How to use the SAPControl Web Service Interface

19. October 2008
Copyright

© Copyright 2008 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, xApps, xApp, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies (“SAP Group”) for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group 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.
# Content

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>4</td>
</tr>
<tr>
<td>2. Web Service Interface</td>
<td>5</td>
</tr>
<tr>
<td>2.1 General Methods</td>
<td>5</td>
</tr>
<tr>
<td>2.2 ABAP Specific Methods</td>
<td>16</td>
</tr>
<tr>
<td>2.3 AS Java (J2EE) Specific Methods</td>
<td>18</td>
</tr>
<tr>
<td>2.4 ICM Specific Methods</td>
<td>28</td>
</tr>
<tr>
<td>3. Error Handling</td>
<td>30</td>
</tr>
<tr>
<td>4. Web Service Clients</td>
<td>31</td>
</tr>
<tr>
<td>5. Web Server Functionality</td>
<td>33</td>
</tr>
<tr>
<td>6. Logfiles</td>
<td>33</td>
</tr>
<tr>
<td>7. Profile Parameters</td>
<td>33</td>
</tr>
<tr>
<td>8. C# Sample Client Using the SAPControl Interface</td>
<td>34</td>
</tr>
<tr>
<td>9. References</td>
<td>34</td>
</tr>
<tr>
<td>10. Interface Version History</td>
<td>35</td>
</tr>
<tr>
<td>11. Index</td>
<td>38</td>
</tr>
</tbody>
</table>
1 Introduction

The SAP Start Service (sapstartsrv) provides basic management services for systems and instances and single server processes. Services include starting and stopping, monitoring the current run-time state, reading logs, traces and configuration files, executing commands and retrieving other technology-specific information, like network access points, active sessions, thread list etc. They are exposed by a SOAP Web service interface named “SAPControl”. This paper describes how to use this Web service interface.

Since release 7.00 sapstartsrv is available for all SAP supported platforms and used to start and stop the SAP instance. On Windows the ISAPControl DCOM interface is still supported but will be deprecated. The new SAPControl Web service interface should be used instead. It offers significantly improved functionality with respect to AS Java monitoring as well as platform independent monitoring. The Web service interface can be used from any Web service enabled client that can handle the doc/literal communication style (e.g. Java, ABAP, .NET, gSOAP …).

On Windows each SAP instance is started by a specific NT service named SAP<SID>_<NR>. Up to release 4.5A the service was implemented in sapntstartb.exe, which offers a simple proprietary interface via named pipe communication. SAP tools like sservmgr.exe, sapstart.exe, sapsrvkill.exe or sapntwaitforhalt.exe used this named pipe interface to start or stop the SAP system. Since release 4.5B the service has been replaced by sapstartsrv.exe.

The following figure shows the architecture of the components involved.

Architectural Overview

Concerning the GUI components, the overview is not complete. In release 7.1 there is also the NetWeaver Administrator and the Developer Studio acting as frontend communicating with the start service.
2 Web Service Interface

The start service offers its Web service interface on port sapctrl<NR> (HTTP) and sapctrls<NR> (HTTPS) where "<NR>" corresponds to the SAP instance number (00…98). If the ports are not defined in etc/services, the default values 5<NR>13 (HTTP) and 5<NR>14 (HTTPS) are used. HTTPS is only available if SAP standard SSL Software “Secude” and the required certificates are installed. Sapstartsrv uses the same certificates as the other parts of the instance (icman, message server, …). Instance number 99 is reserved for SAPHostControl, which is installed once per host to perform host instead of instance specific tasks, e.g. adaptive computing or saposcol. It uses IANA registered ports 1128 / saphostctrl and 1129 / saphostctrls. SAPHostControl is not described in this paper.

On Unix a trusted local connect via Unix domain sockets (SAP NI standard naming /tmp/.sapstream<port-nr>) is also possible. On Windows a trusted connect via named pipe \<hostname\>\pipe\sapcontrol_<NR> is possible. There is no authentication check (see below) for trusted connects. This enables a client to use the protected methods (see below) without any additional authentication in a secure way.

