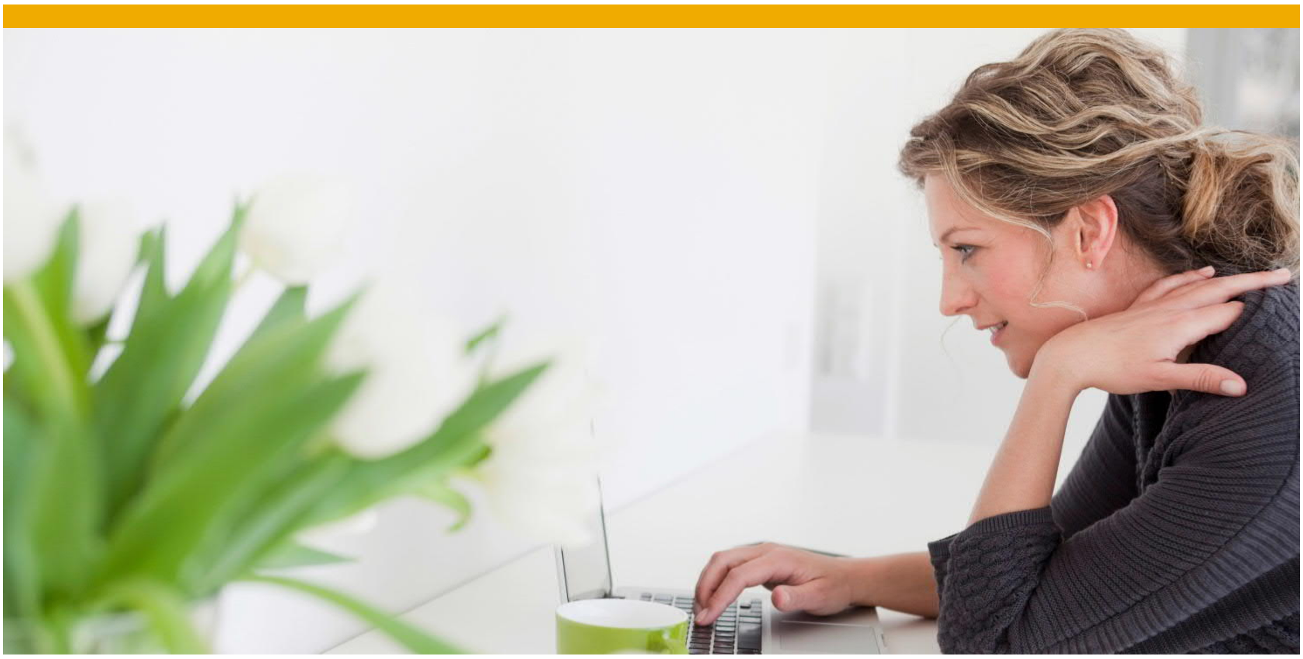


Sapstartsrv in HA environments

Randolf Werner (randolf.werner@sap.com),
SAP NetWeaver Server Infrastructure

7 November 2016

TABLE OF CONTENTS

INTRODUCTION

Sapstartsrv is a Unix daemon / Windows service assigned to a SAP instance. It controls and monitors the assigned instance. Since sapstartsrv is a vital part of any SAP instance and due its specific feature a proper HA setup needs to handle sapstartsrv correctly and can make use of its features. This paper provides some information which should help HA partners to handle it correctly.

There is one additional sapstartsrv installed once per hosts offering OS, database and LM related functionality. The following is not related to this SAP Hostagent sapstartsrv. SAP Hostagent must not be handled by a HA setup. It is installed on a machine and must not be moved by a HA failover.

SAPSTARTSRV STARTUP AND LIFECYCLE

On Windows, sapstartsrv is installed as a Windows service. By default it is installed with startup type “Automatic”. It provides a self-registration / unregistration capability via “-r” and “–u” option.

On UNIX sapstartsrv runs as a daemon. This is achieved by installing “sapinit” once on the machine. Each sapstartsrv gets registered in /usr/sap/services. The self-registration / unregistartion capability is provided via “-reg” and “–unreg” option.

This way sapstartsrv is designed to get started at OS boot and remain running “forever”. Even if the assigned SAP instances gets stopped sapstartsrv remains running. However when applying a kernel patch it will get updated via sapcpe during instance start. In this case sapstartsrv will restart itself 5 minutes after the update happened.

