$CFG-I-DEPEND, successfully added SHD::ORA1_DSK to dependency list of ORA::ORA1
OSC$CFG-I-DEPEND, successfully added SHD::ORA1_REDO to dependency list of ORA::ORA1
OSC$CFG-I-DEPEND, successfully added SHD::ORA_SYSDSK to dependency list of ORA::ORA1

OSC$CFG> DEFINE RESOURCE ORA::ORA2
/CHILD=(SHD::ORA_SYSDSK,SHD::ORA2_*)
OSC$CFG-I-DEPEND, successfully added SHD::ORA2_DSK to dependency list of ORA::ORA2
OSC$CFG-I-DEPEND, successfully added SHD::ORA2_REDO to dependency list of ORA::ORA2
OSC$CFG-I-DEPEND, successfully added SHD::ORA_SYSDSK to dependency list of ORA::ORA2

OSC$CFG> DEFINE RESOURCE FAILIP::ORA1,FAILIP::ORA2
/CHILD=PRC::FAILSAFEIP
OSC$CFG-I-DEPEND, successfully added PRC::FAILSAFEIP to dependency list of FAILIP::ORA1@VMSS1
OSC$CFG-I-DEPEND, successfully added PRC::FAILSAFEIP to dependency list of FAILIP::ORA1@VMSS2
OSC$CFG-I-DEPEND, successfully added PRC::FAILSAFEIP to dependency list of FAILIP::ORA2@VMSS1
OSC$CFG-I-DEPEND, successfully added PRC::FAILSAFEIP to dependency list of FAILIP::ORA2@VMSS2

OSC$CFG> DEFINE RESOURCE ORALS::ORA1
/CHILD=(ORA::ORA1,FAILIP::ORA1)
OSC$CFG-I-DEPEND, successfully added FAILIP::ORA1 to dependency list of ORALS::ORA1
OSC$CFG-I-DEPEND, successfully added ORA::ORA1 to dependency list of ORALS::ORA1

OSC$CFG> DEFINE RESOURCE ORALS::ORA2
/CHILD=(ORA::ORA2,FAILIP::ORA2)
OSC$CFG-I-DEPEND, successfully added FAILIP::ORA2 to dependency list of ORALS::ORA2
OSC$CFG-I-DEPEND, successfully added ORA::ORA2 to dependency list of ORALS::ORA2

7. No service dependencies exist

8. We configure the service membership of the top-level resources for both resource trees (see Fig. 8.2 and Fog. 8.3).

```
OSC$CFG> DEFINE RESOURCE ORALS::ORA1/MEMBER=ORA1
OSC$CFG-I-BIND, successfully added resource ORALS::ORA1 to service ORA1
OSC$CFG-I-BIND, successfully added resource FAILIP::ORA1@VMSS1 to service ORA1
OSC$CFG-I-BIND, successfully added resource FAILIP::ORA1@VMSS2 to service ORA1
OSC$CFG-I-BIND, successfully added resource ORA::ORA1 to service ORA1
OSC$CFG-I-BIND, successfully added resource PRC::FAILSAFEIP to service ORA1
OSC$CFG-I-BIND, successfully added resource SHD::ORA1_DSK to service ORA1
OSC$CFG-I-BIND, successfully added resource SHD::ORA1_REDO to service ORA1
OSC$CFG-I-BIND, successfully added resource SHD::ORA_SYSDSK to service ORA1

OSC$CFG> DEFINE RESOURCE ORALS::ORA2/MEMBER=ORA2
OSC$CFG-I-BIND, successfully added resource ORALS::ORA2 to service ORA2
OSC$CFG-I-BIND, successfully added resource FAILIP::ORA2@VMSS1 to service ORA2
OSC$CFG-I-BIND, successfully added resource FAILIP::ORA2@VMSS2 to service ORA2
OSC$CFG-I-BIND, successfully added resource ORA::ORA2 to service ORA2
OSC$CFG-I-BIND, successfully added resource PRC::FAILSAFEIP to service ORA2
OSC$CFG-I-BIND, successfully added resource SHD::ORA2_DSK to service ORA2
OSC$CFG-I-BIND, successfully added resource SHD::ORA2_REDO to service ORA2
OSC$CFG-I-BIND, successfully added resource SHD::ORA_SYSDSK to service ORA2
```

9. We configure the service group membership of the services ORA1 and ORA2.

```
OSC$CFG> DEFINE SERVICE ORA1/MEMBER=ORA1
OSC$CFG-I-BIND, successfully added service ORA1 to service group ORA1

OSC$CFG> DEFINE SERVICE ORA2/MEMBER=ORA2
OSC$CFG-I-BIND, successfully added service ORA2 to service group ORA2
```

10. We verify the configuration.

```
OSC$CFG> VERIFY DATABASE
OSC$MGR-I-VERIFY, current working CFG database contains a valid configuration
```

11. The configuration verification reports no errors. Thus, we can now compare the configured resource, service and service group layout with the resource dependency diagrams.

```
OSC$CFG> SHO SRVGRP */LAYOUT
```

```
Current working CFG database SRVGRP layout:
-------------------------------------------------

   ORA1                                          OK
     |-VMSS1                                     OK
     |    |-[Srv] ORA1                           OK
     |    |    |-[Res] ORALS::ORA1               OK
     |    |    |    |-[Res] FAILIP::ORA1         OK
     |    |    |    |    |-[Res] PRC::FAILSAFEIP OK
     |    |    |    |-[Res] ORA::ORA1            OK
     |    |    |    |    |-[Res] SHD::ORA1_DSK   OK
     |    |    |    |    |-[Res] SHD::ORA1_REDO  OK
     |    |    |    |    |-[Res] SHD::ORA_SYSDSK OK
     |-VMSS2                                     OK
     |    |-[Srv] ORA1                           OK
     |    |    |-[Res] ORALS::ORA1               OK
     |    |    |    |-[Res] FAILIP::ORA1         OK
     |    |    |    |    |-[Res] PRC::FAILSAFEIP OK
     |    |    |    |-[Res] ORA::ORA1            OK
     |    |    |    |    |-[Res] SHD::ORA1_DSK   OK
     |    |    |    |    |-[Res] SHD::ORA1_REDO  OK
     |    |    |    |    |-[Res] SHD::ORA_SYSDSK OK

   ORA2                                          OK
     |-VMSS1                                     OK
     |    |-[Srv] ORA2                           OK
     |    |    |-[Res] ORALS::ORA2               OK
     |    |    |    |-[Res] FAILIP::ORA2         OK
     |    |    |    |    |-[Res] PRC::FAILSAFEIP OK
     |    |    |    |-[Res] ORA::ORA2            OK
     |    |    |    |    |-[Res] SHD::ORA2_DSK   OK
     |    |    |    |    |-[Res] SHD::ORA2_REDO  OK
     |    |    |    |    |-[Res] SHD::ORA_SYSDSK OK
     |-VMSS2                                     OK
     |    |-[Srv] ORA2                           OK
     |    |    |-[Res] ORALS::ORA2               OK
     |    |    |    |-[Res] FAILIP::ORA2         OK
     |    |    |    |    |-[Res] PRC::FAILSAFEIP OK
     |    |    |    |-[Res] ORA::ORA2            OK
     |    |    |    |    |-[Res] SHD::ORA2_DSK   OK
     |    |    |    |    |-[Res] SHD::ORA2_REDO  OK
     |    |    |    |    |-[Res] SHD::ORA_SYSDSK OK

-------------------------------------------------
```

12. The OSC resources, OSC services and OSC service group layout match the resource dependency diagrams. Thus, finally we configure the OSC event notification service as defined, execute an OSC cluster-wide shutdown, activate the working OSC configuration database as the default configuration database and restart OSC.

    OSC$CFG> MODIFY EVENT OSCCTRL_CNXMAN_EVT

```
Welcome to the EVENT OSC configuration wizard
-------------------------------------------
Dscr: Event Log-File severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtLogPriority} [1]:
Dscr: Event OPCOM-Msg severity
(0=none,1=Inf,2=Warn,3=Err,4=Fat)
Attr: {EvtOpcomPriority} [0]:
Dscr: OSC console severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtMgrFwd} [1]:
Dscr: Event User-Script severity
(0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtUserScriptPriority} [0]: 3
Dscr: Event User-Script filename
Attr: {EvtUserScript} []: OSC$CFG:OSC_SENDMAIL.COM
OSC$CFG-I-MODIFY, item 'OSCCTRL_CNXMAN_EVT' has been
updated in the current working CFG database
```

OSC$CFG> MODIFY EVENT OSCCTRL_STATE_EVT

```
Welcome to the EVENT OSC configuration wizard
-------------------------------------------
Dscr: Event Log-File severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtLogPriority} [1]:
Dscr: Event OPCOM-Msg severity (0=none,1=Inf,2=Warn,3=Err,4=Fat)
Attr: {EvtOpcomPriority} [0]:
Dscr: OSC console severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtMgrFwd} [1]:
Dscr: Event User-Script severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtUserScriptPriority} [0]: 3
Dscr: Event User-Script filename
Attr: {EvtUserScript} []: OSC$CFG:OSC_SENDMAIL.COM
OSC$CFG-I-MODIFY, item 'OSCCTRL_STATE_EVT' has been updated in the current
working CFG database
```

OSC$CFG> MODIFY EVENT OSCSRV_CNXMAN_EVT

```
Welcome to the EVENT OSC configuration wizard
-------------------------------------------
Dscr: Event Log-File severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtLogPriority} [1]:
Dscr: Event OPCOM-Msg severity
(0=none,1=Inf,2=Warn,3=Err,4=Fat)
Attr: {EvtOpcomPriority} [0]:
Dscr: OSC console severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtMgrFwd} [1]:
Dscr: Event User-Script severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtUserScriptPriority} [0]: 3
Dscr: Event User-Script filename
Attr: {EvtUserScript} []: OSC$CFG:OSC_SENDMAIL.COM
OSC$CFG-I-MODIFY, item 'OSCSRV_CNXMAN_EVT' has been updated in the current
working CFG database
```

OSC$CFG> MODIFY EVENT OSCSRV_STATE_EVT

```
Welcome to the EVENT OSC configuration wizard
-------------------------------------------
Dscr: Event Log-File severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtLogPriority} [1]:
Dscr: Event OPCOM-Msg severity (0=none,1=Inf,2=Warn,3=Err,4=Fat)
Attr: {EvtOpcomPriority} [0]:
Dscr: OSC console severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtMgrFwd} [1]:
Dscr: Event User-Script severity
(0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtUserScriptPriority} [0]: 3
Dscr: Event User-Script filename
Attr: {EvtUserScript} []: OSC$CFG:OSC_SENDMAIL.COM
OSC$CFG-I-MODIFY, item 'OSCSRV_STATE_EVT' has been updated in the current
working CFG database

OSC$CFG> MODIFY EVENT OSCAGT_STATE_EVT

Welcome to the EVENT OSC configuration wizard
-------------------------------------------
Dscr: Event Log-File severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtLogPriority} [1]:
Dscr: Event OPCOM-Msg severity
(0=none,1=Inf,2=Warn,3=Err,4=Fat)
Attr: {EvtOpcomPriority} [0]:
Dscr: OSC console severity (0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtMgrFwd} [1]:
Dscr: Event User-Script severity
(0=none,1=Inf,2=War,3=Err,4=Fat)
Attr: {EvtUserScriptPriority} [0]: 3
Dscr: Event User-Script filename
Attr: {EvtUserScript} []: OSC$CFG:OSC_SENDMAIL.COM
OSC$CFG-I-MODIFY, item 'OSCAGT_STATE_EVT' has been updated in the current
working CFG database

OSC$CFG> EXIT

$ RUN OSC$BIN:OSC$MGR
OSC$MGR> SHUTDOWN/CLUSTER
OSC$MGR> EXIT

$ RUN OSC$BIN:OSC$CFG
OSC$CFG> OPEN DATABASE/V=1.0
OSC$CFG> ACTIVATE DEFAULT
OSC$CFG> EXIT

$ @SYS$STARTUP:OSC$STARTUP
```

9

Developing new OSC agents

9.1 Overview

Each OSC agent is a program that manages resources of a particular type, such as a disk, a shadow set, an IP address or a database within the OSC cluster environment. Each resource type requires an OSC agent. A single OSC agent manages multiple resources of the same type on one OSC host.

When the OSC service engine starts up, it automatically starts the OSC agents required for the configured resource types. The OSC agents read the resource type specific information required from the default OSC configuration database. The OSC agents execute online and offline commands received from the OSC service engine, periodically monitor the resources, clean-up faulted resources and send status change notification messages to the OSC service engine. The operative state of OSC agents is monitored by the OSC service engine. Thus, if an OSC agent crashes or hangs, it will be automatically restarted by the OSC service engine

OSC provides an OSC agent framework that makes it easy to create new OSC agents for resources not already handled by agents bundled with OSC (for detailed information regarding the agents provided by OSC see section 10 Bundled OSC agents).

9.2 Design considerations

The resources (application, databases, system resources etc.) for which an OSC agent has to be developed for must be capable of being controlled by the agent and be able to operate in an OSC cluster environment. A particular resource has to meet the following criteria:

- ✓ The resource must be capable of being started by a specific set of commands.
- ✓ Each instance of a resource type (i.e. Oracle database) must be capable of being stopped by a defined procedure. Other instances of the resource type must not be affected by this action.

- ✓ The resource must be capable of being stopped cleanly, also by forcible means if necessary.
- ✓ Each instance of a resource must be capable of being monitored. Monitoring can be simple or in-depth. Monitoring a resource becomes more effective when the monitoring test simulates actual user activity (i.e. monitor an Oracle database by issuing predefined SQL queries).
- ✓ The resource must be crash-tolerant. This means, that the resource must be capable of being run on a system that crashes and of being started on a failover node in a known state.
- ✓ The resource must be host-independent within a cluster; that means that there are no licensing requirements or host name dependencies that prevent successful failover.

9.3 Action routines

Developing an OSC agent means developing resource type specific action routines. An action routine is a user (developer) defined plug-in, that is called by the OSC agent framework to execute resource type specific functions. The OSC agent framework supports a specific set of action routines listed below.

- Open
- Monitor
- Online
- Offline
- Clean

Each of the action routines can be implemented in C or as a DCL script.

9.3.1 Resource type specific attributes

Any of the user defined action routines has to be capable of handling any resource of a defined type (i.e. shadow sets) based on the input parameter it receives when it is called by the OSC agent framework. The content and the number of input parameters depend on the resource type (shadow set agent action routines requires different input parameters compared to Oracle DB action routines) and the way the action routines are implemented. Thus, resource configuration attributes consist of common and resource type specific attributes.

No placeholder attributes exist within the common resource attribute section. The resource type specific attributes have to be defined explicitly for each resource type. When the OSC$CFG command ADD AGENT is executed to add a new OSC agent to the configuration database you are prompted to define the resource type specific attributes or when you execute the ADD TEMPLATE command. For detailed information of how to define resource type specific attributes please refer to the ADD AGENT and ADD TEMPLATE command description of the online help of the OSC$CFG utility.

9.3.2 Open

The OSC agent framework calls the open action routine whenever the OSC agent starts managing a resource:

- When the OSC agent starts
- When a disabled resource is enabled again.

When an OSC agent starts, it is guaranteed that the open action routine for each managed resource is called before its Monitor, Online, Offline or Clean action routine. This allows you to include initializations for specific resources. Most OSC agents do not require this functionality and will not implement this entry point. The OSC failSAFE IP agent (see 10.7 OscAgtFailIP – OSC failSAFE IP agent) is the only OSC agent of the bundled OSC agents that executes an open action routine for each managed resource.

The parameters passed to the open action routines are:

- the resource name
- resource type specific parameters defined by the common **ArgList** resource attribute required by the monitor action routine for execution

The open action routine always provides return code OSC$_ONLINE (return code value: 1)

9.3.3 Monitor

The monitor action routine contains the code to determine the status of a resource.

The OSC agent framework calls the monitor action routine:

- periodically to determine the current state of a resource

- after execution of the online action routines to verify that the resource is online
- after execution of the offline action routines to verify that the resource is offline

The parameters passed to the monitor action routines are:

- the resource name
- resource type specific parameters defined by the common **ArgList** resource attribute required by the monitor action routine for execution

Return codes:

- OSC$_ONLINE return code value: 1
  Resource is online
- OSC$_OFFLINE return code value: 9
  Resource is offline
- OSC$_FAULTED return code value: 19
  Resource has failed unrecoverably. This status code signals to the OSC agent to ignore all automatic fault recovery attributes and to immediately declare the resource as faulted. The resource attributes ignored are:
  - **RestartLimit**
  - **ToleranceLimit**
  - **OnlineRetryLimit**

  If the monitor routine returns this status code the time delay between the first occurrence of a fault condition and initiating service group failover processing can be significantly reduced. For detailed information about these configuration attributes please refer to the section 5.5 Controlling OSC Behavior at the Resource Level.
- Any OpenVMS error code
  If an OpenVMS error code is returned, the resource state is immediately set to ADMIN_WAIT (monitor action routine has failed – status of the resource is undefined).

The monitor action routine is mandatory for any OSC agent regardless if the agent manages:

- **On-Off** resources
- **On-Only** resources
- **Persistent** resources

9.3.4 Online

The online action routine contains the code to bring a resource online.

The parameters passed to the online action routine are:

- the resource name
- resource type specific parameters defined by the common **ArgList** resource attribute required by the online action routine for execution

The value returned by the online action routine defines the time in seconds to wait before calling the monitor action routine to verify if the resource is actually online. If the return value is 0 or negative the monitor action routine is immediately called by the OSC agent framework.

The online action routine is mandatory for OSC agents that manage:

- **On-Off** resources
- **On-Only** resources

9.3.5 Offline

The offline action routine contains the code necessary to take a resource offline.

The parameters passed to the offline action routine are:

- the resource name
- resource type specific parameters defined by the common **ArgList** resource attribute required by the offline action routine for execution

The value returned by the offline action routine defines the time in seconds to wait before calling the monitor action routine to verify if the resource is actually offline. If the return value is 0 or negative the monitor action routine is immediately called by the OSC agent framework.

The offline action routine is mandatory for OSC agents that manage:

- **On-Off** resources

9.3.6 Clean

The clean action routine is called by the OSC agent framework when the resource must be forcibly taken offline due to a fault condition. For detailed information about conditions that causes the OSC agent framework to call the clean action routine please refer to section 5.6 How OSC Handles Resource Faults.

The parameters passed to the clean action routine are:

- the resource name
- resource type specific parameters defined by the common **ArgList** resource attribute required by the clean action routine for execution
- reason code why the clean entry point was called

Possible reason codes:

- OSC$RES_UNEXPEXCTED_OFFLINE value: 2
  An online resource unexpectedly (without been triggered by the OSC agent to go offline) changed to the offline state

- OSC$RES_UNEXPECTED_ONLINE value: 4
  An offline resource unexpectedly changed state to online. The resource was started outside of OSC control.

- OSC$RES_MONITOR_HUNG value: 8
  The monitor action routine consistently failed to complete within the expected time period for a particular resource (see the **FaultOnMonitorTmo** and **FaultOnMonitorTmoLimit** attribute description in the section 5.5 Controlling OSC Behavior at the Resource Level).

- OSC$RES_ONLINE_INEFFECTIVE value: 16
  Online validation of the monitor action routine failed after a resource was triggered to go online (online action routine was called). The resource is still offline.

- OSC$RES_ONLINE_HUNG value: 32
  The online action routine did not complete within the expected time (see the OnlineTmoWaitLimit in the section 5.5 Controlling OSC Behavior at the Resource Level).

- OSC$RES_OFFLINE_INEFFECTIVE value: 64
  Offline validation of the monitor action routine failed after a resource was triggered to go offline (offline action routine was called). The resource is still online.

- OSC$RES_OFFLINE_HUNG value: 128
  The offline action routine did not complete within the expected time period (see the OfflineTmoWaitLimit in the section 5.5 Controlling OSC Behavior at the Resource Level).

Return codes:

- OSC$_ONLINE return code value: 1
  The clean routine succeeded to forcibly shutdown the resource.
- Any other valid OpenVMS failure codes
  The clean routine failed to shutdown the resource completely.

The return code of the clean action routine determines whether or not the OSC agent framework tries to recover the fault condition or not. For more detailed information please refer to the section 5.6 How OSC Handles Resource Faults.

The clean entry point is mandatory for OSC agents that manage:

- **On-Off** resources

9.3.7 Passing parameters to the action routines

The OSC agent framework passes up to 8 parameters to the action routines. The value of the common resource attribute **PassFuncCode** defines the number of freely definable action routine input parameters.

9.3.7.1 PassFuncCode common resource attribute

The value of the **PassFuncCode** defines whether or not the action routine's function code is passed to the action routines. The action routine function code is required to be passed to the action routines if a common entry point (the same DCL script or the same C code function is called for any action routine) is used for all action routines. The function code defines the action to be processed (monitor, online, offline, clean). If the value of the common resource attribute **PassFuncCode** of a resource is TRUE the first parameter (P1) passed to the action routine for that particular resource is a string containing the function code keyword:

- OPEN
- MONITOR
- ONLINE
- OFFLINE
- CLEAN

Due to this mechanism it is possible to create a single DCL script that contains the specific code for all callable action routines.

The advantage is that the user just has to maintain one DCL script. If the common resource attribute **PassFuncCode** is set to FALSE the action routine's function code is not passed to the action routines by the OSC agent framework. Thus, a dedicated DCL script must exist for each action routine for the OSC agent and possibly up to four DCL scripts have to be maintained.

The disadvantage is that only up to 5 user defineable arguments can be passed to an action routine. If the resource attribute **PassFuncCode** is set to FALSE up to 6 user definable arguments can be passed to an action routine.

9.3.7.2 Resource name

If the value of the **PassFuncCode** is TRUE the second parameter (P2) contains the resource name. Otherwise the resource name is passed in P1.

9.3.7.3 ArgList common resource attribute

The common resource attribute **ArgList** specifies the list of resource attributes (common or specific) whose values are passed to the monitor, online, offline and clean routines. If the common resource attribute **PassFuncCode** is TRUE, the **ArgList** argument list may contain up to 5 resource attributes that are passed in P3 - P7 to the action routines. Otherwise the **ArgList** argument list may contain up to 6 resource attributes in P2 - P7.

9.3.7.4 Reason codes

If the clean action routine is called the OSC action routine passes the reason code for calling the clean action routine in P8. Possible reason codes are (see also the description of the clean action routine in the previous sections):

- 2 -> OSC$RES_UNEXPEXCTED_OFFLINE
- 4 -> OSC$RES_UNEXPECTED_ONLINE
- 8 -> OSC$RES_MONITOR_HUNG
- 16 -> OSC$RES_ONLINE_INEFFECTIVE
- 32 -> OSC$RES_ONLINE_HUNG
- 64 -> OSC$RES_OFFLINE_INEFFECTIVE
- 128 -> OSC$RES_OFFLINE_HUNG

The parameter P8 is unused when the OSC agent framework calls the monitor, online or offline action routine.

9.3.7.5 Argument summary

The table below summarizes the arguments passed to the different action routines.

Tab. 9.1: Arguments passed to the OSC agent action routines.

Action routine	Parameter
Open	**PassFuncCode** = FALSE • P1 … Resource name • P2-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList** **PassFuncCode** = TRUE • P1 … "OPEN" • P2 … Resource name • P3-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList**
Monitor	**PassFuncCode** = FALSE • P1 … Resource name • P2-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList** **PassFuncCode** = TRUE • P1 … "MONITOR" • P2 … Resource name • P3-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList**
Online	**PassFuncCode** = FALSE • P1 … Resource name • P2-P7 … Values of the resource attributes addressed by the common

resource attribute **ArgList**

PassFuncCode = TRUE
- P1 … "ONLINE"
- P2 … Resource name
- P3-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList**

Offline

PassFuncCode = FALSE
- P1 … Resource name
- P2-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList**

PassFuncCode = TRUE
- P1 … "OFFLINE"
- P2 … Resource name
- P3-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList**

Clean

PassFuncCode = FALSE
- P1 … Resource name
- P2-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList**
- P8 … Clean reason code

PassFuncCode = TRUE
- P1 … "CLEAN"
- P2 … Resource name
- P3-P7 … Values of the resource attributes addressed by the common resource attribute **ArgList**
- P8 … Clean reason code

The OSC agent framework passes the arguments as command line parameters to DCL scripts using quotation marks for each command line parameter. Thus, one command parameter can contain a list of values.