If LDAP and/or SLD registration are configured (profile parameter ldap/autoregister=1 / slddest.cfg present in DIR_GLOBAL directory), the service registers itself during service startup in an LDAP directory or SAP System Landscape Directory (SLD). Especially it will register the necessary information to bind to the old DCOM and new HTTP/HTTPS interface. The LDAP registration will use the SAP-R3-ServiceConnectionPoint class with CN=ControlService, CN=ControlService_HTTP and CN=ControlService_HTTPS for registration. The SLD registration will use the SAP_BCControlInstance class with name=<SID>.HostName.<Host>.InstanceNumber.<NR>. Please refer to the “SAP System Information in Directory Services” document on the SAP Service Marketplace and the SLD documentation for further details. A trace of the registration process will be written in the working directory of the SAP instance (dev_ldaps, dev_sldregs).

The access to critical methods of the Web service is protected (currently by default: Start, Stop, RestartInstance, Shutdown, StartSystem, StopSystem, J2EEControlProcess, SendSignal, OSExecute, J2EEnableDbgSession, J2EDisableDbgSession, SetProcessParameter, EnqRemoveLocks). The list of protected methods can be changed by using the start profile parameter “service/protectedwebmethods” (blank separated list of method names). To use these methods one has to provide a valid OS user and password via HTTP basic authentication encoded as UTF8 or must use a trusted connect. The service will verify the given credentials, and grant permission only to valid users that additionally have execute permission on the sapstartsrv executable file. Otherwise the request will fail with “Invalid Credentials” or "Permission denied" fault string. Missing credentials when accessing a critical method will result in HTTP error 401. Windows users may be given in format <domain>\<user> or <user>@<domain>. On Unix sapstartsrv will ignore the domain user part. On Windows sapstartsrv will try any trusted domain if no domain is given.

The Web service interface is implemented in C++ by using gSOAP 2.7. Doc/literal encoding style is used. The WSDL interface definition can be obtained directly from the Web service using http://<host>:<port>/?wsdl. It can be used to generate a client proxy in Web service enabled programming environments, like gSOAP, Axis, Microsoft .NET, SAP ABAP, SAP J2EE.

Most methods use similar in and out parameters. Some methods like “Shutdown” require no parameters at all. Some others like “SendSignal” require input parameters. Most of the methods return information in a table like data structure (e.g. “GetProcessList”). The interface is using SOAP exception and HTTP error code for error handling. Below you will find the currently implemented methods in the format gSOAP uses for Web service definition (without “SAPControl__” namespace prefix). The last parameter of each method defines the SOAP response (output parameter). All other parameters define input parameters for a method.

2.1 General Methods

Start(struct StartResponse{} *out)
Stop( int softtimeout=0,
        struct StopResponse{} *out)
RestartInstance( int softtimeout=0,
                  struct RestartInstanceResponse{} *out);
Shutdown(struct ShutdownResponse{} *out)
Use these functions to start, stop or restart a SAP instance. **Start** triggers an instance start. **Stop** triggers an instance stop. **softtimeout** specifies a timeout in sec for a soft shutdown via SIGQUIT, if the timeout expires a hard shutdown is used. **Shutdown** triggers a soft shutdown via SIGQUIT. **RestartInstance** triggers an instance restart. All functions work asynchronously, which means they trigger the operation and return immediately.

```c
// RestartService
struct RestartServiceResponse{} *out
```

```c
// StopService
struct StopServiceResponse{} *out
```

Use these functions to restart of stop the sapstartsrv Web service. However once the Web service is stopped you have to start sapstartsrv before using the Web service interface again.

```c
// ParameterValue
char *parameter,
char **value);
```