In a HA setup sapstartsrv must be moved together with the assigned instance. Furthermore the HA setup must ensure that sapstartsrv is only running on the cluster node which actually owns the assigned instance. Typically one the following two solutions can be used:

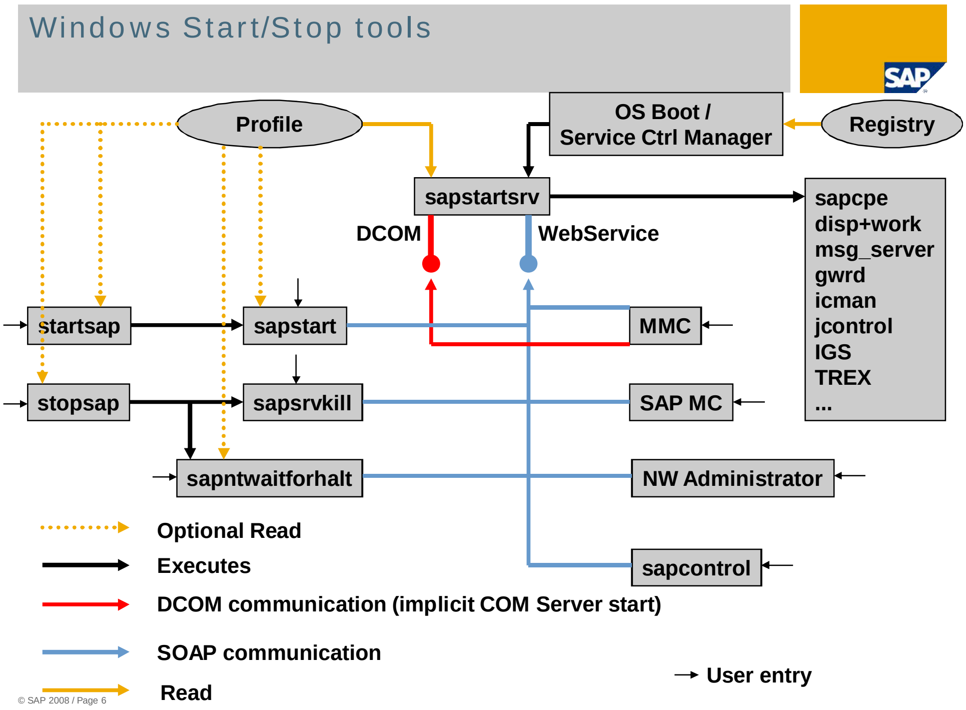
- Sapstartsrv is registered on both/all cluster nodes. In this case the HA solution must ensure it can and will only be started on one of the nodes. E.g. on Windows the service startup type can be switched to “Manually” and service startup is controlled by the HA solution. Another option is to ensure the file system hosting the instance executables is only accessible on one cluster node.
- Sapstartsrv is only registered on the active cluster node. In this case the HA solution must unregister and register sapstartsrv during a failover.

Sapstartsrv can also be restarted manually via its webservice function “RestartService”, e.g. triggered by an Administrator via SAP MMC / MC or sapcontrol to activate certain configuration changes. A HA solution must handle manual and automatic sapstartsrv restarts, e.g. by tolerating short term sapstartsrv unavailable situations. Long term sapstartsrv failure should be handled by the HA solution. The HA solution may try to restart a failed sapstartsrv using function “StartService” of the “sapcontrol” command line tool. If sapstartsrv can be restarted it will pick up the instance processes automatically and continue operating as usual.

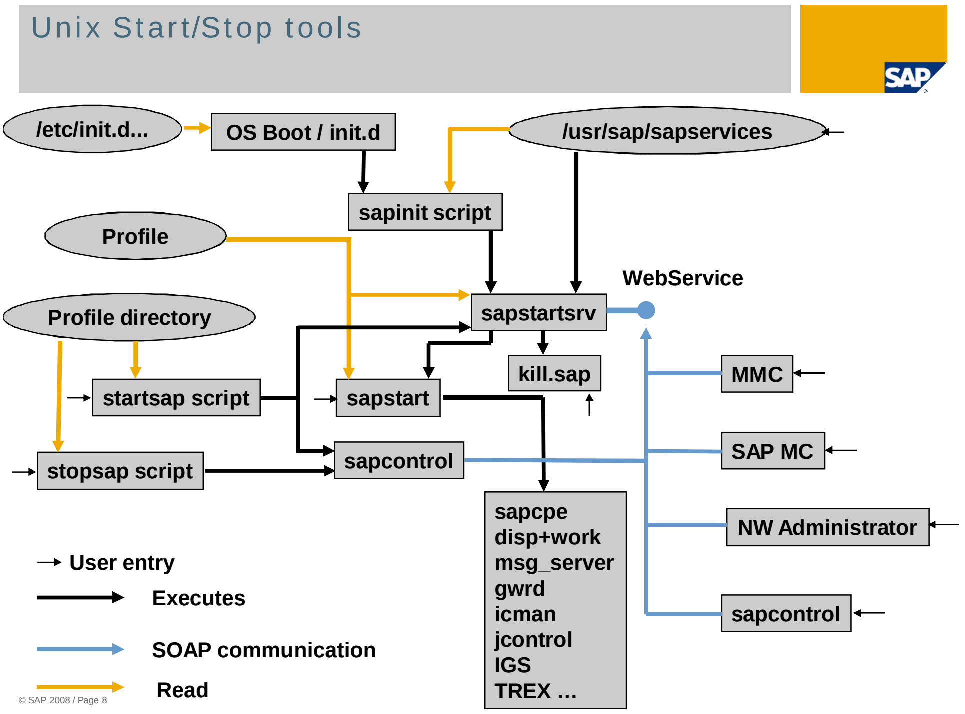
By using webservice function “StopService” sapstartsrv can be stopped without stopping the instance. If sapstartsrv is stopped normally (Windows Service control manager or SIGINT) it will stop the assigned instance as well.