If the OSC agent framework calls a C code action routine the pointer to the **tAttrItemList** descriptor array containing the arguments listed above

is passed to the appropriate C function. For detailed description of the **AttrItemList** descriptor, please refer to the data structure description in section 9.6.2 tAttrItemList descriptor.

9.4 Using C or DCL scripts

OSC agents can be developed either using C or DCL scripts. Both methods have their own advantages and disadvantages

9.4.1 Advantages using C

The advantage of using C is that the action routines are compiled and linked with the OSC agent framework library. They run as a part of the OSC agent process, so there is no system overhead when they are called.

9.4.2 Advantage using DCL scripts

The advantage of using DCL scripts is that the action routines can be modified dynamically. However, a new sub-process is created each time the action routine is called.

9.4.3 Using C and DCL scripts

You can combine both methods for developing a new agent. You can implement some of the action routines in C and the other using DCL scripts.

It is also possible to modify the action routines of a compiled OSC agent without re-compiling and re-linking the agent. The OSC agent framework always checks whether a DCL script is defined to be executed for a particular action routine before calling the appropriate compiled routine. If a DCL script is defined this script is executed instead of the compiled routine. The presence of a DCL action script overrides the execution of a compiled action routine.

DCL action scripts can be defined resource specific. Therefore it is possible to use a complied OSC agent to manage the majority of the resources of a particular type and use resource specific DCL scripts for a few resources of this type that have to be treated differently.

9.5 Implementing action routines using DCL scripts

If you create a new OSC agent using DCL scripts, an OSC agent image has to be defined to run within the process context of the new OSC agent that provides the OSC agent framework functions. Due to the behavior of the OSC agent framework described in the previous section any of the bundled compiled OSC agents can be used to provide the framework functionality. As long you do not mix compiled and DCL script action routines it is recommended that the following image is used:

- OSC$BIN:OSCAGT$GENERIC.EXE

The OSC agent framework image has to be defined when adding the OSC agent to the configuration database.

9.5.1 Example: The FileOnOff OSC Agent using DCL Scripts

This section provides an example of how to implement a simple OSC agent designed to manage single files using DCL scripts. Each FileOnOff resource manages one file. The file name has to be passed to the DCL action routines. Thus, the resource type specific attribute **FileName** must be defined when adding the FileOnOff OSC agent to the OSC configuration database. In addition the common resource attribute **ArgList** of each FileOnOff resource must contain the **FileName** attribute, so that the OSC agent framework passes the value of the **FileName** attribute (file name to be handled) to the appropriate action routine (see section

9.7 Adding an OSC agent to the configuration database).

Separate DCL scripts are created for each action routine except for the open action routine, since no resource initialization is required. This implies that we have to set the common resource attribute **PassFuncCode** to FALSE for each resource and that the file name stored in the **FileName** attribute will be passed to the action routines in P2.

The OSC installation procedure provides the FileOnOff DCL action scripts. They are located in OSC$EXAMPLES directory.

9.5.1.1 DCL monitor script

The monitor script FILEONOFF_MONITOR.COM checks if a file with the file name passed in the P2 parameter exists. If it exists the value 1 (online) is returned to the caller. Otherwise the value 9 (offline) is returned.

```
$! FileOnOff monitor script
$!
$! Checks whether or not a file defined in P2 exists.
$! If it exists OSC$_ONLINE is reurned.
$! If it does not exists OSC$_OFFLINE ireturned.
$!
$! ------------------------------------------------
$!
$ OSC$_ONLINE = 1
$ OSC$_OFFLINE = 9
$!
$ if (F$SEARCH (""'P2'") .EQS. "") then exit (OSC$_OFFLINE)
$!
$ exit (OSC$_ONLINE)
```

9.5.1.2 DCL online script

The online script FILEONOFF_ONLINE.COM creates the file with the name passed in the P2 parameter. The DCL script returns 0 to the caller. Thus, the OSC agent framework immediately triggers the monitor action routine for online check (see the section 9.3 Action routines in this chapter).

```
$! FileOnOff online script
$!
$! Creates a file with the name passed in P2
$!
```

```
$! Return code = 0 to trigger immediate online check
$!
$! ------------------------------------------------
$!
$ CREATE 'P2
$!
$ exit (0)
```

9.5.1.3 DCL offline script

The online script FILEONOFF_OFFLINE.COM deletes the file with the name passed in parameter P2. The DCL script returns 0 to the caller. Thus, the OSC agent framework immediately triggers the monitor action routine for offline check (see the section 9.3 Action routines in this chapter).

```
$! FileOnOff offline script
$!
$! Deletes a file with the name passed in P2
$!
$! Return code = 0 to trigger immediate online check
$!
$! ------------------------------------------------
$!
$ FileName = "'P2'" - ";*" + ";*"
$ DELETE/NOLOG/NOCONF 'FileName'
$!
$ exit (0)
```

9.5.1.4 DCL clean script

The clean action routine is used to clean a resource forcibly. Thus the clean action script FILEONOFF_CLEAN.COM checks whether the file exists. If it exists it tries to delete the file. If the script cannot delete the file, the status code OSC$_FAULTED (value 19) is returned, otherwise the script returns OSC$_SUCCESS (value 1). Due to the simplicity of the cleanup action the clean reason code passed in P8 is not processed.

```
$! FileOnOff clean script
$!
$! Deletes a file with the name passed in P2
$!
$! If the script fails to delete the file (e.g. RMS-E_FLK)
$! OSC$_FAULTED is returned. Otherwise we return
$! OSC$_SUCCESS.
```

```
$!
$! ------------------------------------------------
$!
$ OSC$_SUCCESS = 1
$ OSC$_FAULTED = 19
$!
$ if (f$search (""'P2'"") .EQS. "") then exit (OSC$_SUCCESS)
$!
$ FileName = ""'P2'" - ";*" + ";*"
$ DELETE/NOLOG/NOCONF 'FileName'
$ if ($SEVERITY .NE. 1) then exit (OSC$_FAULTED)
$
$!
$ exit ($STATUS)
```

9.6 Implementing action routines using C

This section describes how to use C to implement agent action routines.

9.6.1 C code syntax

1. Each C module of your OSC agent must contain an include statement that includes the OSC agent framework header file.

   ```
   #include "OSC$INCLUDE:OSCAGENT_MAIN.H"
   ```

2. One C module must contain a main () routine. Call the OSC agent framework function OscAgentMain from the main () routine as shown below. The OscAgentMain routine is a function that returns an integer.

   ```
   main (int argc, char *argv[])
   {
   int   iStatus;

   iStatus = OscAgentMain (argc, argv);
   exit (iStatus);
   }
   ```

3. The OscAgentMain function initializes the OSC agent framework. During the initialization phase the OSC agent framework calls the external function **OscAgtDispFDTInit()** to initialize the function (action routine) dispatch table. Your C code must contain this function being declared as external.

   ```
   extern int     OscAgtFDTInit (void)
   {
            ....
   }
   ```

4. Register the C functions of your code that the OSC agent framework calls when it triggers the monitor, online, offline and clean action routine in **OscAgtFDTInit()**. No open action routine is provided since no resource initialization is required.

   ```
   extern int     OscAgtFDTInit (void)
   {
   int   iStatus;

   /* register the monitor action routine                          */
   /* in this example monitor function name is OscAgtMonitor   */
   ```

```
iStatus = OscAgtDispFDTRegister (OSCAGT$ACTION_MONITOR, (PIF)
OscAgtMonitor);
RETERROR (iStatus);

/* register the online action routine                    */
/* in this example monitor function name is OscAgtOn     */
iStatus = OscAgtDispFDTRegister (OSCAGT$ACTION_ONLINE, (PIF) OscAgtOn);
RETERROR (iStatus);

/* register the offline action routine                   */
/* in this example monitor function name is OscAgtOff    */
iStatus = OscAgtDispFDTRegister (OSCAGT$ACTION_OFFLINE, (PIF) OscAgtOff);
RETERROR (iStatus);

/* register the clean action routine                     */
/* in this example monitor function name is OscAgtClean  */
iStatus = OscAgtDispFDTRegister (OSCAGT$ACTION_CLEAN, (PIF) OscAgtClean);
return (iStatus);
}
```

You do not have to register all the action routines if the new OSC agent only manages **Persistent** or **On-Only** resources.

5. Create the resource type specific C action routines you have registered in OscAgtFDTInit (). These routines have to be declared as C functions returning an integer.

   Prototype:

   ```
   int  function-name (tAttrItemList *prArg)
   {
   ...
   }
   ```

   Example:

   ```
   int  OscAgtMonitor (tAttrItemList *prArg)
   {
   ...
   }
   ```

9.6.2 tAttrItemList descriptor

```
/* Argument descriptor data structure */

typedef struct
{
```

```
void              *sAttr;     /* attribute name */
unsigned short    wLength;    /* length         */
unsigned short    wType;      /* parameter type */
void              *pvAddress;
                  /* buffer address containing */
                  /* the attribute value(s) */
} tAttrItemList;
```

- sAttr contains the resource name
- wLength length of the attribute value data buffer
- wType data type of the attribute value
  - FIELD$_BOOLEAN integer value 0 or 1
  - FIELD$_STRING string type
  - FIELD$_INTEGER integer value
  - FIELS$_QUAD quad word (__int64)
- *pvAddress pointer to the buffer containing the attribute values

Note

The clean reason code (see section 9.3.7 Passing parameters to the action routines) is always passed as a string to the clean action routine. Use the atoi () C run-time function to convert it to integer. E.g. the string "128" is passed to the clean action routine if the reason code is OSC$RES_OFFLINE_HUNG.

9.6.3 Compile and Link

Compile your C modules. The /FLOAT=G_FLOAT qualifier must be applied.

$ CC/LIST/FLOAT=G_FLOAT *your_C_module*+SYS$LIBRARY:SYS$LIB_C/LIB

Link your compiled modules and the OSC agent framework object libraries into your OSC agent program image. To link the OSC agent image on OpenVMS Alpha use:

$ LINK/MAP/EXE=*OscAgent_image-name* SYS$INPUT/OPT
your_C_objects
OSC$INCLUDE:OSC_AGENT_AXP/LIB
OSC$INCLUDE:OSC_COMMON_ROUTINES_AXP/LIB
IDENTIFICATION="V1.4"

To link the OSC agent image on OpenVMS Integrity use:

$ LINK/MAP/EXE=*OscAgent_image-name* SYS$INPUT/OPT
your_C_objects

```
OSC$INCLUDE:OSC_AGENT_IA64/LIB
OSC$INCLUDE:OSC_COMMON_ROUTINES_IA64/LIB
IDENTIFICATION="V1.4"
```

9.6.4 Example: The FileOnOff OSC agent using C

This section provides the C code for the simple FileOnOff OSC agent designed to manage single files. It provides the same functionality as the DCL based agent described in the previous sections. The OSC installation procedure provides these code examples in the OSC$EXAMPLES directory.

```
#define __NEW_STARLET
/*
** FileOnOff OSC agent example
**
** This agent checks if a file passed in P2 exists
** It creates it in the ONLINE action routine
** It deletes it in the OFFLINE action routine
** The CLEAN action routine is identical to the
** OFFLINE action routine except that it check
** the return code of the delet operation.
**
** Compile:
** CC/FLOAT=G_FLOAT –
**                    OASAGENT_FILEONOFF+SYS$LIBRARY:SYS$LIB_C/LIB
**
** Link on AXP:
** $ LINK/EXE=OSCAGT$FILEONOFF.EXE SYS$INPUT/OPT
** OSCAGENT_FILEONOFF
** OSC$INCLUDE:OSC_AGENT_AXP/LIB
** OSC$INCLUDE:OSC_COMMON_ROUTINES_AXP/LIB
** IDENTIFICATION="V1.4"
**
** Link on IA64:
** $ LINK/EXE=OSCAGT$FILEONOFF.EXE SYS$INPUT/OPT
** OSCAGENT_FILEONOFF
** OSC$INCLUDE:OSC_AGENT_IA64/LIB
** OSC$INCLUDE:OSC_COMMON_ROUTINES_IA64/LIB
** IDENTIFICATION="V1.4"
**
** For detailed description please refer to the
** manual:
**      VSI OpenVMS ServiceControl - User's guide
**
**/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ssdef.h>
#include <starlet.h>
#include <lib$routines.h>

#include "osc$include:oscagent_main.h"

int     OscAgtMonitor (tAttrItemList *prArg)
{
int         iStatus;
char        sFileIn[256];
char        sFileFound[256];

    /* P1 ... virtual resource   */
    /* P2 ... file name to check */
    memset (sFileIn, 0, sizeof (sFileIn));
    memset (sFileFound, 0, sizeof (sFileFound));
    strncpy (sFileIn, (char*) prArg[1].pvAddress, prArg[1].wLength);

    /* call a OSC library function */
    /* SS$_NORMAL is return if the file exists */
    /* and the full file name is returned in sFileFound */
    iStatus = HlpGetFileName (    sFileIn, sFileFound, sizeof (sFileFound) - 1);
    if (iStatus != SS$_NORMAL) return (OSC$_OFFLINE);
    return (OSC$_ONLINE);
}

int     OscAgtOnline (tAttrItemList *prArg)
{
int         iStatus;
FILE        *fh;
char        sFileIn[256];

    /* P1 ... virtual resource   */
    /* P2 ... file name to check */
    memset (sFileIn, 0, sizeof (sFileIn));
    strncpy (sFileIn, (char*) prArg[1].pvAddress, prArg[1].wLength);

    /* create the file */
    fh = fopen (sFileIn, "w");
    if (fh != NULL) fclose (fh);

    return (0);
}

int     OscAgtOffline (tAttrItemList *prArg)
```

```
{
int        iStatus;
char       sFileIn[256];

   /* P1 ... virtual resource   */
   /* P2 ... file name to check */
   memset (sFileIn, 0, sizeof (sFileIn));
   strncpy (sFileIn, (char*) prArg[1].pvAddress, prArg[1].wLength);

   /* delete the file - use OSC agent library function*/
   iStatus = HlpDeleteFile (sFileIn);
   /* ignore return code */

   return (0);
}

int     OscAgtClean (tAttrItemList *prArg)
{
int        iStatus;
char       sFileIn[256];

   /* P1 ... virtual resource   */
   /* P2 ... file name to check */
   /* P8 ... reason code -> ignored */
   memset (sFileIn, 0, sizeof (sFileIn));
   strncpy (sFileIn, (char*) prArg[1].pvAddress, prArg[1].wLength);

   /* check if the file exists */
   if (OscAgtMonitor (prArg) == OSC$_OFFLINE) return (OSC$_ONLINE);

   /* delete the file - use OSC agent library function*/
   iStatus = HlpDeleteFile (sFileIn);
   if (iStatus != SS$_NORMAL) return (OSC$_FAULTED);
   /* ignore return code */

   return (OSC$_ONLINE);
}

extern int      OscAgtDispFDTInit (void)
{
int        iStatus;

   /* fill in all action entry point routines */

   /* monitor entry point */
   iStatus = OscAgtDispFDTRegister (OSCAGT$ACTION_MONITOR,
                                                    (PIF) OscAgtMonitor);
   RETERROR (iStatus);
```

```
    /* online entry point */
    iStatus = OscAgtDispFDTRegister (OSCAGT$ACTION_ONLINE,
                                                    (PIF) OscAgtOnline);
    RETERROR (iStatus);

    /* offline entry point */
    iStatus = OscAgtDispFDTRegister (OSCAGT$ACTION_OFFLINE,
                                                    (PIF) OscAgtOffline);
    RETERROR (iStatus);

    /* clean entry point */
    iStatus = OscAgtDispFDTRegister (OSCAGT$ACTION_CLEAN,
                                                    (PIF) OscAgtClean);
    return (iStatus);
}

main (int argc, char *argv[])
{
int         iStatus;

    /* call main loop */
    iStatus = OscAgentMain (argc, argv);
    exit (iStatus);
}
```

9.7 Adding an OSC agent to the configuration database

OSC agents have to be registered in the OSC configuration database before they can be used. The OSC service engine started on the OSC cluster members read the default OSC configuration database and start the OSC agents required to manage the configured resources.

To register an OSC agent in the OSC configuration database:

- Start the OSC$CFG utility
- Create a new or open an existing working CFG configuration database
- Apply the ADD AGENT command

The parameter applied to the ADD AGENT command defines the name of the OSC agent. The OSC agent name has to be unique within an OSC configuration database.

The ADD AGENT command starts the OSC agent registration wizard. This wizard prompts you to enter all required parameters to register an OSC agent in the configuration database:

- Name of the resource type the agent manages
- Agent attributes
  The required attributes are:
  - **AgentName**
    OpenVMS process name of the OSC agent when it is started by the OSC Service engine.
  - **AgentDescription**
    Short description (64 characters) of the new OSC agent.
  - **AgentCategory**
    The resource category (**On-Off**, **On-Only**, **Persistent**) the OSC agent can manage. When the OSC agent starts the **AgentCategory** attribute is passed on to the **ResourceCategory** attribute for all resources the agent manages if not otherwise defined at the resource level. The **ResourceCategory** attribute overrules the AgentCategory attribute. Thus, the user can re-define the resource category (**On-Off**, **On-Only**, **Persistent**) of a resource without modifying the OSC agent. For detailed information about resource categories please refer to the section 4.2.1.3 Resource Categories)
  - **AgentImage**

This attribute defines the image to activate when the OSC agent process is started by the OSC service engine. If a compiled OSC agent is added enter the directory and image name of the compiled OSC agent. If you add a DCL script based OSC agent you have to enter one of the bundled OSC agent images (provided by the OSC installation procedure) located in OSC$BIN that provide the OSC agent framework functionality. VSI recommends you to use the image OSC$BIN:OSCAGT$GENERIC.EXE if no specific functionality of any other bundled OSC agent image is required.

- **AgentUIC**
  The OSC agent will be started under the UIC of the user defined by this attribute.
- **AgentMaxAction**
  This attribute defines how many resources can be handled by the OSC agent in parallel. If more resources are configured than the number defined by this attribute, resource processing is partly serialized. If the value of the attribute is set to 1 resource processing is completely serialized.

Note

Do not enter a value greater than 240.

The Agent attribute section in the appendix lists and describes all agent attributes – required and optional attributes.

- Resource type specific attributes
  The wizard prompts you to define the resource type specific resource attributes (see section 9.3.1 Resource type specific attributes). Any OSC attributes are stored as descriptors. Thus, you are prompted to define
  - Attribute name
  - Attribute description
  - Attribute data type
    - String
    - Integer
    - Quad
    - Boolean
  - Attribute data length if the attribute data type = STRING
  - Attribute configuration option
    - Required

Whenever you add or modify a resource the resource configuration wizard prompts you for input.

  - Optional
    The resource configuration wizards prompts you for input only if you have applied the /ADVANCED qualifier. For more information about the resource configuration wizard please refer to the ADD RESOURCE online help of the OSC$CFG utility.

- Resource type template
  The resource type template contains the resource attribute defaults for a particular resource type.

Use quotation marks for case sensitive string inputs. If you omit quotation marks string inputs will be converted to upper case.

9.7.1 DCL script based OSC agent FileOnOff

In this example the DCL script based OSC agent FileOnOff is registered in the OSC configuration database. Since the action routines are implemented as DCL scripts a bundled OSC agent image has to be defined to provide the OSC agent framework functionality. In this case we use the generic OSC agent framework image OSC$BIN:OSCAGT$GENERIC.EXE.

The DCL action routine scripts are assigned to the appropriate configuration attributes:

- FILEONOFF_MONITOR.COM -> MonitorScript
- FILEONOFF_ONLINE.COM -> OnlineScript
- FILEONOFF_OFFLINE.COM -> OfflineScript
- FILEONOFF_CLEAN.COM -> CleanScript

```
OSC$CFG> ADD AGENT FileOnOff

    Dscr: Agent manages resource prototype
    Attr: {AgentType} []: FILEONOFF


Welcome to the AGENT OSC configuration wizard
---------------------------------------------

Dscr: Agent process name
Attr: {AgentName} []: "OscAgtFile"
Dscr: Agent description
Attr: {AgentDescription} []: "OSC agent managing single files"
```

```
Dscr: Agent resource category (Persistent | On-Only| On-Off)
Attr: {AgentCategory} [On-Off]:
Dscr: Agent image
Attr: {AgentImage} []: OSC$BIN:OSCAGT$GENERIC.EXE
Dscr: Agent User name
Attr: {AgentUIC} [SYSTEM]:
Dscr: Agent max concurrent actions
Attr: {AgentMaxAction} [10]:


Welcome to the RESOURCE type specific attribute def wizard
----------------------------------------------------------

Define the resource type specific attributes of the new OSC
agent. To terminate press ENTER when you are prompted for
the field name.

Field Name []: "FileName"
Field Name Description []: "File to manage"
Field Data Type (S=String/I=Integer/Q=Quad/B=Boolean) [S]:
Field Length [32]: 256
Field Cfg Option (R=Required/O=Optional) [R]:

Field Name []:


Welcome to the OSC resource template configuration wizard
---------------------------------------------------------

Dscr: Resource is critical
Attr: {Critical} [Yes]:
Dscr: Resource Enabled
Attr: {Enabled} [Yes]:
Dscr: Online Monitor interval
Attr: {OnlineMonitorInterval} [60 sec]:
Dscr: Offline Monitor interval
Attr: {OfflineMonitorInterval} [300 sec]: 60
Dscr: Monitor OFFLINE tolerance limit
Attr: {ToleranceLimit} [0]:
Dscr: Resource is a fault candidate if the monitor routine
      times out
Attr: {FaultOnMonitorTmo} [Yes]:
Dscr: Number of monitor timeouts to fault resource
Attr: {FaultOnMonitorTmoLimit} [4]: 2
Dscr: Do not manage faults even if SrvGrp is set to manage
      faults
Attr: {DisableMangeFault} [No]:
Dscr: Online retry limit
Attr: {OnlineRetryLimit} [0]: 1
Dscr: Online wait limit
Attr: {OnlineWaitLimit} [2]: 1
Dscr: Online timeout wait limit
Attr: {OnlineTmoWaitLimit} [2]: 1
Dscr: Offline wait limit
Attr: {OfflineWaitLimit} [2]: 1
Dscr: Offline timeout wait limit
Attr: {OfflineTmoWaitLimit} [2]: 1
```

```
Dscr: Resource restart limit
Attr: {RestartLimit} [0]: 1
Dscr: Clean entry retry limit
Attr: {CleanRetryLimit} [5]:
Dscr: Timeout retry limit for action entries
Attr: {TimeOutRetryLimit} [5]:
Dscr: Confidential Limit
Attr: {ConfLimit} [600]:
Dscr: Monitor Script
Attr: {MonitorScript} []: OSC$EXAMPLES:FILEONOFF_MONITOR.COM
Dscr: Monitor Script timeout
Attr: {MonitorTmo} [60]: 10
Dscr: Online Script
Attr: {OnlineScript} []: OSC$EXAMPLES:FILEONOFF_ONLINE.COM
Dscr: Online Script timeout
Attr: {OnlineTmo} [300]: 10
Dscr: Offline Script
Attr: {OfflineScript} []: OSC$EXAMPLES:FILEONOFF_OFFLINE.COM
Dscr: Offline Script timeout
Attr: {OfflineTmo} [300]: 10
Dscr: Cleanup Script
Attr: {CleanScript} []: OSC$EXAMPLES:FILEONOFF_CLEAN.COM
Dscr: Clean Script timeout
Attr: {CleanTmo} [60]: 10
Dscr: Open Script
Attr: {OpenScript} []:
Dscr: Open Script timeout
Attr: {OpenTmo} [60]:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList} []: "FileName"
Dscr: Pass function code in P1 to action scripts
Attr: {PassFuncCode} [No]:

OSC$CFG-I-ADD, template 'FILEONOFF' has been added to the
current working CFG database
OSC$CFG-I-ADD, item 'OSCAGTFILE' has been added to the
working CFG database
```