Returns a SAP profile parameter **value** for a given profile **parameter**. If the given profile **parameter** is empty it returns a string with all known parameter value pairs separated by newline.

```c
// GetProcessList
ArrayOfOSProcess *process);
```

Returns a list of all processes directly started by the sapstartsrv Web service according to the SAP start profile.

```c
// StartSystem
enum StartStopOption options,
char *prioritylevel,
int waittimeout,
struct StartSystemResponse{} *out);
```

```c
// StopSystem
enum StartStopOption options,
char *prioritylevel,
int softtimeout,
int waittimeout,
struct StopSystemResponse{} *out);
```
### How to use the SAPControl Web Service Interface

#### SAP Technical Documentation

```c
enum StartStopOption
{
    SAPControl_ALL_INSTANCES    = 0,
    SAPControl_SCS_INSTANCES    = 1,
    SAPControl_DIALOG_INSTANCES = 2,
    SAPControl_ABAP_INSTANCES   = 3,
    SAPControl_J2EE_INSTANCES   = 4,
    SAPControl_PRIORITY_LEVEL   = 5,
    SAPControl_TREX_INSTANCES   = 6,
    SAPControl_ENQREP_INSTANCES = 7
};
```

Use these function to start, stop or restart a complete SAP system or parts of it. **StartSystem** triggers a system start. **StopSystem** triggers a system stop. **RestartSystem** triggers a system restart. **options** defines which instances to start/stop/restart. If **SAPControl_PRIORITY_LEVEL** is used, **prioritylevel** defines up/down to which instance priority level instances should be started/stopped. **waittimeout** specifies a timeout in sec to wait for an instance to start/stop. If the timeout expires during a start operation remaining instances with a higher instance priority are not started, since they rely on the other instances to be running. If the timeout expires during a stop operation, the operation will continue stopping the remaining instances. **softtimeout** specifies a timeout in sec for a soft shutdown via SIGQUIT, if the timeout expires a hard shutdown is used for the remaining instances. All functions work asynchronously just triggering the operation and returning immediately.

```c
GetStartProfile(class GetStartProfileResponse
    (char *name; ArrayOfString lines;) *file)
```

Returns start profile name and its content.

```c
GetTraceFile(class GetTraceFileResponse
    (char *name; ArrayOfString lines;) *file)
```

Returns the sapstartsrv Web service trace file name and its content.

```c
ListDeveloperTraces(ArrayOfDirEntry *file)
```

Returns the developer trace file names and its content.
Returns a list of all instance trace files in DIR_HOME (superseded by ListLogFiles). A trace file can be read by using ReadDeveloperTrace.

ReadDeveloperTrace(char *filename, int size, class ReadDeveloperTraceResponse {char *name; ArrayOfString lines;} *file)
enum STATE_COLOR HighAlertValue;
char *AlDescription;
char *AlTime;
char *Tid;
}

class ArrayOfAlertNode
{
   AlertNode *__ptr;
   int __size;
};

Returns CCMS Alert tree as an array. The parent-child node relationship is encoded via the parent index of each node. (similar to rz20 transaction).

GetAlerts( char *RootTid,
           GetAlertsResponse *alertlist)

enum STATE_COLOR
{
   SAPControl_GRAY   = 1,
   SAPControl_GREEN  = 2,
   SAPControl_YELLOW = 3,
   SAPControl_RED    = 4
};

class Alert
{
   char *Object;
   char *Attribute;
   enum STATE_COLOR Value;
   char *Description;
   char *Time;
   char *Tid;
   char *Aid;
};

class ArrayOfAlert
{
   Alert *__ptr;
   int __size;
};

class GetAlertsResponse
{
   char *RootTidName;
   ArrayOfAlert alert;
};

Returns a list of all CCMS alerts for a given node and its child nodes.