INSTANCE START/STOP

Instance start/stop is triggered via webservice functions “Start” / “Stop” of sapstartsrv. Traditional SAP start/stop command line tools have been adapted to use the webservice interface as well (directly or via “sapcontrol” command line tool). On Windows the deprecated DCOM interface of sapstartsrv still exists but should not be used anymore:



On UNIX, sapstartsrv does not launch the instance process directly instead “sapstart” is used to actually start the processes. Anyway “sapstart” should not be used directly to start the instance (bypassing sapstartsrv):



The actually started processes are configured in the profile using Start_Program_<NR> profile parameters. By using Restart_Program_<NR> parameter instead a processes which died unexpectedly gets restarted automatically. This may e.g. be used to restart the message server in case of failure in order to avoid a complete SCS instance failover if possible or to restart the enqueue replication server of the ERS instances automatically after a successful lock table transfer.

INSTANCE MONITORING

The major webservice function to monitor the instance status is “GetProcessList”. It returns a list of all processes started via the profile. Each process has a status color:

- Gray: Stopped
- Yellow: Running with problems or a synchronous temporary process like “sapcpe” executed during startup
- Green: Running
- Red: Could not be started

For most critical process a status monitoring beyond just “process is running” is performed:

- disp+work: Verify instance is connected to message server, verify dialog queue info can be read from shared memory, verify progress in dialog queue processing, in case of double stack jstart/jcontrol status is integrated
- jstart/jcontrol: Verify status information from AS Java startup framework
- enserver: Verify enqueue statistic can be retrieved from enqueue server (>= 720 patchlevel 68)
- msg_server: Verify message server client list can be retrieved from message server (>= 720 patchlevel 68)

An instance is considered as fully available if all processes have a green status. The overall instance status is reported by sapstartsrv every 60 sec in available.log in DIR_HOME of the assigned instance. During first restart after a kernel switch the status of disp+work, jstart, jcontrol may remain yellow for some minutes although the instance is fully operational until sapstartsrv gets (automatically) restarted. This is caused by using shared memory for monitoring which may cause an old sapstartsrv to fail accessing the shared memory created by a new version of disp+work, jstart, jcontrol. A HA solution needs to handle temporary “yellow” status of processes.

“EnqGetLockTable” and “EnqGetStatistic” can be used to monitor the enqueue server in more detail. Especially “EnqGetStatistic” provides information about the replication state which may be used to avoid triggering failovers in case of missing or failed replication (outstanding enqueue server bugfix in 720 required).

“J2EEGetProcessList” provides additional status information for AS Java.

“GetAlertTree” and “GetAlerts” provide general CCMS status information (similar to rz20 transaction)

“GetSystemInstanceList” provide information on all instances of the system (updated asynchronously every 60 sec by each sapstartsrv in the system).

WEBSERVICE AUTHENTICATION

The webservice interface of sapstartsrv requires client authentication for executing critical functions like “Start” or “Stop”. In <= 720 releases only the most critical functions which alter the instance state are protected by default. In >= 800 by default almost all functions are protected. However SAP customers are free change this setting according their security requirements. A HA product using the webservice interface must be able to provide proper authentication. By default http basic authentication with OS <sid>adm user and password can be used. For local connections a so called “trusted connect” can be used via the “sapcontrol” command line tool (Option: Windows: “-prot PIPE”, UNIX: “-prot NI_HTTP” (default)). This way when executed as <sid>adm all functions of the webservice interface can be executed without explicitly providing OS user and password. For more details and advanced security features please refer to SAP note 927637 and1439348.