9.7.2 Compiled OSC agent FileOnOff

In this example the compiled OSC agent FileOnOff is registered in the OSC configuration database. Thus, the image of the compiled OSC agent – OSC$EXAMPLES:OSCAGT$FILEONOFF.EXE – is assigned to the **AgentImage** attribute. Do not assign any script files to the resource type attributes:

- OpenScript
- MonitorScript
- OnlineScript
- OfflineScript
- CleanScript

Otherwise these scripts will be executed instead of the compiled action routines since DCL script action routines take precedence over compiled action routines (see also section Using C or DCL scripts)

OSC$CFG> ADD AGENT FileOnOff

```
    Dscr: Agent manages resource prototype
    Attr: {AgentType} []: FILEONOFF


Welcome to the AGENT OSC configuration wizard
---------------------------------------------

Dscr: Agent process name
Attr: {AgentName} []: "OscAgtFile"
Dscr: Agent description
Attr: {AgentDescription} []: "OSC agent managing single files"
Dscr: Agent resource category (Persistent | On-Only| On-Off)
Attr: {AgentCategory} [On-Off]:
Dscr: Agent image
Attr: {AgentImage} []: OSC$EXAMPLES:OSCAGT$FILEONOFF.EXE
Dscr: Agent User name
Attr: {AgentUIC} [SYSTEM]:
Dscr: Agent max concurrent actions
Attr: {AgentMaxAction} [10]:


Welcome to the RESOURCE type specific attribute def wizard
-----------------------------------------------------------

Define the resource type specific attributes of the new OSC
agent. To terminate press ENTER when you are prompted for
the field name.

Field Name []: "FileName"
Field Name Description []: "File to manage"
Field Data Type (S=String/I=Integer/Q=Quad/B=Boolean) [S]:
Field Length [32]: 256
```

```
Field Cfg Option (R=Required/O=Optional) [R]:

Field Name []:


Welcome to the OSC resource template configuration wizard
---------------------------------------------------------

Dscr: Resource is critical
Attr: {Critical} [Yes]:
Dscr: Resource Enabled
Attr: {Enabled} [Yes]:
Dscr: Online Monitor interval
Attr: {OnlineMonitorInterval} [60 sec]:
Dscr: Offline Monitor interval
Attr: {OfflineMonitorInterval} [300 sec]: 60
Dscr: Monitor OFFLINE tolerance limit
Attr: {ToleranceLimit} [0]:
Dscr: Resource is a fault candidate if the monitor routine
      times out
Attr: {FaultOnMonitorTmo} [Yes]:
Dscr: Number of monitor timeouts to fault resource
Attr: {FaultOnMonitorTmoLimit} [4]: 2
Dscr: Do not manage faults even if SrvGrp is set to manage
      faults
Attr: {DisableMangeFault} [No]:
Dscr: Online retry limit
Attr: {OnlineRetryLimit} [0]: 1
Dscr: Online wait limit
Attr: {OnlineWaitLimit} [2]: 1
Dscr: Online timeout wait limit
Attr: {OnlineTmoWaitLimit} [2]: 1
Dscr: Offline wait limit
Attr: {OfflineWaitLimit} [2]: 1
Dscr: Offline timeout wait limit
Attr: {OfflineTmoWaitLimit} [2]: 1
Dscr: Resource restart limit
Attr: {RestartLimit} [0]: 1
Dscr: Clean entry retry limit
Attr: {CleanRetryLimit} [5]:
Dscr: Timeout retry limit for action entries
Attr: {TimeOutRetryLimit} [5]:
Dscr: Confidential Limit
Attr: {ConfLimit} [600]:
Dscr: Monitor Script
Attr: {MonitorScript} []:
Dscr: Monitor Script timeout
Attr: {MonitorTmo} [60]: 10
Dscr: Online Script
Attr: {OnlineScript} []:
Dscr: Online Script timeout
Attr: {OnlineTmo} [300]: 10
Dscr: Offline Script
Attr: {OfflineScript} []:
Dscr: Offline Script timeout
Attr: {OfflineTmo} [300]: 10
Dscr: Cleanup Script
```

```
Attr: {CleanScript} []:
Dscr: Clean Script timeout
Attr: {CleanTmo} [60]: 10
Dscr: Open Script
Attr: {OpenScript} []:
Dscr: Open Script timeout
Attr: {OpenTmo} [60]:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList} []: "FileName"
Dscr: Pass function code in P1 to action scripts
Attr: {PassFuncCode} [No]:

OSC$CFG-I-ADD, template 'FILEONOFF' has been added to the
current working CFG database
OSC$CFG-I-ADD, item 'OSCAGTFILE' has been added to the
working CFG database
```

9.8 Sending agent specific OSC event

The OSC agent framework provides an interface to the OSC event notification service to send OSC agent specific event notifications to provide more detailed information about the managed resources. For example the bundled RDB and failSAFE IP OSC agents use this feature.

The OSC event service treats OSC agent specific event messages as event messages of the OSCAGT_CONTROL_EVT class. Thus, depending of the definition of the OSCAGT_CONTROL_EVT event class, the OSC event service:

- Logs OSC agent specific events into the common OSC event message file
- Sends an OPCOM message
- Forwards the event to the connected OSC consoles
- Executes a user defined DCL script

For detailed information about OSC event classes and how they are handled by the OSC event service please refer to the section 6 OSC event notification.

9.8.2 DCL script OSC agent

The image OSC$BIN:OSCAGT$SENDEVENT.EXE provides the OSC event service interface for DCL script based OSC agent. To use this interface define a foreign command symbol and pass three command line parameters to the image:

- P1 OSC resource name
  The OSC agent framework passes the resource name in P1 to the OSC action script
- Event Severity
  Use one of the keywords listed below:
  - INFO informational event
  - WARNING warning event
  - ERROR error event
  - FATAL fatal event
- Specific message Test

Example:

The RDB OSC agent triggers an agent specific error message if the online DCL script fails to open an RDB database:

```
$ OscAgtSend :== $OSC$BIN:OSCAGT$SENDEVENT.EXE
$ ...
$ MSG_TXT = "Failed to open RDB database ''rdb_db'"
$ OscAgtSend "'P1'" "ERROR" "'MSG_TXT'"
$ ...
```

9.8.3 Compiled OSC agent

The OSC framework library function **OscAgtSendEvent()** provides the interface to the OSC event service.

C prototype:

int OscAgtSendEvent (char *sResource, int iSeverity, char *sMsg)

- sResource resource name
  The first entry of the tAttrItemList descriptor array passed to a compiled action routine contains the resource name.
- iSeverity event severity
  Valid values are:
  - EVT$SEVERITY_INFO informational event
  - EVT$SEVERITY_WARNING warning event
  - EVT$SEVERITY_ERROR error event
  - EVT$SEVERITY_FATAL fatal event
- sMsg specific message text

10

Bundled OSC agents

10.1 OscAgtDSK - OSC disk agent

10.1.1 Description

The OscAgtDSK agent is designed to monitor the availability and accessibility of physical disk devices. OscAgtDSK is a compiled OSC agent that manages the resource type DSK. The agent's image is:

- OSC$BIN:OSCAGT$DSK.EXE

This agent provides only the monitor action routine. Thus, the agent is defined to manage **Persistent** resources only.

Do not change the agent attribute **AgentCategory**.

10.1.2 Default agent attributes

```
AGENT:      OSCAGTDSK

Attributes [Size]            Values
---------------------------------
AgentType [16]:              DSK
AgentName [32]:              OscAgtDSK
AgentDescription [64]:       OSC Agent to monitor physical
disk access
AgentCategory [16]:          Persistent
AgentImage [256]:            OSC$BIN:OSCAGT$DSK.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          10
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTDSK
```

For detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.1.3 Resource type specific attributes

Attribute Name	Data Type	Description
DiskDevList	String[256]	Mandatory It defines the physical disks associated with the resource as a comma separated list. Full wildcard support is provided for each element of the comma separated list. The asterisk (*) and percentage (%) wildcard character can be placed anywhere within each list element.

10.1.4 Default common Resource attributes

The resource type template DSK contains the resource attribute defaults for the resources managed by the OscAgtDSK agent.

```
TEMPLATE:   DSK                     Managed by: OscAgtDSK

Attributes [Size]                   Values
---------------------------------------
Critical [4]:                       TRUE
Enabled [4]:                        TRUE
OnlineMonitorInterval [4]:          10
OfflineMonitorInterval [4]:         10
ToleranceLimit [4]:                 2
FaultOnMonitorTmo [4]:              TRUE
FaultOnMonitorTmoLimit [4]:         4
DisableMangeFault [4]:              FALSE
OnlineRetryLimit [4]:               0
OnlineWaitLimit [4]:                2
OnlineTmoWaitLimit [4]:             2
OfflineWaitLimit [4]:               2
OfflineTmoWaitLimit [4]:            2
RestartLimit [4]:                   0
CleanRetryLimit [4]:                5
TimeOutRetryLimit [4]:              5
ConfLimit [4]:                      600
```

```
MonitorScript [256]:
MonitorTmo [4]:                    10
OnlineScript [256]:
OnlineTmo [4]:                     300
OfflineScript [256]:
OfflineTmo [4]:                    300
CleanScript [256]:
CleanTmo [4]:                      60
OpenScript [256]:
OpenTmo [4]:                      .60
ArgList [256]:                     DiskDevList, Critical
PassFuncCode [4]:                  FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.1.5 How to define a DSK resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtDSK agent manages the resource type DSK. Thus, the syntax to add a new DSK resource to the OSC configuration database using the OSC$CFG utility is:

OSC$CFG> ADD RESOURCE DSK::resource-name[@node-name]

10.2 OscAgtFSYS - OSC single disk volume agent

10.2.1 Description

The OscAgtFSYS agent is designed to manage single disk volumes. It checks the availability, accessibility and the mount status of a single disk volume. This agent can mount (online action) and dismount (offline, clean action) single disk volumes. Although the agent is capable of managing single disk volumes as **On-Off** OSC resources (it provides all required action routines) it is configured to manage **On-Only** (see **AgentCategory** attribute) resources. The main reasons why it is configured as an OSC agent managing **On-Only** resource are:

- In most cases single disk volumes will not be dismounted on an OpenVMS cluster member due to application (service group) failover.
- When the OSC agent starts the **AgentCategory** attribute is passed on to the **ResourceCategory** attribute for all resources the agent manages if not otherwise defined at the resource level. **On-Off** resources cannot be assigned to different services and service groups. Thus, if a single disk volume is accessed by different applications (OSC services – i.e. system disk) and the associated OSC resource is defined as an **On-Off** resource it cannot be a member of both OSC services, although it should be.

The OscAgtFSYS agent is a compiled agent. The agent's image is:

- OSC$BIN:OSCAGT$FSYS.EXE

Note

If your setup requires that some disks have to be defined as **On-Off** resources (disks are dismounted if the associated service group is taken offline) modify the **AgentCategory** attribute to "**On-Off**" using the MODIFY AGENT command of the OSC$CFG utility. In this case ensure that the resource attribute **ResourceCategory** is set to "**On-Only**" for all disks that will not be dismounted (i.e. cluster mounted disks). The resource attribute **ResourceCategory** overrules the agent's default resource type defined by the attribute **AgentCategory** (see section 9.7 Adding an OSC agent to the configuration database)

10.2.2 Default agent attributes

```
AGENT:      OSCAGTFSYS

Attributes [Size]            Values
-----------------------------------
AgentType [16]:              FSYS
AgentName [32]:              OscAgtFSYS
AgentDescription [64]:       OSC agent to monitor JBOD ODS
file system availability
AgentCategory [16]:          On-Only
AgentImage [256]:            OSC$BIN:OSCAGT$FSYS.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          48
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTFSYS
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.2.3 Resource type specific attributes

Attribute Name	Data Type	Description
DiskDevice	String[32]	Mandatory Disk device name (i.e. 1DGA90). The use of wildcards is not permitted.
VolumeLabel	String[32]	Mandatory Volume label of the device. The use of wildcards is not permitted.

10.2.4 Default common Resource attributes

The resource type template FSYS contains the resource attribute defaults for the resources managed by the OscAgtFSYS agent.

```
TEMPLATE:   FSYS                    Managed by: OscAgtFSYS

Attributes [Size]                   Values
---------------------------------------
Critical [4]:                       TRUE
Enabled [4]:                        TRUE
OnlineMonitorInterval [4]:          30
OfflineMonitorInterval [4]:         30
ToleranceLimit [4]:                 2
FaultOnMonitorTmo [4]:              TRUE
FaultOnMonitorTmoLimit [4]:         4
DisableMangeFault [4]:              FALSE
OnlineRetryLimit [4]:               0
OnlineWaitLimit [4]:                2
OnlineTmoWaitLimit [4]:             2
OfflineWaitLimit [4]:               2
OfflineTmoWaitLimit [4]:            2
RestartLimit [4]:                   0
CleanRetryLimit [4]:                5
TimeOutRetryLimit [4]:              5
ConfLimit [4]:                      600
MonitorScript [256]:
MonitorTmo [4]:                     20
OnlineScript [256]:
OnlineTmo [4]:                      300
OfflineScript [256]:
OfflineTmo [4]:                     300
CleanScript [256]:
CleanTmo [4]:                       60
OpenScript [256]:
OpenTmo [4]:                        60
ArgList [256]:                      DiskDevice, VolumeLabel
PassFuncCode [4]:                   FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.2.5 How to define a FSYS resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtFSYS agent manages the resource type FSYS. Thus, the syntax to add a new FSYS resource to the OSC configuration database using the OSC$CFG utility is:

```
OSC$CFG> ADD RESOURCE FSYS::resource-name[@node-name]
```

10.3 OscAgtSHD – OSC shadow set agent

10.3.1 Description

The OscAgtSHD agent is designed to manage shadow sets. It checks the availability and accessibility of the shadow set members and the mount status of a shadow set. This agent can mount (online action) and dismount (offline, clean action) shadow sets. Although the agent is capable of managing shadow sets as **On-Off** OSC resources (it provides all the required action routines) it is configured to manage **On-Only** resources (see **AgentCategory** attribute). The main reasons why it is configured as an OSC agent managing **On-Only** resource are:

- In most cases shadow sets will not be dismounted on an OpenVMS cluster member due to application (service group) failover.
- When the OSC agent starts the **AgentCategory** attribute is passed on to the **ResourceCategory** attribute for all resources the agent manages if not otherwise defined at the resource level. **On-Off** resources cannot be assigned to different services and service groups. Thus, if a shadow set is accessed by different applications (OSC services – i.e. system disk shadow set) and the associated OSC resource is defined as an **On-Off** resource it cannot be a member of both OSC services although it should be.

The OscAgtSHD agent is a compiled agent. The agent's image is:

- OSC$BIN:OSCAGT$SHD.EXE

If the value of the resource type specific attribute **FullMbrOnMount** of a shadow set resource is TRUE the OscAgtSHD agents mounts the shadow set (brings the resource online) only if all configured shadow set members are online and accessible. Otherwise the shadow set is mounted if at least one member is online and accessible.

If the value of the resource type specific attribute **FullMbrOnMount** of a shadow set resource is TRUE the OscAgtSHD agent considers a shadow set as faulted if one member has been removed from the shadow set.

Note

If your setup requires that some shadow sets are defined as **On-Off** resources (shadow sets are dismounted if the associated service group is taken offline) modify the **AgentCategory** attribute to "**On-Off**" using the MODIFY AGENT command of the OSC$CFG utility. This has the advantage that all the necessary action routines (OFFLINE, CLEAN, etc) to manage "**On-Off**" resources are inheritaed from the AGENT definition, otherwise these action routines would need to be defined for all the defined "**On-Off**" resources individually.

In this case make sure that the resource attribute **ResourceCategory** is set to "**On-Only**" for all shadow sets that will not be dismounted (i.e. cluster mounted shadow sets). The resource attribute **ResourceCategory** overrules the agent's default resource category defined by the attribute **AgentCategory** (see section 9.7 Adding an OSC agent to the configuration database)

10.3.2 Default agent attributes

```
AGENT:      OSCAGTSHD

Attributes [Size]            Values
---------------------------------
AgentType [16]:              SHD
AgentName [32]:              OscAgtSHD
AgentDescription [64]:       OSC agent to monitor Shadow
Sets
AgentCategory [16]:          On-Only
AgentImage [256]:            OSC$BIN:OSCAGT$SHD.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          48
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTSHD
```

For detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.3.3 Resource type specific attributes

Attribute Name	Data Type	Description
ShadName	String[32]	Mandatory Shadow set device name (i.e. DSA10). The use of wildcards is not permitted.
ShadMembers	String[256]	Mandatory This attribute defined the shadow set members. List the shadow set device members as a comma separated list (i.e. 1DGA100, 1DGA200). The use of wildcards is not permitted.
VolumeLabel	String[32]	Mandatory Volume label of the shadow set. The use of wildcards is not permitted.
FullMbrOnMount	Boolean	Mandatory If the value of this attribute is TRUE the OscAgtSHD agents mount the shadow set (bring the resource online) only if all configured shadow set members are online and accessible. Otherwise the shadow set is mounted if at least one member is online and accessible The default value is TRUE.
FullMbrOnMonitor	Boolean	Mandatory If the value of this attribute is TRUE the OscAgtSHD agent considers a shadow set as faulted if one member has been removed from the shadow set The default value is FALSE.

10.3.4 Default common Resource attributes

The resource type template SHD contains the resource attribute defaults for the resources managed by the OscAgtSHD agent.

```
TEMPLATE:   SHD                     Managed by: OscAgtSHD

Attributes [Size]                   Values
---------------------------------------
Critical [4]:                       TRUE
Enabled [4]:                        TRUE
OnlineMonitorInterval [4]:          30
OfflineMonitorInterval [4]:         30
ToleranceLimit [4]:                 2
FaultOnMonitorTmo [4]:              TRUE
FaultOnMonitorTmoLimit [4]:         4
DisableMangeFault [4]:              FALSE
OnlineRetryLimit [4]:               0
OnlineWaitLimit [4]:                2
OnlineTmoWaitLimit [4]:             2
OfflineWaitLimit [4]:               2
OfflineTmoWaitLimit [4]:            2
RestartLimit [4]:                   0
CleanRetryLimit [4]:                5
TimeOutRetryLimit [4]:              5
ConfLimit [4]:                      600
MonitorScript [256]:
MonitorTmo [4]:                     20
OnlineScript [256]:
OnlineTmo [4]:                      300
OfflineScript [256]:
OfflineTmo [4]:                     300
CleanScript [256]:
CleanTmo [4]:                       60
OpenScript [256]:
OpenTmo [4]:                        60
ArgList [256]:                      ShadName, ShadMembers,
                                    VolumeLabel,
                                    FullMbrOnMount,
                                    FullMbrOnMonitor
PassFuncCode [4]:                   FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.3.5 How to define a SHD resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. To define a

resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtSHD agent manages the resource type SHD. Thus, the syntax to add a new SHD resource to the OSC configuration database using the OSC$CFG utility is:

```
OSC$CFG> ADD RESOURCE SHD::resource-name[@node-name]
```

10.4 OscAgtPRC - OSC process agent

10.4.1 Description

The OscAgtPRC agent is a generic process management agent. OscAgtPRC is a compiled OSC agent that manages the resource type PRC. The agent's image is:

- OSC$BIN:OSCAGT$PRC.EXE

The OscAgtPRC agent is typically used if a process existence check is sufficient to determine whether or not an application is still running. This agent provides the monitor (process existence check) and clean action (delete process = STOP/ID) routine to manage OpenVMS processes. It provides no default online and offline routines. The application startup and shutdown scripts have to be defined at the resource level (resource attributes: OnlineScript and OfflineScript). The OscAgtPRC agent is defined (default) to manage **On-Off** resources.

10.4.2 Default agent attributes

```
AGENT:      OSCAGTPRC

Attributes [Size]            Values
---------------------------------
AgentType [16]:              PRC
AgentName [32]:              OscAgtPRC
AgentDescription [64]:       OSC agent to monitor processes
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$PRC.EXE
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          48
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTPRC
```

For detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.4.3 Resource type specific attributes

Attribute Name	Data Type	Description
ProcessList	String[256]	Mandatory It defines the processes associated with the resource as a comma separated list. (i.e. CLUSTER_SERVER, SHADOW_SERVER) Full wildcard support is provided for each element of the comma separated list. The asterisk (*) and percentage (%) wildcard character can be placed anywhere within each list element.
ProcessCount	Integer	Mandatory The number of existing processes that has to match the process list defined by the **ProcessList** attribute. If fewer processes as defined by the attribute exist on a system the monitor routine of the agent resource is considered as offline.

10.4.4 Default common Resource attributes

The resource type template PRC contains the resource attribute defaults for the resources managed by the OscAgtPRC agent.

```
TEMPLATE:   PRC                     Managed by: OscAgtPRC

Attributes [Size]                   Values
---------------------------------------
Critical [4]:                       TRUE
Enabled [4]:                        TRUE
OnlineMonitorInterval [4]:          30
OfflineMonitorInterval [4]:         30
ToleranceLimit [4]:                 2
FaultOnMonitorTmo [4]:              TRUE
FaultOnMonitorTmoLimit [4]:         4
DisableMangeFault [4]:              FALSE
OnlineRetryLimit [4]:               0
OnlineWaitLimit [4]:                2
OnlineTmoWaitLimit [4]:             2
```

```
OfflineWaitLimit [4]:              2
OfflineTmoWaitLimit [4]:           2
RestartLimit [4]:                  0
CleanRetryLimit [4]:               5
TimeOutRetryLimit [4]:             5
ConfLimit [4]:                     600
MonitorScript [256]:
MonitorTmo [4]:                    25
OnlineScript [256]:
OnlineTmo [4]:                     300
OfflineScript [256]:
OfflineTmo [4]:                    300
CleanScript [256]:
CleanTmo [4]:                      60
OpenScript [256]:
OpenTmo [4]:                       60
ArgList [256]:                     ProcessList, ProcessCount
PassFuncCode [4]:                  FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.4.5 How to define a PRC resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtPRC agent manages the resource type PRC. Thus, the syntax to add a new PRC resource to the OSC configuration database using the OSC$CFG utility is:

```
OSC$CFG> ADD RESOURCE PRC::resource-name[@node-name]/ADVANCED
```

The use of the /ADVANCED qualifier is mandatory to define PRC resource since either the online (**OnlineScript** resource attribute) and offline (**OfflineScript** resource attribute) startup script or the resource type (**ResourceCategory** resource attribute) have to be defined. All these resource attributes are optional attributes. Thus, the /ADVANCED qualifier has to be applied, otherwise the resource configuration wizard will not

prompt you to define/modify the values of these resource attributes. For detailed information about the OSC resource configuration wizard please refer to the OSC$CFG utility online help (ADD RESOURCE command).

For example, the agent can be utilized to manage the TCP/IP failSAFE IP service. The failSAFE IP service must be enabled and started on a node in order to manage service IP addresses using the OscAgtFailIP agent (recommended service IP address management agent). The failSAFE IP service will not be stopped if a service IP address is switched to another OSC cluster member. Thus, the resource will be configured as "**On-Only**".