SendSignal( int pid,
            char *signal,
            struct SendSignalResponse() *out)

Sends a given OS signal to a process specified by its pid. The signal can be given by name (HUP, INT, QUIT,ILL, TRAP, ABRT, IOT, BUS, FPE, KILL, SIG, USR1, SEGV, USR2, SIG, PIPE, ALRM, TERM, STKFLT, CHLD, CONT, STOP, TSTP) or number. OS signals are platform depended, some signals are not supported by all platforms.
**GetVersionInfo** (ArrayOfInstanceVersionInfo *version)

```
class InstanceVersionInfo
{
    char *Filename;
    char *VersionInfo;
    char *Time;
}
```

```
class ArrayOfInstanceVersionInfo
{
    InstanceVersionInfo *__ptr;
    int __size;
}
```

Returns a list of version information for the most important files of the instance.

**GetQueueStatistic** (ArrayOfTaskHandlerQueue *queue)

```
class TaskHandlerQueue
{
    char *Typ;
    int Now;
    int High;
    int Max;
    int Writes;
    int Reads;
}
```

```
class ArrayOfTaskHandlerQueue
{
    TaskHandlerQueue *__ptr;
    int __size;
}
```

Returns a list of queue information of ABAP work processes and icm.

**OSExecute** (char *command, int async, int timeout, char *protocolfile,

```
class OSExecuteResponse
{
    int exitcode;
    Int pid;
    ArrayOfString lines;)
```

```
class ArrayOfString
{
    char **__ptr;
    int __size;
}
```

Executes an external OS command. Use **async=0** to execute the command synchronously. The Web service method returns when the command has finished or the **timeout** (specified in sec, 0=infinite) is reached. If the timeout is reached the process will be terminated. Use **async=1** to execute the command asynchronous. The Web service method will return immediately. stdout/stderr of the command can be redirected to a **protocolfile**. Use **protocolfile=** for getting the result in the **lines** output parameter for synchronous commands or redirecting it to the OS NULL device for asynchronous commands. Protocol files will not be deleted automatically by sapstartsrv.
GetInstanceProperties(ArrayOfInstanceProperties *properties)

class InstanceProperty
{
    char  *property;
    char  *propertytype;
    char  *value;
};

class ArrayOfInstanceProperties
{
    InstanceProperty *__ptr;
    int __size;
};

Returns a list of available instance features and which Web service methods are supported to get the
information. GetInstanceProperties provides some meta information about the instance, which allows a
client to display only information relevant for the actual instance type and version. It also enables a client to
work with multiple versions of the Web service interface. Currently 3 propertytype values are defined.

"NodeWebmethod" is used for nodes which provide information via Web service methods. A client should use property as node name for displaying the information and use any of the Web service methods defined by value. A client should use the leftmost method in the methods list it is supporting, e.g.:

property="J2EE Caches"
propertytype="NodeWebmethod"
value="J2EEGetCacheStatistic2,J2EEGetCacheStatistic"

The client should display the information as “J2EE Caches” and use webmethod J2EEGetCacheStatistic2
to get the information. Older clients not aware of J2EEGetCacheStatistic2 can still use J2EEGetCacheStatistic to get most of the information.

"NodeURL" is used for nodes which provide information via a generic URL, e.g.:

property="ICM"
propertytype="NodeURL"
value=HTTP://WDFD00155758A:56000/sap/admin

The client should display the information as “ICM” and use HTTP://WDFD00155758A:56000/sap/admin to
display additional information about the node.

"Attribute" is used to provide additional information about the instance, e.g.:

property="StartPriority"
propertytype= “Attribute”
value= “3”

property="Protected Webmethods"
propertytype= “Attribute”
value="Start,Stop,Shutdown,StartSystem,StopSystem,StopService,J2EEControlProcess,SendSignal,OSExecute"
How to use the SAPControl Web Service Interface

ReadLogFile

```
ReadLogFile(char *filename, char *filter, char *language, int maxentries, char *statecookie, class ReadLogFileResponse *log)
```

Returns the content of a given log file defined by `filename`. `filename` must match with one of the log files returned by `ListLogFiles`. `ReadLogFile` can read various file types like plain text, ABAP Syslog or J2EE log files.