SAPCONTROL COMMAND LINE TOOL

The HA product can access the webservice interface of sapstartsrv by using the “sapcontrol” command line tool or by implementing a webservice client on its own. By using the “sapcontrol” tool a trusted connect can be used (see above), however the output of sapcontrol needs to be analyzed by the HA product in this case. Sapcontrol provides some exit codes to detect success or failure of an operation:

0	Last webmethod call successful
1	Last webmethod call failed, invalid parameter
2	StartWait, StopWait, WaitforStarted, WaitforStopped, RestartServiceWait timed out
3	GetProcessList succeeded, all processes running correctly
4	GetProcessList succeeded, all processes stopped

Furthermore option “-format script” can be used to generate an easy to parse output format.

ERROR HANDLING AND CLEANUP

Since sapstartsrv is a mandatory component for normal instance stop additional steps are necessary to stop an instance in case of permanent sapstartsrv failure. In general SAP processes can be stopped using SIGINT signal. A soft shutdown can be triggered via SIGQUIT. Finally processes can be killed by using “TerminateProcess” / SIGKILL. On wWindows sapstartsrv creates a JobObject named <Host>_<SID>_<NR> (will probably be renamed to SAP<SID>_<NR> soon) which hosts all processes of the assigned instance (including sapstartsrv itself). This can be used to find any dangling processes which need to be stopped for cleanup. On UNIX “kill.sap” and “shutdown.sap” are generated in DIR_HOME during instance startup. They can be used to send a SIGINT to “sapstart” or the child processes of “sapstart”.

During normal instance start/stop shared memory is cleaned up by the instance processes. In case of problems this may not always be performed correctly. On Windows SAP shared memory is automatically removed when no more processes are attached to it. On UNIX SAP shared memory remains until it gets explicitly removed using “cleanipc”. However since sapstartsrv remains running after an instance stop and is attached to shared memory “cleanipc” must not be executed manually for an instance while the corresponding sapstartsrv is still running. If necessary cleanipc execution can be configured in the profile during synchronous processes startup (“Execute_<XY>” or “Start_Program_<XY> = immediate”) or instance stop (“Stop_Program_<XY>”).

SAP HA SHARED LIBRARY INTERFACE

In order to align instance start/stop with a HA product, e.g. to avoid unintended failover in case of normal instance stop, a shared library interface must be implemented by the HA vendor. Via profile parameter “service/halib” a shared library to be loaded by sapstartsrv can be configured. The interface defines 8 functions, however actually sapstarstrv only requires “SAP_HA_Init”, “SAP_HA_FindInstance”, “SAP_HA_StartCluster”, “SAP_HA_StopCluster”, the remaining function may be impleneted by providing empty dummy implementations. The package provides a header file with function prototypes to be implemented, a working implementation for Microsoft Cluster Service integration, an empty dummy implementation, a standalone test program to test the shared library and documentation with some test descriptions. For details please refer to the proj/sapha build directories (e.g. \\wdfroot\sapmnt\bas\720_REL\gen\optU\ntamd64\proj\sapha).

DOCUMENTATION

1. How to use the SAPControl Web Service Interface, http://go.sap.com/documents/2016/09/0a40e60d-8b7c-0010-82c7-eda71af511fa.html
2. SAP note 877795, Problems with sapstartsrv as of Release 7.00 & 640 pacth 169, http://service.sap.com/~form/handler?_APP=01100107900000000342&_EVENT=REDIR&_NNUM=877795
3. SAP note 927637, Web service authentication in sapstartsrv as of Release 7.00, http://service.sap.com/~form/handler?_APP=01100107900000000342&_EVENT=REDIR&_NNUM=927637

4. SAP note 1439348, Extended security settings for sapstartsrv, http://service.sap.com/~form/handler?_APP=01100107900000000342&_EVENT=REDIR&_NNUM=1439348
5. SAP note 729945, Auto-restart function for processes in sapstartsrv, http://service.sap.com/~form/handler?_APP=01100107900000000342&_EVENT=REDIR&_NNUM=729945
6. SAP note 768727, Process automatic restart functions in sapstart, http://service.sap.com/~form/handler?_APP=01100107900000000342&_EVENT=REDIR&_NNUM=768727

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any.

Nothing herein should be construed as constituting an additional warranty. In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.
SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See Copyright Information on SAP.com for additional trademark information and notices.