```
OSC$CFG> ADD RESOURCE PRC::FAILSAFEIP/ADVANCED

Welcome to the RESOURCE OSC configuration wizard
------------------------------------------------

Dscr: Resource is valid for node
Attr: {ResourceNode} [*]:
Dscr: Resource Description
Attr: {ResourceDescription} []: "failSAFE IP service"
Dscr: Resource Category (Persistent | On-Only| On-Off)
Attr: {ResourceCategory} []: On-Only
Dscr: Process list to monitor
Attr: {ProcessList} []: TCPIP$FAILSAF*
Dscr: Number of processes that have to exist
Attr: {ProcessCount} [1]:
Dscr: Monitor Script
Attr: {MonitorScript} []:
Dscr: Monitor Script timeout
Attr: {MonitorTmo} [25]:
Dscr: Online Script
Attr: {OnlineScript} []: SYS$STARTUP:TCPIP$FAILSAFE_STARTUP.COM
Dscr: Online Script timeout
Attr: {OnlineTmo} [300]: 30
Dscr: Offline Script
Attr: {OfflineScript} []:
Dscr: Offline Script timeout
Attr: {OfflineTmo} [300]:
Dscr: Cleanup Script
Attr: {CleanScript} []:
Dscr: Clean Script timeout
Attr: {CleanTmo} [60]:
Dscr: Open Script
Attr: {OpenScript} []:
Dscr: Open Script timeout
Attr: {OpenTmo} [60]:
Dscr: Argument list to be passed to agent entries
Attr: {ArgList} [ProcessList, ProcessCount]:
Dscr: Pass function code in P1 to action scripts
Attr: {PassFuncCode} [No]:
OSC$CFG-I-ADD, RESOURCE 'PRC::FAILSAFEIP' has been added to
the working CFG database.
```

Since the PRC::FAILSAFEIP resource is defined as an **On-Only** resource, only the online DCL action script has to be defined – the failSAFE IP service startup script SYS$STARTUP:TCPIP$FAILSAFE_STARTUP.COM.

10.5 OscAgtETH – OSC Ethernet adapter agent

10.5.1 Description

The OscAgtETH agent is designed to monitor the operational state of Ethernet adapters. OscAgtETH is a compiled OSC agent that manages the resource type ETH. The agent's image is:

- OSC$BIN:OSCAGT$ETH.EXE

This agent provides only the monitor action routine. Thus, the agent is defined to manage **Persistent** resources only.

10.5.2 Default agent attributes

```
AGENT:      OSCAGTETH

Attributes [Size]            Values
----------------------------------
AgentType [16]:              ETH
AgentName [32]:              OscAgtETH
AgentDescription [64]:       OSC agent to monitor link
status of ethernet devices
AgentCategory [16]:          Persistent
AgentImage [256]:            OSC$BIN:OSCAGT$ETH.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          24
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTETH
```

For detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.5.3 Resource type specific attributes

Attribute Name	Data Type	Description
EthDevList	String[256]	Mandatory This attribute defines the Ethernet devices to monitor. Enter the template devices of the Ethernet adapters as a comma separated list (i.e. EWA0,EIA0, EIB0).

10.5.4 Default common Resource attributes

The resource type template ETH contains the resource attribute defaults for the resources managed by the OscAgtETH agent.

```
TEMPLATE:   ETH                     Managed by: OscAgtETH

Attributes [Size]                   Values
----------------------------------------
Critical [4]:                       TRUE
Enabled [4]:                        TRUE
OnlineMonitorInterval [4]:          15
OfflineMonitorInterval [4]:         15
ToleranceLimit [4]:                 2
FaultOnMonitorTmo [4]:              TRUE
FaultOnMonitorTmoLimit [4]:         2
DisableMangeFault [4]:              FALSE
OnlineRetryLimit [4]:               0
OnlineWaitLimit [4]:                2
OnlineTmoWaitLimit [4]:             2
OfflineWaitLimit [4]:               2
OfflineTmoWaitLimit [4]:            2
RestartLimit [4]:                   0
CleanRetryLimit [4]:                5
TimeOutRetryLimit [4]:              5
ConfLimit [4]:                      600
MonitorScript [256]:
MonitorTmo [4]:                     10
OnlineScript [256]:
OnlineTmo [4]:                      300
OfflineScript [256]:
OfflineTmo [4]:                     300
CleanScript [256]:
CleanTmo [4]:                       60
OpenScript [256]:
OpenTmo [4]:                        60
ArgList [256]:                      EthDevList, Critical
PassFuncCode [4]:                   FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.5.5 How to define a ETH resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtETH agent manages the resource type ETH. Thus, the syntax to add a new FSYS resource to the OSC configuration database using the OSC$CFG utility is:

```
OSC$CFG> ADD RESOURCE ETH::resource-name[@node-name]
```

10.6 OscAgtIP – OSC service IP agent

10.6.1 Description

The OscAgtIP agent is designed to manage service IP addresses as **On-Off** resources without utilizing the TCP/IP failSAFE IP. The resource type managed by the agent is IP.

The OscAgtIP agent is a compiled agent. The agent's image is:

- OSC$BIN:OSCAGT$IP.EXE

When the OscAgtIP agent brings a resource online it assigns the service IP address defined by the **IpAddress** resource attribute to the first interface defined by **IpInterfaces** resource attribute. The monitor action routine periodically checks the operational state of the interfaces defined by the **IpInterfaces** attribute. Only two IP interfaces can be defined by the **IpInterface** attribute. If the interface that has the service IP address assigned fails, the service IP address is removed from that interface and it is failed over to the standby interface. If both interfaces fail the service IP address resource is marked faulted.

It is strongly recommended that all IP interfaces have private IP addresses assigned if you utilize the OscAgtIP agent to manage service IP address.

Note

The operational state of the IP interfaces is detected by monitoring the NIC's "Bytes received" counter comparable to the way the TCP/IP failSAFE IP service monitors the interface. However, in a quiet network, there may be insufficient traffic to keep the "Bytes received" counter changing within the monitoring interval, thus causing phantom failure. Please note that the OscAgtIP agent does not send broadcast packets like the TCP/IP failSAFE IP service does to counteract this.

The OscAgtIP agent explicitely removes an IP address from an interface if the interface fails and assigns the IP address to the failover interface. An explicit IP address failover is not transparent to active TCP/IP sessions. Thus, if the OscAgtI agent performs a local IP address switch the TCP/IP sessions using this IP address will be disconnected.

The TCP/IP failSAFE IP service can provide an implicit IP addess failover

capability transparent for all active TCP/IP sessions on the same node.

Thus, it is recommended to utilize the OscAgtFailIP agent for service IP address management instead of the OscAgtIP agent, except where the TCPIP/IP failSAFE IP service cannot be enabled and started on all OSC cluster members.

10.6.2 Default agent attributes

```
    AGENT:      OSCAGTIP

Attributes [Size]            Values
----------------------------------
AgentType [16]:              IP
AgentName [32]:              OscAgtIP
AgentDescription [64]:       OSC agent to monitor IP link
status and assignement
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$IP.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          24
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTIP
```

For detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.6.3 Resource type specific attributes

Attribute Name	Data Type	Description
IpAddress	String[32]	Mandatory - Service IP address (i.e 10.10.38.79)

IpInterfaces	String[256]	Mandatory This attribute defines the IP interfaces the service IP address can be assigned to. Assign the IP interfaces as a comma separated list (i.e. IE0,IE1).
IpNetMask	String[32]	Mandatory - Network mask (i.e. 255.255.255.0)
IpBroadCast	String[32]	Mandatory - Broadcast address (i.e. 10.10.38.255)

10.6.4 Default common Resource attributes

The resource type template IP contains the resource attribute defaults for the resources managed by the OscAgtIP agent.

```
TEMPLATE:   IP                      Managed by: OscAgtIP

Attributes [Size]                   Values
---------------------------------------
Critical [4]:                       TRUE
Enabled [4]:                        TRUE
OnlineMonitorInterval [4]:          30
OfflineMonitorInterval [4]:         30
ToleranceLimit [4]:                 2
FaultOnMonitorTmo [4]:              TRUE
FaultOnMonitorTmoLimit [4]:         2
DisableMangeFault [4]:              FALSE
OnlineRetryLimit [4]:               0
OnlineWaitLimit [4]:                2
OnlineTmoWaitLimit [4]:             2
OfflineWaitLimit [4]:               2
OfflineTmoWaitLimit [4]:            2
RestartLimit [4]:                   0
CleanRetryLimit [4]:                5
TimeOutRetryLimit [4]:              5
ConfLimit [4]:                      600
MonitorScript [256]:
MonitorTmo [4]:                     10
OnlineScript [256]:
OnlineTmo [4]:                      300
OfflineScript [256]:
OfflineTmo [4]:                     300
CleanScript [256]:
CleanTmo [4]:                       60
OpenScript [256]:
OpenTmo [4]:                        60
ArgList [256]:                      IpAddress, IpInterfaces,
                                    IpNetMask, IpBroadCast
```

```
PassFuncCode [4]:                    FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.6.5 How to define a IP resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtIP agent manages the resource type IP. Thus, the syntax to add a new IP resource to the OSC configuration database using the OSC$CFG utility is:

OSC$CFG> ADD RESOURCE IP::resource-name[@node-name]

10.7 OscAgtFailIP – OSC failSAFE IP agent

10.7.1 Description

The OscAgtFailIP agent is designed to manage service IP addresses as **On-Off** resources utilizing the TCP/IP failSAFE IP service. It monitors the state, starts, stops and cleans up failSAFE IP sets. The resource type managed by the agent is FAILIP.

The OscAgtFailIP agent is a DCL script based agent. The agent runs the OSC agent framework image OSC$BIN:OSCAGT$FAILIP.EXE which provides the required OSC agent framework functionality. The DCL scripts listed below are called by the OSC agent framework to monitor, online, offline and cleanup an FAILIP resource:

- Open
  OSC$COMMON:[CFG.FAILIP]FAILSAFEIP_OPEN.COM
- Monitor
  OSC$COMMON:[CFG.FAILIP]FAILSAFEIP_MONITOR.COM
- Online
  OSC$COMMON:[CFG. FAILIP]FAILSAFEIP_ONLINE.COM
- Offline
  OSC$COMMON:[CFG. FAILIP]FAILSAFEIP_OFFLINE.COM
- CLEAN
  OSC$COMMON:[CFG. FAILIP]FAILSAFEIP_CLEAN.COM

When the OscAgtFailIP agent brings a resource online, it assigns the service IP address defined by the **IpAddress** resource attribute to the interfaces defined by **IpInterfaces** resource attribute. The TCP/IP failSAFE IP service manages IP address failover between these interfaces if required. If all interfaces of a failSAFE IP set fail the agent marks the resource as faulted. The TCP/IP failSAFE IP service does not failover the service IP address to an interface on another cluster member. This remains under full OSC control.

Prerequisite

The OscAgtFailIP agent can be utilized for service IP address management only if the TCP/IP failSAFE IP service is enabled and started. Thus, before you start OSC containing FAILIP resources check that the failSAFE IP service of TCP/IP is enabled and started on all OSC cluster members.

Note

The OscAgtFailIP agent is the recommended agent for service IP address management (see the explaination in section 10.6 OscAgtIP – OSC service IP agent) except if the failSAFE IP service cannot be enabled and started on your systems. In this case use the OscAgtIP agent for service IP address management.

10.7.2 Default agent attributes

```
AGENT:      OSCAGTFAILIP

Attributes [Size]            Values
----------------------------------
AgentType [16]:              FAILIP
AgentName [32]:              OscAgtFailIP
AgentDescription [64]:       failSAFE IP Monitoring Agent
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$FAILIP.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          10
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTFAILIP
```

For detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.7.3 Resource type specific attributes

Attribute Name	Data Type	Description
IpAddress	String[32]	Mandatory - Service IP address (i.e

		10.10.38.79)
IpInterfaces	String[256]	Mandatory This attribute defines the IP interfaces the service IP address can be assigned to. Assign the IP interfaces as a comma separated list (i.e. IE0,IE1).
IpNetMask	String[32]	Mandatory - Network mask (i.e. 255.255.255.0)
IpBroadCast	String[32]	Mandatory - Broadcast address (i.e. 10.10.38.255)

10.7.4 Default common Resource attributes

The resource type template FailIP contains the resource attribute defaults for the resources managed by the OscAgtFailIP agent.

```
TEMPLATE:   FAILIP                Managed by: OscAgtFailIP

Attributes [Size]                 Values
---------------------------------------
Critical [4]:                     TRUE
Enabled [4]:                      TRUE
OnlineMonitorInterval [4]:        30
OfflineMonitorInterval [4]:       60
ToleranceLimit [4]:               0
FaultOnMonitorTmo [4]:            TRUE
FaultOnMonitorTmoLimit [4]:       4
DisableMangeFault [4]:            FALSE
OnlineRetryLimit [4]:             0
OnlineWaitLimit [4]:              2
OnlineTmoWaitLimit [4]:           2
OfflineWaitLimit [4]:             2
OfflineTmoWaitLimit [4]:          2
RestartLimit [4]:                 0
CleanRetryLimit [4]:              5
TimeOutRetryLimit [4]:            5
ConfLimit [4]:                    600
MonitorScript [256]:
OSC$COMMON:[CFG.FAILIP]FAILSAFEIP_MONITOR.COM
MonitorTmo [4]:                   30
OnlineScript [256]:
OSC$COMMON:[CFG.FAILIP]FAILSAFEIP_ONLINE.COM
OnlineTmo [4]:                    30
OfflineScript [256]:
OSC$COMMON:[CFG.FAILIP]FAILSAFEIP_OFFLINE.COM
OfflineTmo [4]:                   30
```

```
CleanScript [256]:
OSC$COMMON:[CFG.FAILIP]FAILSAFEIP_CLEAN.COM
CleanTmo [4]:                    30
OpenScript [256]:
OSC$COMMON:[CFG.FAILIP]FAILSAFEIP_OPEN.COM
OpenTmo [4]:                     60
ArgList [256]:                   IpAddress, IpInterfaces,
                                 IpNetMask, IpBroadCast
PassFuncCode [4]:                FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.7.5 How to define a FailIP resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtFailIP agent manages the resource type FailIP. Thus, the syntax to add a new FailIP resource to the OSC configuration database using the OSC$CFG utility is:

OSC$CFG> ADD RESOURCE FailIP::resource-name[@node-name]

10.8 OscAgtORA – OSC Oracle 10 agent

10.8.1 Description

The OscAgtORA agent is designed to manage Oracle 10 databases as **On-Off** resources. It is capable of monitoring the state, to start, to stop and to cleanup a specific Oracle 10 database. The resource type managed by the agent is ORA.

The OscAgtORA agent is a DCL script based agent. The agent runs the generic OSC agent framework image OSC$BIN:OSCAGT$GENERIC.EXE which provides the required OSC agent framework functionality. The DCL scripts listed below are called by the OSC agent framework to monitor, online, offline and cleanup an ORA resource:

- Monitor
  OSC$COMMON:[CFG.ORA]ORACLE_MONITOR.COM
- Online
  OSC$COMMON:[CFG.ORA]ORACLE_ONLINE.COM
- Offline
  OSC$COMMON:[CFG.ORA]ORACLE_OFFLINE.COM
- CLEAN
  OSC$COMMON:[CFG.ORA]ORACLE_CLEAN.COM

Prerequisite

The OscAgtORA agent cannot manage an Oracle database if the database is configured to start on a dedicated node. If an Oracle database is configured to start on a dedicated OpenVMS OSC cluster member any attempt to switch-over/failover the database to another OSC cluster member will fail. Thus, make sure that the Oracle database is configured to start on all execution nodes of the OSC service group the Oracle resource is member of. Check the NODE property in the Oracle database properties file:

- sid_*OracleSID*.properties

It is best practice to use the no node restriction (comment out the NODE property in the properties file).

For more information about the properties file and the NODE property please refer to the appropriate Oracle documentation.

Note

The OscAgtORA agent cannot be utilized to monitor Oracle 8 or 9 databases with default agent attributes. The agent has to run under the UIC of the appropriate Oracle user. The default value of the ORA agent attribute **AgentUIC** is ORCALE10. Thus, if the OSC agent is to manage Oracle 9 databases the user ORACLE9 has to be assigned to the AgentUIC attribute. Either modify the default agent attributes using the OSC$CFG utility MODIFY AGENT command or create a new OSC agent managing Oracle 9 databases. The agent and resource type specific attributes will be the same as the Oracle 10 agent and resource type specific attributes except the user assigned to the **AgentUIC** attribute.

10.8.2 Default agent attributes

```
AGENT:      OSCAGTORA

Attributes [Size]            Values
-----------------------------------
AgentType [16]:              ORA
AgentName [32]:              OscAgtORA
AgentDescription [64]:       OSC Oracle 10 DB agent
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$GENERIC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               ORACLE10
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          48
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTORA
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.8.3 Resource category specific attributes

Attribute Name	Data Type	Description
OracleSID	String[32]	Mandatory - Oracle SID
OracleRootDir	String[128]	Mandatory - Oracle root directory.
OracleParaFile	String[256]	Optional - parameter file
OnlineMonitorDelay	Integer	Optional This attribute defines the time in seconds the OSC agent framework waits before calling the monitor action routine to verify if the online transaction succeeded. The default value is 5 seconds
ScriptDebug	Boolean	Optional This attribute defines whether or not the DCL action scripts insert debug output into the agent's log file.

10.8.4 Default common Resource attributes

The resource type template ORA contains the resource attribute defaults for the resources managed by the OscAgtORA agent.

```
TEMPLATE:   ORA                    Managed by: OscAgtORA

Attributes [Size]                  Values
---------------------------------------
Critical [4]:                      TRUE
Enabled [4]:                       TRUE
OnlineMonitorInterval [4]:         60
OfflineMonitorInterval [4]:        120
ToleranceLimit [4]:                0
FaultOnMonitorTmo [4]:             TRUE
FaultOnMonitorTmoLimit [4]:        4
DisableMangeFault [4]:             FALSE
OnlineRetryLimit [4]:              2
OnlineWaitLimit [4]:               2
OnlineTmoWaitLimit [4]:            2
OfflineWaitLimit [4]:              2
OfflineTmoWaitLimit [4]:           2
RestartLimit [4]:                  1
CleanRetryLimit [4]:               5
```

```
TimeOutRetryLimit [4]:            5
ConfLimit [4]:                    600
MonitorScript [256]:
OSC$COMMON:[CFG.ORA]ORACLE_MONITOR.COM
MonitorTmo [4]:                   45
OnlineScript [256]:
OSC$COMMON:[CFG.ORA]ORACLE_ONLINE.COM
OnlineTmo [4]:                    180
OfflineScript [256]:
OSC$COMMON:[CFG.ORA]ORACLE_OFFLINE.COM
OfflineTmo [4]:                   180
CleanScript [256]:
OSC$COMMON:[CFG.ORA]ORACLE_CLEAN.COM
CleanTmo [4]:                     180
OpenScript [256]:
OpenTmo [4]:                      60
ArgList [256]:                    OracleSID, OracleRootDir,
                                  OnlineMonitorDelay,
                                  ScriptDebug
PassFuncCode [4]:                 FALSE
OnlineMonitorDelay [4]:           5
ScriptDebug [4]:                  FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.8.5 How to define an ORA resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtORA agent manages the resource type ORA. Thus, the syntax to add a new ORA resource to the OSC configuration database using the OSC$CFG utility is:

OSC$CFG> ADD RESOURCE ORA::resource-name[@node-name]

10.10 OscAgtRDB - OSC RDB agent

10.10.1 Description

The OscAgtRDB agent is designed to manage RDB databases as **On-Off** resources. It is capable of monitoring the state, to start, to stop and to cleanup a specific RDB database. The resource type managed by the agent is RDB.

The OscAgtRDB agent is a DCL script based agent. The agent runs the generic OSC agent framework image OSC$BIN:OSCAGT$GENERIC.EXE which provides the required OSC agent framework functionality. The DCL scripts listed below are called by the OSC agent framework to monitor, online, offline and cleanup a RDB resource:

- Monitor OSC$COMMON:[CFG.RDB]RDB_MONITOR.COM
- Online OSC$COMMON:[CFG.RDB]RDB_ONLINE.COM
- Offline OSC$COMMON:[CFG.RDB]RDB_OFFLINE.COM
- Clean OSC$COMMON:[CFG.RDB]RDB_CLEAN.COM

Note

You can also utilize the OscAgtRDB agent to manage RDB database opened cluster-wide. In this case define the OSC resource that represents the cluster-wide open RDB database as **On-Only** (resource attribute **ResourceCategory**). Since the resource specific resource category overrules the agent's default resource category the OscAgtRDB agent does not shutdown cluster-wide RDB databases defined as **On-Only** when the service group that contains the RDB resource fails over to another execution node.

10.10.2 Default agent attributes

```
AGENT:      OSCAGTRDB

Attributes [Size]            Values
-----------------------------------
AgentType [16]:              RDB
AgentName [32]:              OscAgtRDB
AgentDescription [64]:       RDB database monitoring agent
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$GENERIC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
```

```
AgentHbtInterval [4]:         5
AgentHbtTimout [4]:           30
AgentStartTimeout [4]:        60
AgentShutdownTimeout [4]:     60
AgentAutoRestartLimit [4]:    5
AgentConfLimit [4]:           10
AgentMaxAction [4]:           10
AgentInformational [4]:       FALSE
AgentDebug [4]:               FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTRDB
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.10.3 Resource type specific attributes

Attribute Name	Data Type	Description
RootFile	String[256]	Mandatory This attribute defines the RDB root file. Either the .RDB file name or a logical that refers the .RDB file can be assigned.
RDMSScript	String[256]	Mandatory Oracle RDB monitor (RDMS_MONITOR) process startup script.
OnlineMonitorDelay	Integer	Optional This attribute defines the time in seconds the OSC agent framework waits before calling the monitor action routine to verify if the online transaction succeeded. The default value is 5 seconds
ScriptDebug	Boolean	Optional This attribute defines whether or not the DCL action scripts insert debug output into the agent's log file.

10.10.4 Default common Resource attributes

The resource type template RDB contains the resource attribute defaults for the resources managed by the OscAgtRDB agent.

```
TEMPLATE:   RDB                     Managed by: OscAgtRDB

Attributes [Size]                   Values
---------------------------------------
Critical [4]:                       TRUE
Enabled [4]:                        TRUE
OnlineMonitorInterval [4]:          60
OfflineMonitorInterval [4]:         180
ToleranceLimit [4]:                 0
FaultOnMonitorTmo [4]:              TRUE
FaultOnMonitorTmoLimit [4]:         4
DisableMangeFault [4]:              FALSE
OnlineRetryLimit [4]:               2
OnlineWaitLimit [4]:                2
OnlineTmoWaitLimit [4]:             2
OfflineWaitLimit [4]:               2
OfflineTmoWaitLimit [4]:            2
RestartLimit [4]:                   1
CleanRetryLimit [4]:                5
TimeOutRetryLimit [4]:              5
ConfLimit [4]:                      600
MonitorScript [256]:
OSC$COMMON:[CFG.RDB]RDB_MONITOR.COM
MonitorTmo [4]:                     45
OnlineScript [256]:
OSC$COMMON:[CFG.RDB]RDB_ONLINE.COM
OnlineTmo [4]:                      180
OfflineScript [256]:
OSC$COMMON:[CFG.RDB]RDB_OFFLINE.COM
OfflineTmo [4]:                     180
CleanScript [256]:
OSC$COMMON:[CFG.RDB]RDB_CLEAN.COM
CleanTmo [4]:                       180
OpenScript [256]:
OpenTmo [4]:                        60
ArgList [256]:                      RootFile, RDMSScript,
                                    OnlineMonitorDelay,
                                    ScriptDebug
PassFuncCode [4]:                   FALSE
OnlineMonitorDelay [4]:             5
ScriptDebug [4]:                    FALSE
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.10.5 How to define a RDB resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtRDB agent manages the resource type RDB. Thus, the syntax to add a new RDB resource to the OSC configuration database using the OSC$CFG utility is:

```
OSC$CFG> ADD RESOURCE RDB::resource-name[@node-name]
```

10.9 OscAgtORALS – OSC Oracle 10 Listener agent

10.9.1 Description

The OscAgtORALS agent is designed to manage Oracle 10 listener processes as **On-Off** resources. It is capable of monitoring the state, to start, to stop and to cleanup a particular Oracle 10 listener process. The resource type managed by the agent is ORALS.