`filter` can be used to limit the result to certain columns and only matching entries:

- `filter="\"": Read all entries and columns.
- `filter="<Column1>#<Column2>#...#<ColumnN>\"": Read all entries but only specified columns, e.g.:
  - `filter="Time#Severity\"\"`
- `filter="[CASEIGNORE:\"]<Column1><PatternSet1>#<Column2><PatternSet2>#...#<ColumnN><PatternSetN>\""
  - Use "CASEIGNORE:" at filter beginning to define a case insensitive pattern matching
  - PatternSet syntax: <Pattern1>|…|<PatternN>
  - Pattern syntax:
    - "=...." lexicographical equal, use "*,?" for wildcard matching
    - "!...." lexicographical non equal, use "*,?" for wildcard matching
    - "<...." lexicographical smaller
    - "(...." semantical smaller
    - ">...." lexicographical bigger
    - ">...." semantical bigger
    - "]<Begin>,<End>" lexicographical between <Begin> and <End
    - "]<Begin>,<End>" lexicographical outside <Begin> and <End> interval

  e.g.:
  - `filter="CASEIGNORE:Time#Severity)Info#Text=*timeout*|=*null*"
```

`language` is reserved for future usage.

`statecookie` specifies the starting position to read from. Use `statecookie="\""` to read from the beginning, `statecookie="EOF"` to read from the end or `statecookie=<endcookie>` / `statecookie=<startcookie>` to continue reading from a previous call returning `endcookie` / `startcookie`.

Use `maxentries` to specify an upper limit of returned entries (0=all) and reading direction (>0: forward,<0: backward).

On return `format` contains a “#” separated string containing the column names, e.g.:

**J2EE log file:**
- `format="Version#Guid#Time#SourceName#Application#Location#User#Session#Transaction#DSRComponent#DSRUser#DSRTransaction#ThreadName#GroupId#GroupLevel#GroupIndent#Severity#Relatives#MsgType#MsgCode#ResourceBundle#Text"

**Plain text file:**
- `format="Line"`
ABAP Syslog:
format＝"Severity#Time#Typ#Client#User#Tcode#MNo#Pid#Terminal#Program#Session#Text"

startcookie / endcookie identify file start and end position of the response and can be used additional calls of ReadLogFile to continue reading.
fields contains the log file entries. A log entry corresponds to a single string. The columns of an entry are separated by Tabs matching with the format string.
ReadLogFile supercedes ReadDeveloperTracse.

ListLogFiles (ArrayOfLogFile *file)

    class LogFile
    {
        char *filename;
        unsigned int size;
        char *modtime;
        char *format;
    };
    class ArrayOfLogFile
    {
        LogFile *__ptr;
        int __size;
    };

Returns a list of all instance log files (supersedes ListDeveloperTraces). format identifies the log file format ("Text", "J2EE Fileset", "J2EE Fileset Part", "SAP Syslog"). A log file can be read by using ReadLogFile.

AnalyseLogFiles ( char *starttime,
                char *endtime,
                int severity_level=2,
                int maxentries = 10000,
                class AnalyseLogFilesResponse
                {
                    char *format;
                    ArrayOfString fields;
                } *log)

    class ArrayOfString
    {
        char **__ptr;
        int __size;
    };

Scans all log files for a given time period and returns a merged list of all matching log file entries. Time period of interest can be defined by starttime and endtime (Format: "YYYY MM DD HH:MM:SS"), if not defined the last 10 minutes of the last instance run are used. severity_level the log entry severity level to serach for (2=Only errors, 1=Errors and Warnings, 0=All). Use maxentries to limit the amount of log entries to return.