The OscAgtORALS agent is a mixed compiled and DCL script based agent. The agent runs the OSC agent image OSC$BIN:OSCAGT$PRC.EXE which provides the required OSC agent framework functionality and the monitor action and clean action routines. The DCL scripts listed below are called by the OSC agent to online and offline an ORALS resource:

- Online
  OSC$COMMON:[CFG.ORA]ORA_LISTENER_ONLINE.COM
- Offline
  OSC$COMMON:[CFG.ORA]ORA_LISTENER_OFFLINE.COM

Prerequisite

The OscAgtORALS agent cannot manage an Oracle listener process if the HOST parameter in the listener.ora network configuration file refers to a node specific IP address. Check the listener.ora file and make sure that the HOST parameter is either not defined or the HOST parameter refers to a service IP address that can be reassigned to the IP interfaces of another cluster member, and that this service IP address is also handled by OSC as a child resource of the OSC Oracle listener resource. Otherwise any attempt to switchover/failover an Oracle listener process to another OSC cluster member will fail.

For more information about the properties file and the HOST parameter in listener.ora please refer to the appropriate Oracle documentation.

Note

The OscAgtORALS agent cannot be utilized to monitor Oracle 8 or 9 listener processes with default agent attributes. The agent has to run under the UIC of the appropriate oracle user. The default value of the ORALS agent attribute **AgentUIC** is ORCALE10. Thus, if the agent is to manage an Oracle 9 listener process the user ORACLE9 has to be

assigned to the **AgentUIC** attribute. Either modify the default agent attributes using the OSC$CFG utility MODIFY AGENT command or create a new OSC agent managing Oracle 9 listener processes. The agent and resource type specific attributes will be the same as the Oarcle 10 listener agent and resource type specific attributes except the user assigned to the **AgentUIC** attribute.

10.9.2 Default agent attributes

```
AGENT:      OSCAGTORALS

Attributes [Size]            Values
-----------------------------------
AgentType [16]:              ORALS
AgentName [32]:              OscAgtOraLS
AgentDescription [64]:       Oracle SID Listener Agent
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$PRC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               ORACLE10
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          10
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTORALS
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.9.3 Resource type specific attributes

Attribute Name	Data Type	Description
OracleLSNRPrc	String[32]	Mandatory Oracle listener process name. Full wildcard support is provided for this attribute. You can place asterisk (*) and percentage (%)

		wildcard characters anywhere within the process name string. If you are using wildcards make sure that only the Oracle listener process referenced by the OSC resource matches the wildcard string. Otherwise the OSC agent's behavior is unpredictable.
OracleLSNRPrcCnt	Integer	Optional Number of active Oracle listener processes. The default value is 1. **NOTE:** Do not modify the default value.
OracleLSNRSid	String[32]	Mandatory - Oracle SID
OracleHomeDir	String[128]	Mandatory - Oracle root directory.
MonitorDelayTime	Integer	Optional This attribute defines the time in seconds the OSC agent waits before calling the monitor action routine to verify if the online transaction succeeded. The default value is 9 seconds

10.9.4 Default common Resource attributes

The resource type template ORALS contains the resource attribute defaults for the resources managed by the OscAgtORALS agent.

```
TEMPLATE:   ORALS                  Managed by: OscAgtOraLS

Attributes [Size]                  Values
---------------------------------------
Critical [4]:                      TRUE
Enabled [4]:                       TRUE
OnlineMonitorInterval [4]:         30
OfflineMonitorInterval [4]:        30
ToleranceLimit [4]:                1
FaultOnMonitorTmo [4]:             TRUE
FaultOnMonitorTmoLimit [4]:        4
DisableMangeFault [4]:             FALSE
OnlineRetryLimit [4]:              1
OnlineWaitLimit [4]:               2
OnlineTmoWaitLimit [4]:            2
OfflineWaitLimit [4]:              2
OfflineTmoWaitLimit [4]:           2
RestartLimit [4]:                  1
CleanRetryLimit [4]:               5
```

```
TimeOutRetryLimit [4]:            5
ConfLimit [4]:                    600
MonitorScript [256]:
MonitorTmo [4]:                   60
OnlineScript [256]:
OSC$COMMON:[CFG.ORA]ORA_LISTENER_ONLINE.COM
OnlineTmo [4]:                    60
OfflineScript [256]:
OSC$COMMON:[CFG.ORA]ORA_LISTENER_OFFLINE.COM
OfflineTmo [4]:                   60
CleanScript [256]:
CleanTmo [4]:                     60
OpenScript [256]:
OpenTmo [4]:                      60
ArgList [256]:                    OracleLSNRPrc,
                                  OracleLSNRPrcCnt,
                                  OracleLSNRSid,
                                  OracleHomeDir,
                                  MonitorDelayTime
PassFuncCode [4]:                 FALSE
MonitorDelayTime [4]:             9
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.9.5 How to define an ORALS resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtORALS agent manages the resource type ORALS. Thus, the syntax to add a new ORALS resource to the OSC configuration database using the OSC$CFG utility is:

OSC$CFG> ADD RESOURCE ORALS::resource-name[@node-name]

10.11 OscAgtPERF - OSC VSI PERFDAT agent

10.11.1 Description

The OscAgtPERF agent is designed to manage VSI PERFDAT OpenVMS data collections as **On-Off** resources. It is capable of monitoring the state, to start, to stop and to cleanup a particular VSI PERFDAT OpenVMS data collection. The resource type managed by the agent is PERF.

The OscAgtPERF agent is a DCL script based agent. The agent runs the generic OSC agent framework image OSC$BIN:OSCAGT$GENERIC.EXE which provides the required OSC agent framework functionality. The DCL scripts listed below are called by the OSC agent framework to monitor, online, offline and cleanup a PERF resource:

- Monitor
  OSC$COMMON:[CFG.PERFDAT]PERF_COLL_MONITOR.COM
- Online
  OSC$COMMON:[CFG.PERFDAT]PERF_COLL_ONLINE.COM
- Offline
  OSC$COMMON:[CFG.PERFDAT]PERF_COLL_OFFLINE.COM
- Clean
  OSC$COMMON:[CFG.PERFDAT]PERF_COLL_CLEAN.COM

10.11.2 Default agent attributes

```
AGENT:      OSCAGTPERF

Attributes [Size]            Values
---------------------------------
AgentType [16]:              PERF
AgentName [32]:              OscAgtPERF
AgentDescription [64]:       VSI PERFDAT OSC agent
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$GENERIC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          1
AgentInformational [4]:      FALSE
```

```
AgentDebug [4]:                FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTPERF
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.11.3 Resource type specific attributes

Attribute Name	Data Type	Description
CollectProfile	String[32]	Mandatory This attribute defines the VSI PERFDAT collection profile to be managed. For detailed information about VSI PERFDAT OpenVMS performance data collections please refer to the documentation of VSI PERFDAT.
OnlineMonitorDelay	Integer	Optional This attribute defines the time in seconds the OSC agent framework waits before calling the monitor action routine to verify if the online transaction succeeded. The initialization phase of a VSI PERFDAT OpenVMS data collection may last from few seconds up to minutes. Thus, check if the default value of 45 seconds is sufficient on your systems.

10.11.4 Default common Resource attributes

The resource type template PERF contains the resource attribute defaults for the resources managed by the OscAgtPERF agent.

```
TEMPLATE:   PERF                    Managed by: OscAgtPERF

Attributes [Size]                   Values
----------------------------------------
```

```
Critical [4]:                       TRUE
Enabled [4]:                        TRUE
OnlineMonitorInterval [4]:          60
OfflineMonitorInterval [4]:         30
ToleranceLimit [4]:                 0
FaultOnMonitorTmo [4]:              TRUE
FaultOnMonitorTmoLimit [4]:         4
DisableMangeFault [4]:              FALSE
OnlineRetryLimit [4]:               1
OnlineWaitLimit [4]:                1
OnlineTmoWaitLimit [4]:             2
OfflineWaitLimit [4]:               2
OfflineTmoWaitLimit [4]:            2
RestartLimit [4]:                   1
CleanRetryLimit [4]:                5
TimeOutRetryLimit [4]:              5
ConfLimit [4]:                      600
MonitorScript [256]:
OSC$COMMON:[CFG.PERFDAT]PERF_COLL_MONITOR.COM
MonitorTmo [4]:                     30
OnlineScript [256]:
OSC$COMMON:[CFG.PERFDAT]PERF_COLL_ONLINE.COM
OnlineTmo [4]:                      120
OfflineScript [256]:
OSC$COMMON:[CFG.PERFDAT]PERF_COLL_OFFLINE.COM
OfflineTmo [4]:                     120
CleanScript [256]:
OSC$COMMON:[CFG.PERFDAT]PERF_COLL_CLEAN.COM
CleanTmo [4]:                       45
OpenScript [256]:
OpenTmo [4]:                        60
ArgList [256]:                      CollectProfile,
                                    OnlineMonitorDelay
PassFuncCode [4]:                   FALSE
CollectionProfile [32]:
OnlineMonitorDelay [4]:             45
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.11.5 How to define a PERF resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To

define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtPERF agent manages the resource type PERF. Thus, the syntax to add a new PERF resource to the OSC configuration database using the OSC$CFG utility is:

```
OSC$CFG> ADD RESOURCE PERF::resource-name[@node-name]
```

10.12 OscAgtSQLSRV - OSC SQL service agent

10.12.1 Description

The OscAgtSQLSRV agent is designed to manage SQL services as **On-Off** resources. It is capable of monitoring the state, to start, to stop and to cleanup a particular SQL service. The resource type managed by the agent is SQLSRV.

The OscAgtSQLSRV agent is a DCL script based agent. The agent runs the generic OSC agent framework image OSC$BIN:OSCAGT$GENERIC.EXE which provides the required OSC agent framework functionality. The DCL scripts listed below are called by the OSC agent framework to monitor, online, offline and cleanup a SQLSRV resource:

- Monitor
  OSC$COMMON:[CFG.SQLSRV]SQLSRV_MONITOR.COM
- Online
  OSC$COMMON:[CFG.SQLSRV]SQLSRV_ONLINE.COM
- Offline
  OSC$COMMON:[CFG.SQLSRV]SQLSRV_OFFLINE.COM
- Clean
  OSC$COMMON:[CFG.SQLSRV]SQLSRV_CLEAN.COM

10.12.2 Default agent attributes

```
AGENT:      OSCAGTSQLSRV

Attributes [Size]            Values
---------------------------------
AgentType [16]:              SQLSRV
AgentName [32]:              OscAgtSQLSRV
AgentDescription [64]:       OSC SQL service agent
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$GENERIC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          1
AgentInformational [4]:      FALSE
```

```
AgentDebug [4]:                FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTSQLSRV
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.12.3 Resource type specific attributes

Attribute Name	Data Type	Description
SqlService	String[32]	Mandatory This attribute defines the name of the SQL service to be managed.
MonitorDelay	Integer	Optional This attribute defines the time in seconds the OSC agent framework waits before calling the monitor action routine to verify if the preceeding online or offline transaction succeeded.

10.12.4 Default common Resource attributes

The resource type template SQLSRV contains the resource attribute defaults for the resources managed by the OscAgtSQLSRV agent.

```
TEMPLATE:   SQLSRV                 Managed by: OscAgtSQLSRV

Attributes [Size]                  Values
---------------------------------------
Critical [4]:                      TRUE
Enabled [4]:                       TRUE
OnlineMonitorInterval [4]:         60
OfflineMonitorInterval [4]:        60
ToleranceLimit [4]:                3
FaultOnMonitorTmo [4]:             TRUE
FaultOnMonitorTmoLimit [4]:        1
DisableMangeFault [4]:             FALSE
OnlineRetryLimit [4]:              1
OnlineWaitLimit [4]:               1
OnlineTmoWaitLimit [4]:            1
```

```
OfflineWaitLimit [4]:              1
OfflineTmoWaitLimit [4]:           1
RestartLimit [4]:                  2
CleanRetryLimit [4]:               5
TimeOutRetryLimit [4]:             5
ConfLimit [4]:                     600
MonitorScript [256]:
OSC$COMMON:[CFG.PERFDAT]SQLSRV_MONITOR.COM
MonitorTmo [4]:                    45
OnlineScript [256]:
OSC$COMMON:[CFG.PERFDAT]SQLSRV_ONLINE.COM
OnlineTmo [4]:                     30
OfflineScript [256]:
OSC$COMMON:[CFG.PERFDAT]SQLSRV_OFFLINE.COM
OfflineTmo [4]:                    30
CleanScript [256]:
OSC$COMMON:[CFG.PERFDAT]SQLSRV_CLEAN.COM
CleanTmo [4]:                      45
OpenScript [256]:
OpenTmo [4]:                       60
ArgList [256]:                     SqlService,MonitorDelay
PassFuncCode [4]:                  FALSE
SqlService [32]:
MonitorDelay [4]:                  1
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.12.5 How to define a SQLSRV resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtSQLSRV agent manages the resource type SQLSRV. Thus, the syntax to add a new SQLSRV resource to the OSC configuration database using the OSC$CFG utility is:

OSC$CFG> ADD RESOURCE SQLSRV::resource-name[@node-name]

10.13 OscAgtSQLDIS - OSC SQL dispatcher agent

10.13.1 Description

The OscAgtSQLSRV agent is designed to manage SQL dispatcher as **On-Off** resources. It is capable of monitoring the state, to start, to stop and to cleanup a particular SQL dispatcher. The resource type managed by the agent is SQLDISP.

The OscAgtSQLDIS agent is a DCL script based agent. The agent runs the generic OSC agent framework image OSC$BIN:OSCAGT$GENERIC.EXE which provides the required OSC agent framework functionality. The DCL scripts listed below are called by the OSC agent framework to monitor, online, offline and cleanup a SQLDISP resource:

- Monitor
  OSC$COMMON:[CFG.SQLSRV]SQLDISP_MONITOR.COM
- Online
  OSC$COMMON:[CFG.SQLSRV]SQLDISP_ONLINE.COM
- Offline
  OSC$COMMON:[CFG.SQLSRV]SQLDISP_OFFLINE.COM
- Clean
  OSC$COMMON:[CFG.SQLSRV]SQLDISP_CLEAN.COM

10.13.2 Default agent attributes

```
AGENT:      OSCAGTSQLDIS

Attributes [Size]            Values
---------------------------------
AgentType [16]:              SQLDISP
AgentName [32]:              OscAgtSQLDIS
AgentDescription [64]:       OSC SQL dispatcher agent
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$GENERIC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          1
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTSQLDIS
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.13.3 Resource type specific attributes

Attribute Name	Data Type	Description
SqlSrvDispatcher	String[32]	Mandatory This attribute defines the name of the SQL dispatcher to be managed.
MonitorDelay	Integer	Optional This attribute defines the time in seconds the OSC agent framework waits before calling the monitor action routine to verify if the preceding online or offline transaction succeeded.

10.13.4 Default common Resource attributes

The resource type template SQLDISP contains the resource attribute defaults for the resources managed by the OscAgtSQLDIS agent.

```
TEMPLATE:   SQLDISP                Managed by: OscAgtSQLDIS

Attributes [Size]                  Values
---------------------------------------
Critical [4]:                      TRUE
Enabled [4]:                       TRUE
OnlineMonitorInterval [4]:         60
OfflineMonitorInterval [4]:        60
ToleranceLimit [4]:                3
FaultOnMonitorTmo [4]:             TRUE
FaultOnMonitorTmoLimit [4]:        1
DisableMangeFault [4]:             FALSE
OnlineRetryLimit [4]:              1
OnlineWaitLimit [4]:               1
OnlineTmoWaitLimit [4]:            1
OfflineWaitLimit [4]:              1
```

```
OfflineTmoWaitLimit [4]:          1
RestartLimit [4]:                 2
CleanRetryLimit [4]:              5
TimeOutRetryLimit [4]:            5
ConfLimit [4]:                    600
MonitorScript [256]:
OSC$COMMON:[CFG.PERFDAT]SQLDISP_MONITOR.COM
MonitorTmo [4]:                   45
OnlineScript [256]:
OSC$COMMON:[CFG.PERFDAT]SQLDISP_ONLINE.COM
OnlineTmo [4]:                    30
OfflineScript [256]:
OSC$COMMON:[CFG.PERFDAT]SQLDISP_OFFLINE.COM
OfflineTmo [4]:                   30
CleanScript [256]:
OSC$COMMON:[CFG.PERFDAT]SQLDISP_CLEAN.COM
CleanTmo [4]:                     45
OpenScript [256]:
OpenTmo [4]:                     .60
ArgList [256]:                    SqlSrvDispatcher,
                                  MonitorDelay
PassFuncCode [4]:                 FALSE
SqlSrvDispatcher [32]:
MonitorDelay [4]:                 1
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.13.5 How to define a SQLDISP resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-type::resource-name[@node-name]

The resource-type string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtSQLDIS agent manages the resource type SQLDISP. Thus, the syntax to add a new SQLDISP resource to the OSC configuration database using the OSC$CFG utility is:

OSC$CFG> ADD RESOURCE SQLDISP::resource-name[@node-name]

10.14 OscAgtMYSQL - OSC MySQL agent

10.14.1 Description

The OscAgtMYSQL agent is designed to manage MySQL databases as **On-Off** resources. It is capable of monitoring the state, to start, to stop and to cleanup a particular MySQL database. The resource type managed by the agent is MYSQL.

The OscAgtMYSQL agent is a mixed compiled and DCL script based agent. The agent runs the OSC agent image OSC$BIN:OSCAGT$PRC.EXE which provides the required OSC agent framework functionality and the clean action routines. The DCL scripts listed below are called by the OSC agent to monitor, online and offline a MYSQL resource:

- Monitor
  OSC$COMMON:[CFG.MYSQL]MYSQL_MONITOR.COM
- Online
  OSC$COMMON:[CFG.MYSQL]MYSQL_ONLINE.COM
- Offline
  OSC$COMMON:[CFG.MYSQL]MYSQL_OFFLINE.COM

10.14.2 Default agent attributes

```
AGENT:      OSCAGTMYSQL

Attributes [Size]            Values
---------------------------------
AgentType [16]:              MYSQL
AgentName [32]:              OscAgtMYSQL
AgentDescription [64]:       OSC agent for MySQL
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$PRC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          1
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTMYSQL
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.14.3 Resource type specific attributes

Attribute Name	Data Type	Description
PrcName	String[32]	Mandatory Process name of the MySQL server (i.e. MYSQL051_SERVER = MySQL server V5.1)
PrcCount	Integer	Optional Number of active MySQL server processes. The default value is 1. **NOTE:** Do not modify the default value
MySQLPwd	String[32]	Mandatory MySQL server password. **NOTE:** The MySQL server password is case sensitive. Use quotation marks when you enter the password.
MySQLRootDir	Integer	Mandatory MySQL root directory.

10.14.4 Default common Resource attributes

The resource type template MYSQL contains the resource attribute defaults for the resources managed by the OscAgtMYSQL agent.

```
TEMPLATE:   MYSQL                  Managed by: OscAgtMYSQL

Attributes [Size]                  Values
----------------------------------------
Critical [4]:                      TRUE
Enabled [4]:                       TRUE
```

```
OnlineMonitorInterval [4]:        120
OfflineMonitorInterval [4]:       300
ToleranceLimit [4]:               2
FaultOnMonitorTmo [4]:            TRUE
FaultOnMonitorTmoLimit [4]:       4
DisableMangeFault [4]:            FALSE
OnlineRetryLimit [4]:             1
OnlineWaitLimit [4]:              2
OnlineTmoWaitLimit [4]:           2
OfflineWaitLimit [4]:             2
OfflineTmoWaitLimit [4]:          2
RestartLimit [4]:                 1
CleanRetryLimit [4]:              5
TimeOutRetryLimit [4]:            5
ConfLimit [4]:                    600
MonitorScript [256]:
OSC$COMMON:[CFG.MYSQL]MYSQL_MONITOR.COM
MonitorTmo [4]:                   15
OnlineScript [256]:
OSC$COMMON:[CFG.MYSQL]MYSQL_ONLINE.COM
OnlineTmo [4]:                    20
OfflineScript [256] ]:
OSC$COMMON:[CFG.MYSQL]MYSQL_OFFLINE.COM
OfflineTmo [4]:                   20
CleanScript [256
CleanTmo [4]:                     30
OpenScript [256]:
OpenTmo [4]:                      60
ArgList [256]:                    PrcName, PrcCount
                                  MySQLPwd, MySQLRootDir
PassFuncCode [4]:                 FALSE
PrcName [32]:
PrcCount [4]:                     1
MySQLPwd [32]:
MySQLRootDir [64]:
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.14.5 How to define a MYSQL resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-catgeory::resource-name[@node-name]

The resource-catgeory string defines the resource type of the resource. The resource-name string defines the name of the resource. The resource-type and the resource-name string must be separated by a double colon.

To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtMYSQL agent manages the resource category MYSQL. Thus, the syntax to add a new MYSQL resource to the OSC configuration database using the OSC$CFG utility is:

```
OSC$CFG> ADD RESOURCE MYSQL::resource-name[@node-name]
```

10.15 OscAgtDECnet - OSC DECnet alias agent

10.15.1 Description

The OscAgtDECnet agent is designed to manage DECnet PhaseV aliases as **On-Off** resources. It is capable of monitoring the state and to assign and deassign a particular DECnet PhaseV alias. The resource type managed by the agent is DECNET.

The OscAgtDECnet agent is a DCL script based agent. The agent runs the generic OSC agent framework image OSC$BIN:OSCAGT$GENERIC.EXE which provides the required OSC agent framework functionality. The DCL scripts listed below are called by the OSC agent framework to monitor, online, offline and cleanup a DECnet alias resource:

- Monitor
  OSC$COMMON:[CFG.DECNET]DECNET_ALIAS_MONITOR.COM
- Online
  OSC$COMMON:[CFG.DECNET]DECNET_ALIAS_ONLINE.COM
- Offline
  OSC$COMMON:[CFG.DECNET]DECNET_ALIAS_OFFLINE.COM
- Clean
  OSC$COMMON:[CFG.DECNET]DECNET_ALIAS_OFFLINE.COM

When this OSC agent triggers either the offline or the clean action routine the same script is executed.

10.15.2 Default agent attributes

```
AGENT:      OSCAGTDECNET

Attributes [Size]            Values
---------------------------------
AgentType [16]:              DECNET
AgentName [32]:              OscAgtDECnet
AgentDescription [64]:       OSC DECnet alias agent
AgentCategory [16]:          On-Off
AgentImage [256]:            OSC$BIN:OSCAGT$GENERIC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
```

```
AgentMaxAction [4]:           1
AgentInformational [4]:       FALSE
AgentDebug [4]:               FALSE
```

To modify the agent's attributes use the OSC$MGR command:

```
OSC$CFG> MODIFY AGENT OSCAGTDECNET
```

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.15.3 Resource category specific attributes

Attribute Name	Data Type	Description
DECnetAlias	String[32]	Mandatory DECnet PhaseV alias port (i.e. LOCAL:.VNOTSC). For detailed information about the DECnet alias port please refer to the DECnet PhaseV documentation).
DECnetNodeID	String[32]	Mandatory DECnet PhaseV alias port node ID (i.e. AA-00-04-00-FE-C2). For detailed information about the DECnet alias port node ID please refer to the DECnet PhaseV documentation).

10.15.4 Default common Resource attributes

The resource category template DECNET contains the resource attribute defaults for the resources managed by the OscAgtDECnet agent.

```
TEMPLATE:   DECNET                  Managed by: OscAgtDECnet

Attributes [Size]                  Values
-----------------------------------------
Critical [4]:                      TRUE
Enabled [4]:                       TRUE
OnlineMonitorInterval [4]:         60
OfflineMonitorInterval [4]:        60
ToleranceLimit [4]:                1
FaultOnMonitorTmo [4]:             TRUE
```

```
FaultOnMonitorTmoLimit [4]:       4
DisableMangeFault [4]:            FALSE
OnlineRetryLimit [4]:             1
OnlineWaitLimit [4]:              1
OnlineTmoWaitLimit [4]:           1
OfflineWaitLimit [4]:             1
OfflineTmoWaitLimit [4]:          1
RestartLimit [4]:                 1
CleanRetryLimit [4]:              5
TimeOutRetryLimit [4]:            5
ConfLimit [4]:                    600
MonitorScript [256]:
OSC$COMMON:[CFG.DECNET]DECNET_ALIAS_MONITOR.COM
MonitorTmo [4]:                   15
OnlineScript [256]:
OSC$COMMON:[CFG.DECNET]DECNET_ALIAS_ONLINE.COM
OnlineTmo [4]:                    15
OfflineScript [256]:
OSC$COMMON:[CFG.DECNET]DECNET_ALIAS_OFFLINE.COM
OfflineTmo [4]:                   15
CleanScript [256]:
OSC$COMMON:[CFG.DECNET]DECNET_ALIAS_OFFLINE.COM
CleanTmo [4]:                     15
OpenScript [256]:
OpenTmo [4]:                      15
ArgList [256]:                    DECnetAlias, DECnetNodeID
PassFuncCode [4]:                 FALSE
DECnetAlias [32]:
DECnetNodeID [32]:
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.15.5 How to define a DECNET alias resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-category::resource-name[@node-name]

The resource-category string defines the resource category of the resource. The resource-name string defines the name of the resource. The resource-category and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtDECnet agent manages the resource category DECNET. Thus, the syntax to add a new DECNET alias resource to the OSC configuration database using the OSC$CFG utility is:

```
OSC$CFG> ADD RESOURCE DECNET::resource-name[@node-name]
```

10.16 OscAgtNCLObj - OSC DECnet object agent

10.16.1 Description

The OscAgtNCLObj agent is designed to manage DECnet PhaseV objects as **On-Only** resources. It is capable of monitoring the state and to define a particular DECnet PhaseV object. The resource type managed by the agent is NCLOBJ.

The OscAgtNCLObj agent is a DCL script based agent. The agent runs the generic OSC agent framework image OSC$BIN:OSCAGT$GENERIC.EXE which provides the required OSC agent framework functionality. The DCL scripts listed below are called by the OSC agent framework to monitor and to online a NCLOBJ resource (no offline or cleanup scripts exist since the agent manages **On-Only** resources):

- Monitor
  OSC$COMMON:[CFG.DECNET]NCLOBJ_MONITOR.COM
- Online
  OSC$COMMON:[CFG.DECNET]NCLOBJ_ONLINE.COM

10.16.2 Default agent attributes

```
AGENT:      OSCAGTNCLOBJ

Attributes [Size]            Values
---------------------------------
AgentType [16]:              NCLOBJ
AgentName [32]:              OscAgtNCLObj
AgentDescription [64]:       OSC agent managing NCL objects
AgentCategory [16]:          On-Only
AgentImage [256]:            OSC$BIN:OSCAGT$GENERIC.EXE
AgentShutdown [256]:
AgentPriority [4]:           8
AgentUIC [32]:               SYSTEM
AgentHbtInterval [4]:        5
AgentHbtTimout [4]:          30
AgentStartTimeout [4]:       60
AgentShutdownTimeout [4]:    60
AgentAutoRestartLimit [4]:   5
AgentConfLimit [4]:          10
AgentMaxAction [4]:          1
AgentInformational [4]:      FALSE
AgentDebug [4]:              FALSE
```

To modify the agent's attributes use the OSC$MGR command:

OSC$CFG> MODIFY AGENT OSCAGTNCLOBJ

For a detailed description of the agent attributes please refer to A.5 OSC agent attributes.

10.16.3 Resource category specific attributes

Attribute Name	Data Type	Description
NCLObjName	String[32]	Mandatory This attribute defines the name of the DECnet PhaseV object to be managed.
NCLObjStartScript	String[256]	Optional This attribute defines the NCL script that will be executed by the online action script to setup and start the DECnet PhaseV object.
OnlineMonitorDelay	Integer	Optional This attribute defines the time in seconds the OSC agent framework waits before calling the monitor action routine to verify if the online transaction succeeded.

10.16.4 Default common Resource attributes

The resource category template NCLOBJ contains the resource attribute defaults for the resources managed by the OscAgtNCLObj agent.

```
TEMPLATE:   NCLOBJ                  Managed by: OscAgtNCLObj

Attributes [Size]                  Values
-----------------------------------------
Critical [4]:                      TRUE
Enabled [4]:                       TRUE
OnlineMonitorInterval [4]:         30
OfflineMonitorInterval [4]:        60
ToleranceLimit [4]:                1
FaultOnMonitorTmo [4]:             TRUE
FaultOnMonitorTmoLimit [4]:        4
DisableMangeFault [4]:             FALSE
```

```
OnlineRetryLimit [4]:              1
OnlineWaitLimit [4]:               2
OnlineTmoWaitLimit [4]:            2
OfflineWaitLimit [4]:              2
OfflineTmoWaitLimit [4]:           2
RestartLimit [4]:                  1
CleanRetryLimit [4]:               5
TimeOutRetryLimit [4]:             5
ConfLimit [4]:                     600
MonitorScript [256]:
OSC$COMMON:[CFG.DECNET]NCLOBJ_MONITOR.COM
MonitorTmo [4]:                    30
OnlineScript [256]:
OSC$COMMON:[CFG.DECNET]NCLOBJ_ONLINE.COM
OnlineTmo [4]:                     30
OfflineScript [256]:
OfflineTmo [4]:                    30
CleanScript [256]:
CleanTmo [4]:                      30
OpenScript [256]:
OpenTmo [4]:                       30
ArgList [256]:                     NCLObjName,
                                   NCLObjStartScript,
                                   OnlineMonitorDelay
PassFuncCode [4]:                  FALSE
NCLObjName [32]:
NCLObjStartScript [256]:
OnlineMonitorDelay [4]:            3
```

For a detailed description of the common resource attributes and how they affect OSC behavior please refer to 5 Controlling OSC behavior and A.7 OSC resource attributes

10.16.5 How to define a NCLOBJ resource

OSC resources have to be defined according to the formatting rule shown below:

Resource-category::resource-name[@node-name]

The resource-category string defines the resource category of the resource. The resource-name string defines the name of the resource. The resource-category and the resource-name string must be separated by a double colon. To define a resource to be node specific the node name has to be appended to the resource name string with the @ prefix.

The OscAgtNCLObj agent manages the resource category NCLOBJ. Thus, the syntax to add a new NCLOBJ resource to the OSC configuration database using the OSC$CFG utility is:

OSC simulation

This section provides detailed information about the OSC simulation mode.

VSI OpenVMS ServiceControl can either be started in control (normal opration) mode or in simulation mode.

When VSI OpenVMS ServiceControl is started in control mode OSC takes full control of all resources, services and service groups as defined in the OSC configuration datatabase used by the OSC components.

If VSI OpenVMS ServiceControl is started in simulation mode the OSC agents do not execute the resource type specific commands, scripts or compiled functions whenever one of the action routines listed below are triggered by the OSC framework (see also 4.3.1 OSC Agents and 9.3 Action routines):

- Open
- Monitor
- Online
- Offline
- Clean

Instead the OSC agents execute the scripts listed in Tab. 11.1.

Tab. 11.1 Action scripts executed by the OSC agents in simulation mode

Action triggerd	Script executed
Open	OSC$COMMON:[CFG.TEST]TEST_OPEN.COM
Monitor	OSC$COMMON:[CFG.TEST]TEST_MONITOR.COM
Online	OSC$COMMON:[CFG.TEST]TEST_ONLINE.COM
Offline	OSC$COMMON:[CFG.TEST]TEST_OFFLINE.COM
Clean	OSC$COMMON:[CFG.TEST]TEST_CLEAN.COM

Common to these test action routines is that they just define and test system-wide logicals. The names of these system-wide logicals are the names of the resources defined in the OSC configuration database in use. For examples if a resource named SHD::PERFDAT exists in the OSC configuration database, OSC creates the logical name SHD::PERFDAT when it is started in simulation mode.

Starting the OSC environment in simulation mode is usefull for:

- Training of how to manage OSC
- Testing a new OSC configuration
- Learning how OSC behaves in particular fault scenarios

without affecting the applications (service groups) already running on the OpenVMS cluster.

11.1 OSC test open script

As described in section 4.3.1.2 OSC Agent Operations and 9.3.2 Open the open action routine is called by an OSC agent to initialize a resource. When OSC is started in simulation mode the OSC test open script creates the system wide logical for a resource.

If the system-wide logical that represents a resource does not exist, the OSC test open script initializes the system-wide logical depending on the resource type as listed in Tab. 11.2:

Tab. 11.2 Action scripts executed by the OSC agents in simulation mode

Resource type	Initial value of the corresponding system-wide logical
On-Off	OFFLINE
On-Only	OFFLINE
Persistent	ONLINE

Script: OSC$COMMON:[CFG.TEST]TEST_OPEN.COM

```
$! OSC Test open script
$!
$! Any OSC agent passes the virtual resource name in P1
$! to this routine. This routine checks if a system-wide logical
```

```
$! as passed in P1 exists.
$! If not this routine defines the logical and assigns the values
$! listed below depending on the value passed in P2
$!
$!      P2                  Value assigned
$!      ------------------------------------------------------------------
$!      ONOFF               OFFLINE
$!      ONONLY              OFFLINE
$!      PERSISTENT           ONLINE
$!
$! ------------------------------------------------------------------------
$!
$! Return Codes
$!
$ OSC$_ONLINE  = 1
$ OSC$_OFFLINE = 9
$ OSC$_FAULTED = 19
$ OSC$_ERROR   = 12
$!
$!
$! V1.0        W.Burger / 19.6.2008
$!      initial Version
$!
$!
$!==============================================================
$!
$ Logical = f$trnlnm (P1)
$!
$! write sys$output "''f$time()': OPEN Call, Resource: ''P1', Cat: ''P2'"
$ if (Logical .eqs. "")
$ then
$       if (P2 .eqs. "PERSISTENT")
$       then
$               define/system 'P1 "ONLINE"
$               exit (OSC$_ONLINE)
$       endif
$       define/system 'P1 "OFFLINE"
$       exit (OSC$_ONLINE)
$ endif
$!
```

11.2 OSC test monitor script

As described in section 4.3.1.2 OSC Agent Operations and 9.3.2 Open

The OSC agent framework calls the open action routine whenever the OSC agent starts managing a resource:

- When the OSC agent starts
- When a disabled resource is enabled again.

When an OSC agent starts, it is guaranteed that the open action routine for each managed resource is called before its Monitor, Online, Offline or Clean action routine. This allows you to include initializations for specific resources. Most OSC agents do not require this functionality and will not implement this entry point. The OSC failSAFE IP agent (see 10.7 OscAgtFailIP – OSC failSAFE IP agent) is the only OSC agent of the bundled OSC agents that executes an open action routine for each managed resource.

The parameters passed to the open action routines are:

- the resource name
- resource type specific parameters defined by the common **ArgList** resource attribute required by the monitor action routine for execution

The open action routine always provides return code OSC$_ONLINE (return code value: 1)

9.3.3 Monitor the monitor action routine is called by an OSC agent to to determine the state of a resource. As described in previous sections OSC triggers control actions based on the return codes received from the monitor action routine called. Valid return codes are (see also 9.3.2 Open

The OSC agent framework calls the open action routine whenever the OSC agent starts managing a resource:

- When the OSC agent starts
- When a disabled resource is enabled again.

When an OSC agent starts, it is guaranteed that the open action routine for each managed resource is called before its Monitor, Online, Offline or Clean action routine. This allows you to include initializations for specific resources. Most OSC agents do not require this functionality and will not implement this entry point. The OSC failSAFE IP agent (see 10.7 OscAgtFailIP – OSC failSAFE IP agent) is the only OSC agent of the bundled OSC agents that executes an open action routine for each managed resource.

The parameters passed to the open action routines are:

- the resource name

- resource type specific parameters defined by the common **ArgList** resource attribute required by the monitor action routine for execution

The open action routine always provides return code OSC$_ONLINE (return code value: 1)

9.3.3 Monitor):

- OSC$_ONLINE return code value: 1
- OSC$_OFFLINE return code value: 9
- OSC$_FAULTED return code value: 19
- Any valid OpenVMS error code

The OSC test monitor script tests the value assigned to the system-wide logical that represents the OSC resource and returns one of the return codes listed above depending on the logicals value as listed in Tab. 11.3.

Tab. 11.3 Logical name value to return code mapping table for the test monitor script

Logical name value	Return code
Logical does not exist	OSC$_OFFLINE The OSC agent will consider the resource as offline.
OFFLINE	OSC$_OFFLINE The OSC agent will consider the resource as offline.
OFFLINE_WAIT	OSC$_OFFLINE The OSC agent will consider the resource as offline.
ONLINE	OSC$_ONLINE The OSC agent will consider the resource as online.
OFFLINE_WAIT	OSC$_ONLINE The OSC agent will consider the resource as offline.
FAULTED	OSC$_FAULTED The OSC agent will consider the resource as

	faulted and OSC triggers resource fault handling (i.e. failover handling is triggered for all service groups the resource is a member of).
ERROR	SS$_ACCVIO (OpenVMS error code) This return status signals to the OSC agent, that the action routine has failed to determine the state of the resource and the OSC agent will trigger the action routine error handling.
WAIT	Test monitor script starts looping. The test monitor script will not terminate within the timeout period configured for the resource which in turn causes the OSC agent to trigger the action routine timeout handling.
None of these values	SS$_ACCVIO (OpenVMS error code) This return status signals to the OSC agent, that the action routine has failed to determine the state of the resource and the OSC agent will trigger the action routine error handling.

Since the return code provided to the OSC agent just depends on the value assigned to system-wide logical that represents a resource and all OSC decisions and course of action OSC performs just rely on the return codes provided by action routines called by the OSC agents it is very easy to simulate particular failure scenarios and to test how OSC behaves according to the control attributes (see 5 Controlling OSC behavior) defined in the OSC configuration database in use in response to such failure scenarios.

Thus, if you want to test the behavior of OSC according to the control attributes defined when a particular resource fails just assign the value "FAULTED" to the system-wide logical that represents this resource. For example if you want to test how OSC behaves when the Oracle database resource ORA::ORADWH fails just re-define the logical ORA::ORADWH:

- $DEFINE/SYSTEM ORA::ORADWH "FAULTED"

If you want to test the OSC behavior when a monitor action routines does not complete within the timeout period defined (either because it lasts that long to determine the state of a resource or the monitor action routine loops) when it is called for a particular resource, assign the value

"WAIT" to the system-wide logical that represents the resource. For example:

- $DEFINE/SYSTEM ORA::ORADWH "WAIT"

If you want to test the behavior of OSC when a monitor action routine fails to determine the state of a particular resource (monitor action routine is buggy) assign the value "ERROR" to the system-wide logical that represents the resource: For example:

- $DEFINE/SYSTEM ORA::ORADWH "ERROR"

Script: OSC$COMMON:[CFG.TEST]TEST_MONITOR.COM

```
$! OSC Test monitor script
$!
$! Any OSC agent passes the virtual resource name in P1
$! to this routine. This routine checks if a system-wide logical
$! as passed in P1 exists. Depending on the avalue assigned to
$! this logical exit codes are returned as defined in the table
$! below.
$!
$!     Logical Value       Return Code
$!     ------------------------------------------------------------------
$!     does not exists     OSC$_OFFLINE
$!     OFFLINE             OSC$_OFFLINE
$!     OFFLINE_WAIT        OSC$_OFFLINE
$!     ONLINE              OSC$_ONLINE
$!     ONLINE_WAIT         OSC$_ONLINE
$!     FAULTED             OSC$_FAULTED
$!     ERROR               OSC$_ERROR (script error = SS$_ACCVIO)
$!     WAIT                script loops
$!
$! ------------------------------------------------------------------
$!
$! Return Codes
$!
$ OSC$_ONLINE  = 1
$ OSC$_OFFLINE = 9
$ OSC$_FAULTED = 19
$ OSC$_ERROR   = 12
$!
$!
$! V1.0        W.Burger / 19.6.2008
$!     initial Version
$!
$!
================================================================
====
$!
$ Logical = f$trnlnm (P1)
$!
$ if (Logical .eqs. "")
$ then
$       define/system 'P1 "OFFLINE"
$       exit (OSC$_OFFLINE)
$ endif
$!
$ if (Logical .eqs. "OFFLINE") then exit (OSC$_OFFLINE)
$ if (Logical .eqs. "ONLINE") then exit (OSC$_ONLINE)
```

```
$ if (Logical .eqs. "OFFLINE_WAIT") then exit (OSC$_OFFLINE)
$ if (Logical .eqs. "ONLINE_WAIT") then exit (OSC$_ONLINE)
$ if (Logical .eqs. "FAULTED") then exit (OSC$_FAULTED)
$ if (Logical .eqs. "ERROR") then exit (OSC$_ERROR)
$!
$ if (Logical .eqs. "WAIT")
$ then
$       loop:
$               wait 00:00:02
$               goto loop
$ endif
$!
$ exit (OSC$_ERROR)
```

11.3 OSC test online script

As described in section 4.3.1.2 OSC Agent Operations and 9.3.4 Online the online action routine is called by an OSC agent to bring an OSC resource online (start the resource). When the OSC test online action routine is called for a particular OSC resource it assigns the value "ONLINE" to the system-wide logical that represents the resource except when the value of the system-wide logical is:

- FAULTED
- ERROR
- OFFLINE_WAIT
- WAIT

If the system-wide logical has one of these values listed above assigned, the OSC test online action routine does not modify the value of the logical.

Thus, if you want to test how OSC behaves according to the control attributes defined in the OSC configuration database in use when an online action routine fails to bring a particular OSC resource online just change the value of the logical that represents the OSC resource from "OFFLINE" to "OFFLINE_WAIT" (keep in mind that only offline resources will be started). For example if you want to test what happens if the OSC oracle agent fails to bring the Oracle 10 database ORADWH represented by the OSC resource ORA::ORADWH online (fails to start the database) perform the following actions:

1. Redefine the value of the system-wide logical ORA::ORADWH manually (see section 11.2 OSC test monitor script for a detailed description of the "OFFLINE_WAIT" value assigned to a resource logical):

```
$DEFINE/SYSTEM ORA::ORADWH "OFFLINE_WAIT"
```

2. Try to start the service group the resource ORA::ORADWH is member of using the OSC$MGR utility ONLINE SRVGRP command or the appropriate OscMgrGUI (graphical management user interface for OSC) command.

Script: OSC$COMMON:[CFG.TEST]TEST_ONLINE.COM

```
$! OSC Test online script
$!
$! Any OSC agent passes the virtual resource name in P1
$! to this routine. This routine 'onlines' the resource depending
$! on the actual state of the logical in P1
$!
$!     Logical Value        Action
$!     ----------------------------------------------------
$!     FAULTED              No Action
$!     ERROR                No Action
$!     OFFLINE_WAIT         No Action
$!     WAIT                 routine loops
$!     any other state      state change to "ONLINE"
$!
$! ------------------------------------------------------------
$!
$! Return Codes
$!
$ OSC$_ONLINE  = 1
$ OSC$_OFFLINE = 9
$ OSC$_FAULTED = 19
$ OSC$_ERROR   = 12
$!
$ OSC$_RETURN = 5
$!
$! V1.0        W.Burger / 19.6.2008
$!     initial Version
$!
$!
==========================================================
====
$!
$ Logical = f$trnlnm (P1)
$!
$ if (Logical .eqs. "FAULTED") then exit (OSC$_RETURN)
$ if (Logical .eqs. "ERROR") then exit (OSC$_RETURN)
$ if (Logical .eqs. "OFFLINE_WAIT") then exit (OSC$_RETURN)
$!
```

```
$ if (Logical .eqs. "WAIT")
$ then
$       loop:
$               wait 00:00:02
$               goto loop
$ endif
$!
$ define/system 'P1 "ONLINE"
$ exit (OSC$_RETURN)
```

11.4 OSC test offline script

As described in section 4.3.1.2 OSC Agent Operations and 9.3.5 Offline the offline action routine is called by an OSC agent to shutdown an OSC resource. When the OSC test offline action routine is called for a particular OSC resource it assigns the value "OFFLINE" to the system-wide logical that represents the resource except when the value of the system-wide logical is:

- FAULTED
- ERROR
- ONLINE_WAIT
- WAIT

If the system-wide logical has one of these values listed above assigned, the OSC test offline action routine does not modify the value of the logical.

Thus, if you want to test how OSC behaves according to the control attributes defined in the OSC configuration database in use when an offline action routine fails to shutdown a particular OSC resource just change the value of the logical that represents the OSC resource from "ONLINE" to "ONLINE_WAIT" (keep in mind that only online resources can be shutdown). For example if you want to test what happens if the OSC oracle agent fails to shutdown the Oracle 10 database ORADWH represented by the OSC resource ORA::ORADWH perform the following actions:

1. Redefine the value of the system-wide logical ORA::ORADWH manually (see section 11.2 OSC test monitor script for a detailed description of the "ONLINE_WAIT" value assigned to a resource logical):
   $DEFINE/SYSTEM ORA::ORADWH "ONLINE_WAIT"
2. Try to shutdown (offline) the service group the OSC resource ORA::ORADWH is member of using the OSC$MGR utility OFFLINE

SRVGRP command or the appropriate OscMgrGUI (graphical management user interface for OSC) command.

Script: OSC$COMMON:[CFG.TEST]TEST_OFFLINE.COM

```
$! OSC Test offline routine
$!
$! Any OSC action routine passes the virtual resource name in P1
$! to this routine. This routine 'offlines' the resource depending
$! on the actual state of the logical in P1
$!
$!     Logical Value        Action
$!     ------------------------------------------------
$!     FAULTED              No Action
$!     ERROR                No Action
$!     ONLINE_WAIT          No Action
$!     WAIT                 routine loops
$!     any other state      state change to "OFFLINE"
$!
$! ---------------------------------------------------------------
$!
$! Return Codes
$!
$ OSC$_ONLINE  = 1
$ OSC$_OFFLINE = 9
$ OSC$_FAULTED = 19
$ OSC$_ERROR   = 12
$!
$ OSC$_RETURN = 5
$!
$! V1.0        W.Burger / 19.6.2008
$!     initial Version
$!
$!
=====================================================
====
$!
$ Logical = f$trnlnm (P1)
$!
$ if (Logical .eqs. "FAULTED") then exit (OSC$_RETURN)
$ if (Logical .eqs. "ERROR") then exit (OSC$_RETURN)
$ if (Logical .eqs. "ONLINE_WAIT") then exit (OSC$_RETURN)
$!
$ if (Logical .eqs. "WAIT")
$ then
$       loop:
$               wait 00:00:02
$               goto loop
$ endif
$!
$ define/system 'P1 "OFFLINE"
```

```
$ exit (OSC$_RETURN)
```

11.5 OSC test clean script

As described in section 4.3.1.2 OSC Agent Operations and 9.3.6 Clean the clean action routine is called (forced shutdown) for an OSC resource after a resource has failed to come online, failed to go offline, or failed while in an online state. When the OSC test clean action routine is called for a particular OSC resource it assigns the value "OFFLINE" to the system-wide logical that represents the resource except when the value of the system-wide logical is:

- ERROR
  In this case the OSC test clean routine return SS$_ACCVIO (error code value 12) to signal that the clean action routine has failed to forcibly shutdown the resource.
- WAIT
  The OSC clean action scripts start looping.

Script: OSC$COMMON:[CFG.TEST]TEST_CLEAN.COM