GetAccessPointList (ArrayOfAccessPoint *accesspoint)

    class AccessPoint
    {
        char *address;
        int port;
        char *protocol;
        char *processname;
        char *active;
    };

13
class ArrayOfAccessPoint
{
    AccessPoint *__ptr;
    int __size;
};

Returns a list of all network access points of the instance.

GetSystemInstanceList(ArrayOfSAPInstance *instance)

enum STATE_COLOR
{
    SAPControl_GRAY   = 1,
    SAPControl_GREEN  = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED    = 4
};

class SAPInstance
{
    char *hostname;
    int  instanceNr;
    int  httpPort;
    int  httpsPort;
    char *startPriority;
    char *features;
    enum STATE_COLOR dispstatus;
};

class ArrayOfSAPInstance
{
    SAPInstance *__ptr;
    int __size;
};

Returns a list of all instances of the SAP system. features identifies the instance type (ABAP, J2EE, GATEWAY, MESSAGESERVER, ENQUE, ICMAN, TREX, IGS, ENQREP), e.g.:

Dual-stack dialog instance: “ABAP|J2EE|GATEWAY|ICMAN”
SCS instance: “MESSAGESERVER|ENQUE”

AccessCheck(char *function, struct AccessCheckResponse{} *out)

Check if execution of the specified webmethod is granted.

GetProcessParameter(char *processtype, int pid = -1, ArrayOfProfileParameter *parameter)

enum RESTRICTION_TYPE
{
    SAPControl_RESTRICT_NONE = 0,
    SAPControl_RESTRICT_INT = 1,
    SAPControl_RESTRICT_FLOAT = 2,
    SAPControl_RESTRICT_INTRANGE = 3,
    SAPControl_RESTRICT_FLOATRANGE = 4,
    SAPControl_RESTRICT_ENUM = 5,
    SAPControl_RESTRICT_BOOL = 6
};
class ArrayOfString
{
    char **__ptr;
    int __size;
};

class ParameterRestriction
{
    enum RESTRICTION_TYPE    type;
    LONG64  *int_min;
    LONG64  *int_max;
    double  *float_min;
    double  *float_max;
    ArrayOfString *valuemap;
};

class SubProfileParameter
{
    char    *name;
    char    *description;
    char    *unit;
    bool    mandatory;
    ParameterRestriction  restriction;
};

class ArrayOfSubProfileParameter
{
    SubProfileParameter *__ptr;
    int __size;
};

class SAPControl__ProfileParameter
{
    char    *name;
    char    *group;
    char    *description;
    char    *unit;
    ParameterRestriction  restriction;
    ArrayOfSubProfileParameter  *sub_para;
    char    *value;
    char    *dynamic_value;
    ArrayOfString    *values;
    ArrayOfString    *dynamic_values;
};

class ArrayOfProfileParameter
{
    ProfileParameter *__ptr;
    int __size;
};

Returns a list of actual profile parameters for a given process. Known **processtype** values are “ICM”, “Web Dispatcher”, “MessageServer”, “Gateway”, “EnqueueServer”, “Dispatcher”. **pid** needs to be set if multiple processes of the same type exist within the instance.

```c
SetProcessParameter( char *processtype,
                    int pid = -1,
                    ArrayOfSetProfileParameter parameter,
                    struct SetProcessParameterResponse{} *out);
```
Sets dynamic Profile Parameters for a given process. Known `processtype` values are “ICM”, “Web Dispatcher”, “MessageServer”, “Gateway”, “EnqueueServer”, “Dispatcher”. `pid` needs to be set if multiple processes of the same type exist within the instance.