```
$! OSC Test clean script
$!
$! Any OSC agent passes the virtual resource name in P1
$! to this routine. This routine 'cleans' the resource depending
$! on the actual state of the logical in P1
$!
$!     Logical Value        return code
$!     ------------------------------------------------------------
$!     ERROR                OSC$_ERROR
$!     WAIT                 routine loops
$!     any other state      OSC$_ONLINE
$!
$! ------------------------------------------------------------------
$!
$! Return Codes
$!
$ OSC$_ONLINE  = 1
$ OSC$_OFFLINE = 9
$ OSC$_FAULTED = 19
$ OSC$_ERROR   = 12
$!
$! V1.0        W.Burger / 19.6.2008
$!     initial Version
$!
```

```
$!
==========================================================
====
$!
$ Logical = f$trnlnm (P1)
$!
$ if (Logical .eqs. "ERROR") then exit (OSC$_ERROR)
$!
$ if (Logical .eqs. "WAIT")
$ then
$       loop:
$               wait 00:00:02
$               Logical = f$trnlnm (P1)
$               if (Logical .nes. "WAIT")
$               then
$                       goto exit_loop
$               endif
$               goto loop
$ endif
$!
$ exit_loop:
$ define/system 'P1 "OFFLINE"
$ exit (OSC$_ONLINE)
```

11.6 Starting OSC in simulation mode

You cannot switch from control to simulation mode or from simulation to control mode during the OSC run-time. The operational mode is defined when OSC is started cluster-wide.

The easiest way to start OSC cluster-wide in simulation mode is to execute the OSC$MGR utility command:

- OSC$MGR> STARTUP/CLUSTER/MODE=SIMULATION

Alternatively one can modify the OSC master control attribute **OscCtrlSimulate** to TRUE (see A.6 OSC master control and service engine attributes) and then execute the OSC startup script

- @SYS$STARTUP:OSC$STARTUP.COM

In order to modify the **OscCtrlSimulate** attribute one has to:

- Copy the default OSC configuration database into a working OSC configuration database
- Modify the **OscCtrlSimulate** attribute in the new working OSC configuration database
- Finally activate this working OSC configuration database as the new default OSC configuration database.

You can also use the graphical management user interface OscMgrGUI to start VSI OpenVMS ServiceControl in simulation mode.

A

Appendix

A.1 OSC cluster and system states

The OSC$MGR utility SHOW CLUSTER command provides the information about the current status of the OSC cluster.

The status information contains five columns:

1. OSC cluster member name
2. OSC cluster member state
3. Votes of the OSC cluster member
4. OSC cluster quorum
5. OSC master control process status on the OSC cluster member

A.1.1 OSC Cluster membership states

Status	Description
MEMBER	Node is a valid OSC cluster member
MBR_RQST	The active OSC master control process has received a cluster membership request from a potential OSC node. OSC cluster membership processing is in progress.
LOST_CONN	Lost SCS connection to the node
RECNX_PEND	The active OSC master control process has lost connection to a OSC member node and is waiting for a reconnect request from the OSC node (reconnection timer has not expired – see the OscCtrlReconnInterval attribute description in section A.6.1 Mandatory OSC master control attributes)

USR_LCK	The OSC node is user locked. An OSC cluster member is user locked if it has been removed from the expected active OSC node member set of the OSC cluster on user request (see 7.2.8 How to lock an OSC node). An OSC node can be removed from the expected active OSC node member set of the OSC cluster only if it has been previously shutdown (OSC cluster member state: SHUTDWN) or if it has previously left the OSC cluster unexpectedly (state: BROKEN). Once an OSC cluster member is user locked the OSC environment prevents OSC from starting up on that OSC node. The OSC node has to be unlocked before OSC can be started on that node again.
BROKEN	The OSC node has been removed from the OSC cluster after the active OSC master control process has lost connection to the OSC node and the OSC node has not re-joined the OSC cluster within the reconnection interval (typically the node has been shutdown or crashed).
SHUTDWN	OSC has been manually shutdown on the node with SHUTDOWN/NODE command of the OSC$MGR utility.

A.1.2 OSC master control states

Status	Description
ACTIVE	OSC master control process is active on this OSC node
STANDBY	OSC master control process is standby on this OSC node.
STS_TRANS	OSC cluster state transition. The OSC state transition phase is initiated whenever: • OSC cluster is started • An OSC cluster member joins the OSC cluster • An OSC cluster member is removed from the OSC cluster • The active OSC master control process moves from one OSC cluster member to another During the state transition phase the OSC master control process requests the current state of all managed resources, services and service groups from all OSC agents and OSC service engines on the available OSC cluster

	members. This is done to ensure that the status information of the managed items maintained by the active OSC master control process is up to date before it starts service group processing. This mechanism avoids wrong service group offline/failover decisions based on incomplete and/or out of date status information.
CLU_BLK	OSC cluster quorum lost The OSC master control process is in blocking mode. The OSC master control process stops performing automatic service group activities (taking service group offline, failing over service groups) in response to resource fault conditions. In addition it blocks any interactive service group, service and resource management commands except the SHOW commands.
RECNX_PEND	The active OSC master control process lost connection to a OSC node and is waiting for a reconnect request from the standby OSC master control process on this OSC node (reconnection timer has not been expired – see also A.1.1 OSC Cluster membership state).
USR_LCK	The OSC node is user locked. An OSC cluster member is user locked if it has been removed from the expected active OSC node member set of the OSC cluster on user request (see 7.2.8 How to lock an OSC node). An OSC node can be removed from the expected active OSC node member set of the OSC cluster only if it has been previously shutdown (OSC cluster member state: SHUTDWN) or if it has previously left the OSC cluster unexpectedly (state: BROKEN). Once an OSC cluster member is user locked the OSC environment prevents OSC from starting up on that OSC node. The OSC node has to be unlocked before OSC can be started on that node again.
BROKEN	The OSC node has been removed from the OSC cluster after the active OSC master control process lost connection to the OSC node and the OSC node did not re-join the OSC cluster within the reconnection interval (typically the node has been shutdown or crashed).
SHUTDWN	OSC has been manually shutdown on the node with SHUTDOWN/NODE command of the OSC$MGR utility.

A.2 OSC service group states

A.2.1 OSC service group status keywords

This section lists and describes the status keywords used to display the status for service groups. The actual status of a service group can be a combination of the keywords listed below.

Status	Description
ONLINE	Service group is online
OFFLINE	Service group is offline
FAULTED	Service group has faulted.
ADMIN_WAIT	Service group is in ADMIN_WAIT state waiting for system management intervention.
UNKNOWN	Service group state is UNKNOWN. The state of at least one of its resources is UNKNOWN. A resource is set to UNKNOWN when the managing OSC agent is (re) started. It remains UNKNOWN until the initial monitor action routine call for a resource returns or times out.
FROZEN	Service group is frozen. This status bit indicates that at least one of its resources is frozen.
DISABLED	Service group is disabled. This status bit indicates that at least one of its resources is disabled.
PARTIAL	This indicates that the status of some services and resources of the service group differ from the status displayed for the service group. For example, the service group state ONLINE | PARTIAL indicates that at least one resource is not online. The service group state UNKNOWN | PARTIAL indicates that at least one resource is in known state. This bit is typically set during service group ONLINE processing or when a non-critical resource has failed.
RESOURCE EXECPTION	This status bit indicates that a service group status change was caused by a resource exception. Check the resources with the SHOW RESOURCE command of the OSC$MGR utility to determine the resources that caused the status change.

PART. UNKNOWN	Only set in the service group summary status field It indicates that a service group on one of its execution nodes is at least in an UNKNOWN state.
PART. FAULT	Only set in the service group summary status field It indicates that a service group has faulted on one of its execution nodes.
PART. ADMIN_WAIT	Only set in the service group summary status field It indicates that a service group is in ADMIN_WAIT state waiting for system management intervention on one of its execution nodes.
PART. FROZEN	Only set in the service group summary status field It indicates that a service group has been frozen on one of its execution nodes.
PART. DISABLED	Only set in the service group summary status field It indicates that a service group has been disabled on one of its execution nodes.
TOO MANY INSTANCES	Only set in the service group summary status field Too many instances of the service group are online within the OSC cluster.
TOO FEW INSTANCES	Only set in the service group summary status field Too few instances of the service group are online within the OSC cluster.

A.2.2 OSC service group transaction

This section lists and describes the keywords used to display the transaction state when a service group transaction is in progress.

Transaction	Description
GOING ONLINE	Service group online transaction is in progress.
GOING OFFLINE	Service group offline transaction is in progress.
CLEAR FAULT	Transaction to clear the service group FAULT state is in progress.

CLEAR ADMIN_WAIT	Transaction to clear the service group ADMIN_WAIT state is in progress.
FREEZE	Service group freeze transaction is in progress.
UNFREEZE	Service group unfreeze transaction is in progress.
ENABLE	Service group enable transaction is in progress.
DISABLE	Service group disable transaction is in progress.

A.3 OSC service states

A.3.1 OSC service status keywords

This section lists and describes the status keywords used to display the status for services. The actual status of a service can be a combination of the keywords listed below.

Status	Description
ONLINE	Service is online
OFFLINE	Service is offline
FAULTED	Service has faulted.
ADMIN_WAIT	Service is in ADMIN_WAIT state waiting for system management intervention.
UNKNOWN	Service state is UNKNOWN. The state of at least one of its resources is UNKNOWN. . A resource is set to UNKNOWN when the managing OSC agent is (re) started. It remains UNKNOWN until the initial monitor action routine call for a resource returns or times out.
FROZEN	Service is frozen. This status bit indicates that at least one of its resources is frozen.
DISABLED	Service is disabled. This status bit indicates that at least one of its resources is disabled.
PARTIAL	It indicates that the status of some resources of the service differ from the status displayed for the service. For example, the service state ONLINE | PARTIAL indicates that at least one resource is not online. The service state UNKNOWN | PARTIAL indicates that at least one resource is in a known state. This bit is typically set during service ONLINE processing or when a non-critical resource has failed.
RESOURCE EXECPTION	This status bit indicates that a service status change was caused by a resource exception. Check the resources with the SHOW RESOURCE command of the OSC$MGR utility to determine the resources that caused the status change.

A.3.2 OSC service transactions

This section lists and describes the keywords used to display the transaction state when a service transaction is in progress.

Transaction	Description
GOING ONLINE	Service online transaction is in progress.
GOING OFFLINE	Service offline transaction is in progress.
CLEAR FAULT	Transaction to clear the service FAULT state is in progress.
CLEAR ADMIN_WAIT	Transaction to clear the service ADMIN_WAIT state is in progress.
FREEZE	Service freeze transaction is in progress.
UNFREEZE	Service unfreeze transaction is in progress.
ENABLE	Service enable transaction is in progress.
DISABLE	Service disable transaction is in progress.

A.4 OSC resource states

A.4.1 OSC resource status keywords

This section lists and describes the status keywords used to display the status for resources. The actual status of a resource can be a combination of the keywords listed below.

Status	Description
ONLINE	Resource is online
OFFLINE	Resource is offline
LOCKED	Resource is a cluster locked resource online on any of the OSC cluster members.
FAULTED	Resource has faulted.
ADMIN_WAIT	Resource is in ADMIN_WAIT state waiting for system management intervention.
UNKNOWN	Resource state is UNKNOWN. A resource is set to UNKNOWN when the managing OSC agent is (re) started. It remains UNKNOWN until the initial monitor action routine call for a resource returns or times out.
FROZEN	Resource is frozen.
DISABLED	Resource is disabled.
UNEXP. OFFLINE	Reason for resource FAULT state An online resource state has unexpectedly changed its state to offline without being requested to go offline.
UNEXP. ONLINE	Resource has been started outside OSC control.
MONITOR HUNG	The monitor action routine has not completed within the expected time. If the resource is an **On-Off** resource the clean action routine will be called.
ONLINE INEFF	Resource online transaction failed.
ONLINE HUNG	The online action routine has not completed within the expected time. If the resource is an **On-Off** resource the clean action routine will be called.
OFFLINE INEFF	Resource offline transaction failed.

OFFLINE HUNG	The offline action routine has not completed within the expected time. If the resource is an **On-Off** resource the clean action routine will be called.
UNABLE OFFLINE	The clean action routine failed to forcibly shutdown a resource.
SCRIPT ERROR	Reason for resource ADMIN_WAIT state. The monitor or the clean action routine retuned an OpenVMS error code indicating that the action/clean routine failed due to a run-time error.
SCRIPT TMO	Reason for resource ADMIN_WAIT state. Consecutive action routine calls (>**TimeOutRetryLimit**) have timed out. The action routine seems to loop due to a code bug.
SCRIPT MWAIT	Reason for resource ADMIN_WAIT state. The sub process execution of a particular DCL action script timed out and the sub process cannot be killed due to MWAIT state.

A.4.2 OSC resource transactions

This section lists and describes the keywords used to display the transaction state when a service transaction is in progress.

Transaction	Description
GOING ONLINE	Resource online transaction is in progress.
GOING OFFLINE	Resource offline transaction is in progress.
CLEAN UP	Resource clean up transaction (forced shutdown) is in progress.
RESTART	Restart transaction in response to a resource fault condition is in progress.
CLEAR FAULT	Transaction to clear the resource FAULT state is in progress.
CLEAR ADMIN_WAIT	Transaction to clear the resource ADMIN_WAIT state is in progress.

FREEZE	Resource freeze transaction is in progress.
UNFREEZE	Resource unfreeze transaction is in progress.
ENABLE	Resource enable transaction is in progress.
DISABLE	Resource disable transaction is in progress.

A.5 OSC agent attributes

A.5.1 Mandatory OSC agent attributes

Attribute Name	Data Type	Description
AgentType	String[16]	Resource type managed by the OSC agent. This attribute has to be defined when the OSC agent is added to the OSC configuration database (see 9.7 Adding an OSC agent to the configuration database). This attribute cannot be modified.
AgentName	String[32]	OpenVMS process name of the OSC agent when it is started by the OSC Service engine. Use quotation marks for case sensitive input.
AgentDescription	String[64]	Agent description - use quotation marks for case sensitive input.
AgentCategory	String[16]	The resource category the OSC agent can manage. When the OSC agent starts the **AgentCategory** attribute is passed on to the **ResourceCategory** attribute for all resources the agent manages if not otherwise defined at the resource level. The **ResourceCategory** attribute overrules the **AgentCategory** attribute. Thus, the user can re-define the resource category (**On-Off**, **On-Only**, **Persistent**) of a resource without modifying the managing OSC agent. For detailed information about resource categories please refer to the section 4.2.1.3 Resource)
AgentImage	String[256]	This attribute defines the image to activate when the OSC agent process is started by the OSC service engine. If a compiled OSC agent is added enter the directory and image name of the compiled OSC agent. If you add a DCL script based OSC agent you have to enter one of the bundled OSC agent images (provided by the OSC installation procedure) located in OSC$BIN that provide the OSC agent framework functionality. VSI recommends the use of the OSC$BIN:OSCAGT$GENERIC.EXE image if no specific functionality of any other bundled OSC agent image is required.
AgentUIC	String[32]	The OSC agent will be started under the UIC of the user

		defined by this attribute.
AgentMaxAction	Integer	This attribute defines how many resources can be handled by the OSC agent in parallel. If more resources are configured than the number defined by this attribute, resource processing is partially serialized. If the value of the attribute is 1 resource processing is completely serialized. **Note** Do not enter any value greater than 240

A.5.2 Optional OSC agent attributes

Attribute Name	Data Type	Description
AgentPriority	Integer	Base priority of the OSC agent process. The default value is 8.
AgentHbtInterval	Integer	The OSC agent sends heartbeat messages to the OSC service engine to signal proper operation. This attribute defines the heartbeat send interval in seconds The default value is 5 seconds.
AgentHbtTimeout	Integer	If the OSC service engine does not receive any heartbeat message from an OSC agent within the time interval in seconds defined by this attribute the OSC agent is considered as faulted. The OSC service engine tries to restart the OSC agent. The default value is 30 seconds
AgentStartTimeout	Integer	OSC agents are automatically started by the OSC service engine. This attribute defines the maximum time in seconds the OSC service engine waits for the first heartbeat message of the OSC agent after startup. If the OSC agent does not respond with this time the OSC service engine tries to restart the OSC agent. The default value is 60 seconds.
AgentShutdown	String[256]	Shutdown script to be executed when the OSC agent is requested to shutdown.

AgentShutdownTimeout	Integer	This attribute defines the maximum time in seconds the OSC service engine waits for the OSC agent process termination message when it requests the agent to shutdown. If the OSC agent does not terminate within this time period the OSC service engine unconditionally kills the OSC agent process. The default value is 60 seconds.
AgentAutoRestartLimit	Integer	When the OSC service engine restarts an OSC agent an internal restart counter is incremented. After the OSC agent has been restarted it has to run properly for the time defined by the **AgentConfLimit** before the restart counter is cleared. This attribute defines the maximum number of OSC agent restart attempts if the OSC agent continuously fails within the **AgentConfLimit** time interval. If the restart attempts exceeds the value defined by this attribute the states of all resources managed by the OSC agent are set to ADMIN_WAIT and a fatal ADMIN_WAIT state change event message is triggered for immediate system management intervention. This mechanism prevents OSC from managing applications using unstable OSC agents. The default value is 5.
AgentConfLimit	Integer	This attribute defines the number of **AgentHbtTimout** intervals the OSC service engine has to continuously receive proper heartbeat messages from an OSC agent before the OSC agent's restart counter is reset. The default value is 10. The time interval in seconds = **AgentConfLimt** * **AgentHbtTimeout**.
AgentInformational	Boolean	This attribute defines whether or not the OSC agent logs informational messages in the OSC event file. The default is FALSE.
AgentDebug	Boolean	This attribute defines whether or not the OSC agent logs debug messages in the OSC agents log file. The default is FALSE.

A.6 OSC master control and service engine attributes

A.6.1 Mandatory OSC master control attributes

Attribute Name	Data Type	Description
OscCtrlClusterName	String[32]	Name of the OSC cluster. Use quotation marks for case sensitive input.
OscCtrlNode	String[256]	This attribute defines the members of an OSC cluster. The OSC cluster node list is a comma separated list of node configuration blocks. A node configuration block contains the SCS node name of an OSC cluster member and the number of votes of an OSC cluster member separated by a colon. Example: VMSTM1:3,VMSTM2:2,VMSTM3:1 In this example the OSC cluster consists of the nodes • VMSTM1 having 3 votes • VMSTM2 having 2 votes • VMSTM3 having 1 vote If no votes are assigned to an OSC cluster member vote count 1 is assumed. Example: VMSTM1,VMSTM2:2,VMSTM3 Since VMSTM1 and VMSTM3 have no votes explicitly assigned the OSC automatically assigns 1 vote to each node.
OscCtrlFailoverPolicy	String[32]	Two different OSC failover policies can be configured: • Static failover • Load-balanced failover The OSC cluster-wide failover policy is defined by the keyword assigned to this attribute: • STATIC • LOAD-BALANCING For detailed information about OSC failover policies please refer to section 5.2 Controlling OSC Failover Policy.
OscCtrlNodeLoadCap	String[256]	If the OSC failover policy is set to LOAD-BALANCING, OSC uses a load-balancing mechanism that

		determines which system hosts an application (service group) during startup, or after an application or server fault. This load-balancing algorithm requires that the user defines: • The maximum workload capability of each OSC cluster member (= maximum workload a particular OSC cluster member is able to handle = user defined number). • Workload values for each service group configured. The maximum workload capability of all OSC cluster member is defined by this attribute. This attribute contains a comma separated list of OSC workload capability configuration blocks. An OSC workload capability configuration block contains the SCS node name of an OSC cluster member and the maximum system workload the node is able to handle. Example: VMSTM1:200,VMSTM2:200,VMSTM3:100 The OSC workload capability list in this example defines that the OSC cluster member VMSTM1 and VMSTM2 can run service groups with a maximum workload (= sum of the workload values assigned to the service groups) of 200. VMSTM3 can only run service groups up to a maximum workload of 100 since VMSTM3 has less power than VMSTM1 and VMSTM2. This attribute is mandatory if the OSC failover policy is set to LOAD-BALANCING. If the OSC failover policy is set to STATIC this attribute is hidden.
OscCtrlProcPriority	Integer	Base priority of the OSC master control process OSC$CTRL. The default is 10.
OscCtrlReconnInterval	Integer	When the OSC master control process detects that an OSC cluster member has left the OSC cluster (crash, SCS link lost) it waits for the OSC reconnection interval whether or not the node re-joins the cluster. After the OSC reconnection interval has expired and the node has not re-joined the OSC cluster the node is removed from the OSC cluster and the OSC master control process initiates service group failover processing. This mechanism is similar to the OpenVMS cluster reconnection interval. The default value of the OSC reconnection interval

		defined by this attribute is 30 seconds.
OscCtrlStartupWait	Integer	This attribute defines the duration of a state transition. The OSC state transition phase is initiated whenever: • An OSC cluster is started • An OSC cluster member joins the OSC cluster • An OSC cluster member is removed from the OSC cluster • The active OSC master control process moves from one OSC cluster member to another During the state transition phase the OSC master control process requests the current state of all managed resources, services and service groups from all OSC agents and OSC service engines on the available OSC cluster members. This is done to ensure that the status information of the managed items maintained by the active OSC master control process is up to date before it starts service group processing. This mechanism avoids wrong service group offline/failover decisions based on incomplete and/or out of date status information. This attribute defines the duration of the OSC state transition in seconds. The default value is 60 seconds.
OscCtrlExpVotes	Integer	Initial OSC quorum is calculated based on the value of this attribute. The OSC master control process blocks any service group activities and interactive management commands (except the SHOW commands) until the enough OSC cluster members have joined the OSC cluster to gain quorum. The default value is 1.
OscCtrlAutoAdjustQuorum	Boolean	If this attribute is set to TRUE OSC automatically adjusts quorum whenever an OSC cluster member unexpectedly leaves the OpenVMS cluster (crash, SCS link lost). VSI recommends that this feature is enabled only if the OpenVMS cluster OSC is installed on contains a quorum disk instead of a quorum system. A quorum disk cannot be configured in OSC. Thus, in order to avoid that OSC is blocked, although the OpenVMS cluster is still valid, automatic quorum adjustment

		has to be enabled.
OscCtrlSimulate	Boolean	This attribute defines whether OSC is started in full control mode (value FALSE) or in simulation mode (value TRUE) when OSC is started using the startup script SYS$STARTUP:OSC$STARTUP.COM. Starting the OSC environment in simulation mode is usefull for: • Training of how to manage OSC • Testing a new OSC configuration • Learning how OSC behaves on particular fault scenarios without affecting the applications (service groups) already running on the OpenVMS cluster. The value of this attribute is ignored if the OSC environment is started using the OSC$MGR STARTUP/CLUSTER command. When the OSC environment is started using the START/CLUSTER command the operational mode of OSC is defined by the keyword applied to the /MODE qualifier of the command (see 7.1 How to start OSC)

A.6.2 Read-Only OSC master control attributes

Attribute Name	Data Type	Description
OscCtrlNodeLocked	String[256]	This read-only attribute contains the OSC cluster members that have been removed from the active OSC cluster node set (user locked OSC nodes). This attribute is updated whenever the user removes an OSC cluster member from or adds an OSC cluster member to the active OSC cluster node set with the LOCK NODE and UNLOCK NODE command. For detailed information about adding and removing OSC cluster members during run-time please refer to the section 7.2.8 How to lock an OSC node and 7.2.9 How to unlock an OSC node.

A.6.3 Optional OSC service engine attributes

Attribute Name	Data Type	Description
OscSrvProcPrio	Integer	Base priority of the OSC service engine process OSC$SRV. The default is 8.
OscSrvHbtInterval	Integer	The OSC service engine sends heartbeat messages to the OSC master control process to signal proper operation. This attribute defines the heartbeat send interval in seconds. The default value is 5 seconds.
OscSrvHbtInterval	Integer	The OSC service engine sends heartbeat messages to the OSC master control process to signal proper operation. This attribute defines the heartbeat send interval in seconds. The default value is 5 seconds.
OscSrvHbtTimeout	Integer	If the OSC master control process does not receive any heartbeat message from an OSC service engine within the time interval in seconds defined by this attribute the OSC service engine is considered faulted. The OSC master control process tries to restart the OSC agent. The default value is 30 seconds This mechanism does not apply if an OSC service engine terminates. The OSC master control process immediately detects unexpected OSC service engine termination and consequently tries to restart OSC$SRV.
OscSrvRestartLimit	Integer	When the OSC master control process restarts an OSC service engine an internal restart counter is incremented. After the OSC service engine has been restarted it has to run properly for the time defined by the attribute OscSrvConfidentialLimit before the restart counter is cleared. Thus, this attribute defines the maximum number of OSC service engine restart attempts if the OSC service engine continuously fails within OscSrvConfidentialLimit time interval. If the restart attempts exceeds the value defined by this attribute the states of all service groups, services and resources managed by the OSC service engine are set to ADMIN_WAIT and a fatal ADMIN_WAIT state change event message is triggered for immediate system management intervention. This mechanism prevents OSC from managing applications using unstable OSC service engines. The default value is 5.
OscSrvConfidentialLimit	Integer	This attribute defines the number of **OscSrvHbtTimout** intervals the OSC master control has to continuously

	receive proper heartbeat messages from an OSC service engine before the OSC service engine's restart counter is reset. The default value is 10. The time interval in seconds = **OscSrvConfidentialLimit** * **OscSrvHbtTimeout**.

A.7 OSC resource attributes

OSC resource attributes consist of common and resource type specific attributes. This section describes the common OSC resource attributes. For detailed information about resource type specific resource attributes please refer to section 9 Developing new OSC agents.

A.7.1 Mandatory common OSC resource attributes

Attribute Name	Data Type	Description
ResourceDescription	String[64]	Resource description -use quotation marks for case sensitive input.
ServiceMember	String[1024]	This attribute defines the OSC service membership of the resource. Enter the OSC services as a comma separated list. The content of this attribute is automatically updated when the user (re) defines the OSC service membership of a resource and all its child resources using the DEFINE RESOURCE /MEMBERSHIP command of the OSC$CFG utility. For detailed information about the DEFINE RESOURCE command please refer to the online help of the OSC$CFG utility.
ResourceDependency	String[1024]	This attribute defines the child resources of an OSC resource (=resource the resource directly depends on). Enter the child resources as a comma separated list. **Note** Child resources have to be defined here node independent (that is without the @*node-name* OSC resource name extension) The content of this attribute is automatically updated if the user (re) defines the child resources of a resource using the DEFINE RESOURCE/CHILD command of the OSC$CFG utility- For detailed information about the DEFINE RESOURCE command please refer to the OSC$CFG utility online help.
ArgList	Integer	This common resource attribute specifies the list of resource attributes (common or specific) whose values are passed to the monitor, online, offline and clean

	action routines of the OSC agent that manages the resource. If the optional common resource attribute PassFuncCode is TRUE, up to 5 resource attributes can be defined herein. Otherwise 6 resource attributes can be defined.

A.7.2 Optional common OSC resource attributes

Attribute Name	Data Type	Description
ResourceCategory	String[16]	This attribute defines the resource category. Valid resource categories are: • **On-Off** • **On-Only** • **Persistent** If this attribute contains no or an invalid value the resource category is automatically inherited from the common OSC agent attribute **AgentCategory**. The option to define the resource category at the resource level enables the user to define how an OSC agent handles resource faults of the resource without having to modify the OSC agent. The attribute is valid if it contains one of the keywords: • **On-Off** • **On-Only** • **Persistent** In addition the OSC agent has to be capable of handling the resource category defined. An OSC agent designed to handle **On-Off** resources can also handle: • **On-Only** resources • **Persistent** resources An OSC agent designed to handle **On-Only** resources can also handle: • **Persistent** resources An OSC agent designed to handle **Persistent** resources cannot handle other resource categories. For more information about resource categories and

		resource fault management please refer to the sections 4 Basic Concepts and Terminology and 5 Controlling OSC behavior. No default exists.
Critical	Boolean	This attribute defines whether the resource is critical or not. OSC does not propagate resource faults of non-critical resources to the service or service-group level. OSC does not initiate failover processing when a non-critical resource fails. For more information please refer to the section 5.1.3 Critical and Non-Critical Resources. The default value is TRUE.
Enabled	Boolean	This attribute defines whether or not the resource is enabled to be managed by the OSC agent. For detailed information about the effect of disabling a resource please refer to the section 5.5.3 Disabling Resources. The default value is TRUE.
ClusterLocked	Boolean	This attribute defines whether or not a resource is a cluster-locked resource. A cluster locked resource is a resource that will be started only on one OSC cluster node regardless if the service group that owns this resource is configured to run concurrently on different nodes (**MultiInstance** or **Parallel** service group). For detailed information about cluster locked resources please refer to section 5.5 Controlling OSC Behavior at the Resource Level. Only **On-Off** resources can be configured as cluster locked resources. This attribute is ignored for **On-Only** or **Persistant** resources. The default value is FALSE
CluLckResDisAllow	String[1024]	The user can define a list of cluster locked resources that are not allowed to run on the same node. A cluster locked resource is not allowed to run on a particular node if any of the resources defined by this attribute is already online or starting up on that OSC node. The resource exclude list has to be entered as a comma separated list. This attribute applies to cluster locked resources only. If the resource is not configured as a cluster locked resource (see the description of the **ClusterLocked** attribute) this

		attribute is ignored. The resources listed in this attribute must be cluster locked resources too. OSC will not startup if this attribute of a cluster locked resource contains non-cluster locked resources. The cluster locked resource exclude list only applies to automatic failover and service group startup requests. It is ignored if a user requests to stop a cluster locked on one node and to start it on another node. In this case the cluster locked resource will start even if a resource listed in the cluster locked exclude list is already online on this particular node. In addition OSC does not relocate mutually exclusive cluster locked resources when OSC starts up and finds that mutually exclusive cluster locked resources are online on the same node (OSC implicitly assumes that this is a user defined setup).
OnlineMonitorInterval	Integer	Interval (in seconds) between two consecutive monitor calls for an ONLINE resource. The default value is 60 seconds.
OfflineMonitorInterval	Integer	Interval (in seconds) between two consecutive monitor calls for an OFFLINE resource. The default value is 300 seconds.
ToleranceLimit	Integer	This attribute defines the number of times the monitor routine has to consecutively return offline status for a resource that is considered online before OSC consideres the resource as unexpected offline. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default value is 0.
FaultOnMonitorTmo	Booleanr	This attribute defines whether or not OSC considers a monitor routine timeout as a resource fault. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default value is TRUE.
FaultOnMonitorTmoLimit	Integer	This attribute defines the number of times the monitor routine will consecutively time out before the resource is considered as faulted. For detailed information please refer to section 5.5

		Controlling OSC Behavior at the Resource Level. The default value is 4.
DisableMangeFault	Boolean	If **DisableManageFault**=TRUE fault management is disabled regardless if fault management is enabled on the service group level. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is FALSE.
OnlineRetryLimit	Integer	This attribute specifies the number of retries to online a resource if the initial attempt to bring it online fails. This attribute does not apply to **On-Only** and **Persistent** resources. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is 0.
OnlineWaitLimit	Integer	This attribute defines the number of the times the monitor routines will consecutively return offline status after an attempt to online a resource before OSC considers the online transaction as failed. This attribute does not apply to **Persistent** resources. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is 2.
OnlineTmoWaitLimit	Integer	This attribute defines the maximum number of online timeout intervals OSC waits for the completion of the online routine before OSC triggers online timeout processing. This attribute does not apply to **Persistent** resources. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is 2.
OfflineWaitLimit	Integer	This attribute defines the number of the times the monitor routines should return online status after an attempt to offline a resource before OSC considers the offline transaction as failed. This attribute does not apply to **On-Only** and **Persistent** resources.

		For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is 2.
OfflineTmoWaitLimit	Integer n	This attribute defines the maximum number of offline timeout intervals OSC waits for the completion of the offline routine before OSC triggers offline timeout processing. This attribute does not apply to **On-Only** and **Persistent** resources. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is 2.
RestartLimit	Integer	This attribute defines the number of times the OSC agent attempts to restart a failed **On-Off** resource before it reports the fault condition to the OSC service engine. This attribute does not apply to **On-Only** and **Persistent** resources. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is 0.
CleanRetryLimit	Integer	If the clean routine signals that it has failed to "clean up" a resource this attribute defines the number of clean retry attempts an OSC agent may perform. If the clean retry attempts exceed the number defined by this attribute OSC accepts that the resource cannot be "cleaned up" and the resource status is set to ADMIN_WAIT. This attribute does not apply to **On-Only** and **Persistent** resources. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is 5.
TimeOutRetryLimit	Integer	This attribute defines the number of retries to • Online • Offline • Clean an **On-Off** resource if the appropriate action routine has timed out.

		This attribute does not apply to **On-Only** and **Persistent** resources. For detailed information please refer to section 5.5 Controlling OSC Behavior at the Resource Level. The default is 5.
ConfLimit	Integer	This attribute defines how long a resource must remain online without encountering problems before previous problem counters are cleared. This attribute controls when OSC clears the **RestartCnt**, **ToleranceCnt** and **CurrentMonitorTmoCnt** values. For detailed information please refer to section 5.6 How OSC Handles Resource Faults. The default is 600.
MonitorScript	String[256]	This attribute defines the script the OSC agent executes to determine the state of the resource (monitor action routine). Apply the file name of the script and the directory path. This attribute is mandatory for all resource types (**On-Off**, **On-Only**, **Persistent**) except if the OSC agent runs a compiled OSC agent image that provides the monitor functionality implicitly. If so, do not define this attribute (leave it blank) except if you want to overrule the compiled monitor action routine. For detailed information please refer to section 9 Developing new OSC agents.
MonitorTmo	Integer	This attributes defines the time interval the OSC agent expects the monitor action routine to complete. The default is 60 seconds.
OnlineScript	String[256]	This attribute defines the script the OSC agent executes to bring the resource online (online action routine). Apply the file name of the script and the directory path. This attribute is mandatory for all **On-Off**, **On-Only** resources except if the OSC agent runs a compiled OSC agent image that provides the online functionality. If so, do not define this attribute (leave it blank) except if you want to overrule the compiled online action routine. For detailed information please refer to section 9 Developing new OSC agents.

OnlineTmo	Integer	This attributes defines the time interval the OSC agent expects the online action routine to complete. The default is 300 seconds.
OfflineScript	String[256]	This attribute defines the script the OSC agent executes to take the resource offline (offline action routine). Apply the file name of the script and the directory path. This attribute is mandatory for all **On-Off** resources except if the OSC agent runs a compiled OSC agent image that provides the offline functionality. If so, do not define this attribute (leave it blank) except if you want to overrule the compiled offline action routine. For detailed information please refer to section 9 Developing new OSC agents.
OfflineTmo	Integer	This attributes defines time interval the OSC agent expects the offline action routine to complete. The default is 300 seconds.
CleanScript	String[256]	This attribute defines the script the OSC agent executes to forcibly shutdown a resource (clean action routine). Apply the file name of the script and the directory path This attribute is mandatory for all **On-Off** resources except if the OSC agent runs a compiled OSC agent image that provides the clean functionality. If so, do not define this attribute (leave it blank) except if you want to overrule the compiled clean action routine. For detailed information please refer to section 9 Developing new OSC agents.
CleanTmo	Integer	This attributes defines time interval the OSC agent expects the clean action routine to complete. The default is 60 seconds.
OpenScript	String[256]	This attribute defines the script the OSC agent executes to initialize a resource. Apply the file name of the script and the directory path If the managed resources do not have to be initialized or if the OSC agent runs a compiled image that provides the open functionality, do not define this attribute (leave it blank) except if you want to overrule the compiled open action routine.

		For detailed information please refer to section 9 Developing new OSC agents.
OpenTmo	Integer	This attributes defines time interval the OSC agent expects the open action routine to complete. The default is 60 seconds.
PassFuncCode	Boolean	This attribute defines whether or not the action routine's function code is passed to the action routines. For detailed information please refer to section 9 Developing new OSC agents.
ScriptExecUser	String[32]	This attribute defines the user account used to execute DCL action scripts at the resource level. If this attribute is defined for a particular resource the OSC agent executes the DCL action scripts in the context of the user defined in this attribute. If the attribute value defined is invalid (user defined is not found in SYSUAF), or this attribute is empty, the user context of the OSC agent is used to run the DCL action scripts

A.8 OSC service attributes

A.8.1 Mandatory OSC service attributes

Attribute Name	Data Type	Description
ServiceDescription	String[64]	Service description - use quotation marks for case sensitive input.
ServiceGrpMember	String[32]	This attribute defines the OSC service group membership of the service. Enter the service group the service is member of. **Note** A service can be member of only one service group (see section 8.2 OSC configuration rules). The content of this attribute is automatically updated when the user (re) defines the OSC service group membership of a service and all its child services using the DEFINE SERVICE /MEMBERSHIP command of the OSC$CFG utility: For detailed information about the DEFINE SERVICE command please refer to the online help of the OSC$CFG utility.
ServiceDependency	String[1024]	This attribute defines the child services of an OSC service (=services the services directly depends on). Enter the child services as a comma separated list. The content of this attribute is automatically updated if the user (re) defines the child services using the DEFINE SERVICE/CHILD command of the OSC$CFG utility- For detailed information about the DEFINE SERVICE command please refer to the online help of the OSC$CFG utility.
ServicePriority	Integer	This attribute defines the “importance” of a service within a service group. For detailed information please refer to section 5.4 Controlling OSC Behavior at the Service Level.

A.9 OSC service group attributes

A.9.1 Mandatory OSC service group attributes

Attribute Name	Data Type	Description
SrvGrpType	String[16]	This attribute defines the service group categories. Enter one of the valid keyword listed below: • **Failover** • **MultiInstance** • **Parallel** For detailed information about service group categories please refer to the section Fig. 4.3: Service failover due to a system failure. 4.2 Understanding OSC components.
SrvGrpDescription	String[64]	Service group description – use quotation marks for case sensitive input.
SrvGrpNodes	String[1024]	This attribute defines the service group execution node list. The service group execution node list defines the OSC cluster members the service group is allowed to be started on. It contains a comma separated list of OSC node configuration blocks. An OSC node configuration block contains the SCS node name of an OSC cluster member and the node priority where to run the service group on. These two attributes are separated by a colon. Example: VMSTM1:2,VMSTM2:3,VMSTM3:1 The execution node list in the example above defines that a service group can be started on node VMSTM1, VMSTM2 and VMSTM3. VMTM2 has the highest execution priority (3) followed by node VMSTM1 (2) and VMSTM3 (1). If the OSC failover policy is set to STATIC and the service group is a **Failover** service group, OSC tries to initially start the service group on VMSTM2. VMSTM1 is the primary and VMSTM3 the secondary failover node. If the service group is configured as a **MultiInstance** service group, the service group is started on as many

		OSC nodes as defined by the service group attribute **SrvGrpMultInstance** according to the node priority of the OSC nodes. If the service group is a **Parallel** service group the service group is started on all OSC defined in the execution node list regardless of the node priority. If the OSC failover policy is set to LOAD-BALANCING the node priorities of the OSC nodes are ignored. The OSC node selection algorithm to start **Failover** and **MultiInstance** service group is based on the OSC free workload capabilities of the OSC members listed in the node execution list and the service group workload value (see section 5.1.7.2 LOAD-BALANCING Failover Policy).
SrvGrpLoad	Integer	This attribute defines the workload value of the service group. This attribute is mandatory if the selected OSC failover policy is LOAD-BALANCING (see section 5.1.7.2 LOAD-BALANCING Failover Policy).

A.9.2 Optional OSC service group attributes

Attribute Name	Data Type	Description
SrvGrpDisAllow	String[1024]	The user can define that particular service groups are not allowed to run on the same node. The service group is not allowed to run on a particular OSC node if any of the service groups defined by the **OscSrvDisAllow** attribute is already running or starting up on that OSC node (see also 5.2 Controlling OSC Failover Policy). The service group exclude list has to be entered as a comma separated list. This attribute only applies to automatic failover and service group startup requests. The service group exclude list is ignored if a user requests a service group to start on a particular OSC node (see 7.3.3 How to online a Service Group). Example: If the **SrvGrpDisAllow** attribute of the service group XMLT contains "ORADWH, MYSQL" the service group XMLT will not be started on a particular OSC node due to an automatic startup or failover request if either of the service groups ORADWH and/or MYSQL is already running on that OSC node, regardless of the OSC failover policy defined (see 5.2 Controlling

		OSC Failover Policy).
SrvGrpMultiInstance	Integer	This attribute defines how many instances of a **MultiInstance** service group have to be active on the OSC cluster. If the service group is not a **MultiInstance** service group the attribute is ignored. The default value is 1.
SrvGrpAutoStart	Boolean	This attribute defines whether or not OSC automatically starts the service group (if it is not already online) when OSC starts up. The default value is TRUE.
SrvGrpManageFaults	Boolean	This attribute specifies whether OSC calls the clean action routine when a resource of the service group fails or not. You can configure each service group to operate as desired. • If the value of the **SrvGrpManageFault** attribute is TRUE, OSC calls the clean action routine when an **On-Off** resource fails. If an **On-Only** or **Persistent** resource fails the resource is marked faulted and no further actions are performed. • If the value of the **SrvGrpManageFault** attribute is FALSE, OSC takes no action on a resource fault; it "hangs" the service group until system management has intervened. OSC marks the resource, the affected service and the service group state as ADMIN_WAIT and does not fail over the service group until the resource fault is removed and the ADMIN_WAIT state is manually cleared via the OSC$MGR utility. The default value is TRUE.
SrvGrpDisable	Boolean	This attribute defines whether or not the service group is enabled to be managed by OSC. For detailed information about the effect of disabling service groups please refer to the section 5.3.11 Disabling Service Groups. The default value is FALSE.
SrvGrpFaultProp	Boolean	This attribute defines whether a resource fault is propagated up the resource dependency tree to the service and service group level. • If the value of the **SrvGrpFaultProp** attribute

		is TRUE (default), a resource fault is propagated up the dependency tree. If the faulted resource is critical, the service group is taken offline and failed over, provided the **SrvGrpFailover** attribute is set to TRUE (default) and the service group is not a **Parallel** service group. • If the **SrvGrpFaultProp** is set to FALSE, resource faults are contained at the resource level. OSC does not take the service group offline, thus preventing failover. The default value is TRUE.
SrvGrpFailover	Boolean	This attribute defines the service group behavior in response to service group and system faults. • If the **SrvGrpFailover** attribute is TRUE, the service group fails over when a system or a service group faults, provided that a suitable OSC system exists to failover the service group, and the service group is not defined as a **Parallel** service group. • If the **SrvGrpFailover** attribute is FALSE, the service group does not failover when a system or service group fails. If a fault occurs in a service group, the service group is taken offline. If an OSC system faults, the service group is not started on another system. The default value is TRUE.
SrvGrpCleanOnUnexpOn	Boolean	This attribute specifies the behavior of OSC when OSC detects that an **On-Off** resource was started outside of OSC control: • If the value of the **SrvGrpCleanOnUnexpOn** attribute is FALSE (default), OSC marks the resource, the affected service and the service group state as ADMIN_WAIT. Thus, the service group is removed from resource fault and failover processing until system management has cleaned up the situation and the ADMIN_STATE state is manually cleared via the OSC$MGR utility. OSC triggers a fatal ADMIN_WAIT state change event message for immediate system management intervention. • If the value of the **SrvGrpCleanOnUnexpOn** attribute is TRUE, OSC unconditionally takes this **On-Off** resource offline.

		This attribute does not apply to **On-Only** or **Persistent** resources, since these resources are cluster aware and thus no concurrency violation risk exists, even if the resources were started outside OSC control. OSC just updates the status of the resources. The default value is FALSE.
SrvGrpOnlineLocked	Boolean	This controls the online behavior of **MultiInstance** and **Parallel** service groups if the service group contains cluster locked resources. A cluster locked resource is a resource that will be started only on one OSC cluster node regardless if the service group that owns this resource is configured to run concurrently on different nodes (**MultiInstance** or **Parallel** service group – see also section 5.5.1 Resource Type Attributes). If a service group starts up on any node and it detects that a cluster locked resource is already online on any other OSC node the value of this Boolean attributes decides whether or not OSC starts the parent resources of this particular cluster locked resource or not. If the value of this attribute is FALSE, OSC will not start the parent resources of the cluster locked resource, if the cluster locked resource cannot be started due to the lock state (already online on another node) of the cluster locked resource. If the value of this attribute is TRUE, the parent resources will be started. The default value is TRUE.

A.10 OSC directory structure

- OSC$COMMON
  This concealed device is the root directory of the VSI OpenVMS ServiceControl directory tree

- OSC$BIN
  This directory contains all OSC images provided by the VSI OpenVMS ServiceControl installation procedure and the common startup scripts for all OSC components (OSC master control engine, OSC service engine, bundled OSC agents).

- OSC$COMMON:[BIN.AXP.V732]
  This directory contains all OpenVMS V7.3-2 AXP specific OSC.

- OSC$COMMON:[BIN.AXP.V82]
  This directory contains all OpenVMS V8.2 AXP specific OSC.

- OSC$COMMON:[BIN.AXP.V83]
  This directory contains all OpenVMS V8.3 AXP specific OSC.

- OSC$COMMON:[BIN.IA64.V82]
  This directory contains all OpenVMS V8.2 IA64 specific OSC.

- OSC$COMMON:[BIN.IA64.V821]
  This directory contains all OpenVMS V8.2-1 IA64 specific OSC.

- OSC$COMMON:[BIN.IA64.V83]
  This directory contains all OpenVMS V8.3 AI64 specific OSC.

- OSC$COMMON:[BIN.IA64.V831]
  This directory contains all OpenVMS V8.3-1H1 AI64 specific OSC.

- OSC$CFG
  This directory contains the default and working OSC configuration databases. The DCL action scripts of the DCL script based OSC agents bundled with VSI OpenVMS are installed in subdirectories of the OSC$CFG directory:

OSC Agent	DCL action script directory
OscAgtNCLObj	OSC$COMMON:[CFG.DECNET]
OscAgtFailIP	OSC$COMMON:[CFG.FAILIP]
OscAgtMYSQL	OSC$COMMON:[CFG.MYSQL]
OscAgtORA	OSC$COMMON:[CFG.ORA]
OscAgtORALS	OSC$COMMON:[CFG.ORA]
OscAgtPERF	OSC$COMMON:[CFG.PERFDAT]
OscAgtRDB	OSC$COMMON:[CFG.RDB]
OscAgtSQLSRV	OSC$COMMON:[CFG.SQLSRV]
OscAgtSQLDIS	OSC$COMMON:[CFG.SQLSRV]
OscAgtTEST	OSC$COMMON:[CFG.TEST]

- OSC$EXAMPLES
  This directory contains sample C code and DCL script examples of how to develop a new OSC agent (see section 9 Developing new OSC agents).

- OSC$HELP
  This directory contains the documentation (this file), the release notes of this realese and the online help files for OSC$MGR and OSC$CFG.

- OSC$INCLUDE
  This directory contains all header and object library files required to develop a compiled OSC agent using C (see section 9 Developing new OSC agents).

- OSC$LOG
  This directory contains the common OSC event message files (see section 6 OSC event notification) and the process log files of all OSC components started on any OSC cluster member.

- OSC$SCRATCH
  Scratch directory used by (some) of the OSC agents bundles with VSI OpenVMS ServiceControl.

- OSC$STARTUP

This directory contains backup copies of the OSC startup and logicals definition files stored in SYS$STARTUP and some support command scripts required to install, upgrade and start OSC cluster-wide.