```cpp
GetNetworkId(char *service_ip,
int service_port,
int version,
    class SAPControl__GetNetworkIdResponse{ char *key; } *id);```

Returns a unique network ID for a network service given by `service_ip` and `service_port`. `version` specifies the algorithm version used to calculate the ID. Since this function doesn’t provide any verification, it should only be used get the network ID value (e.g. for requesting a matching license in advance). To verify the network ID (e.g. during license verification `GetSecNetworkId` should be used instead).

```cpp
GetSecNetworkId(char *service_ip,
int service_port,
int version,
char *challenge,
    class GetSecNetworkIdResponse{ char *key;
char *proof; } *id);```

Returns a unique network ID for a network service given by `service_ip` and `service_port` and a verification `proof` based on `service_ip`, `service_port` caller defined (typically random) `challenge` and `key`. `version` specifies the algorithm version used to calculate the ID. The caller can use the `proof` to verify authenticity of the response. If `challenge` is not given it is read from the Message Server. Since `GetSecNetworkId` additionally uses the client IP address from the actual socket communication the result may differ from GetNetworkId in case `service_ip` is not the real client IP address used to connect to SAPControl Web service.

### 2.2 ABAP Specific Methods

```cpp
ABAPReadSyslog(ArrayOfSyslogEntry *log)
enum STATE_COLOR
{   
    SAPControl_GRAY   = 1,
    SAPControl_GREEN  = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED    = 4
};```
class SyslogEntry
{
    char *Time;
    char *Typ;
    char *Client;
    char *User;
    char *Tcode;
    char *MNo;
    char *Text;
    enum STATE_COLOR Severity;
}

class ArrayOfSyslogEntry
{
    SyslogEntry *__ptr;
    int __size;
};

Reads the ABAP Syslog and returns it as an array of entries (similar to SM21 transaction).

\textbf{ABAPReadRawSyslog}(ArrayOfRawSyslogEntry *log)

class ArrayOfRawSyslogEntry
{
    char **__ptr;
    int __size;
};

Reads the SAP ABAP Syslog and returns the raw file content.

\textbf{ABAPGetWPTable}(ArrayOfWorkProcess *workprocess)

class WorkProcess
{
    int No;
    char *Typ;
    int Pid;
    char *Status;
    char *Reason;
    char *Start;
    char *Err;
    char *Sem;
    char *Cpu;
    char *Time;
    char *Program;
    char *Client;
    char *User;
    char *Action;
    char *Table;
};

class ArrayOfWorkProcess
{
    WorkProcess *__ptr;
    int __size;
};

Returns a list of the ABAP work processes (similar to SM50 transaction).
ABAPAcknowledgeAlerts(    char  *R3Client,    char  *R3User,    char  *R3Password,    ArrayOfString  Aid,    ArrayOfInt  *alert)

class ArrayOfString    {
    char **__ptr;
    int __size;
    }

class ArrayOfInt    {
    int *__ptr;
    int __size;
    }

Acknowledge CCMS Alerts in the SAP ABAP system. Requires SAP user credentials and a list of alert ids to acknowledge. Returns a list of success code for each alert (1=sucess, 0=failure).

2.3 AS Java (J2EE) Specific Methods

J2EEGetProcessList(ArrayOfJ2EEProcess *process)

enum J2EE_PSTATE    {
    SAPControl_J2EE_STOPPED = 1,
    SAPControl_J2EE_STARTING = 2,
    SAPControl_J2EE_CORE_RUNNING = 3,
    SAPControl_J2EE_RUNNING = 4,
    SAPControl_J2EE_STOPPING = 5,
    SAPControl_J2EE_MAINTENANCE = 6,
    SAPControl_J2EE_UNKNOWN = 7
    };

class J2EEProcess    {
    int    telnetPort;
    char *name;
    int    pid;
    char *type;
    char *restart;
    char *exitCode;
    enum J2EE_PSTATE state;
    char *statetext;
    char *startTime;
    char *elapsedTime;
    int    restartCount;
    int    errorCount;
    char *cpu;
    char *debug;
    }

class ArrayOfJ2EEProcess    {
    J2EEProcess *__ptr;
    int __size;
    };
Returns a list of AS Java server processes (j2ee processes) controlled by jcontrol / jstart (superseded by J2EEGetProcessList2).

J2EEGetProcessList2(ArrayOfJ2EEProcess2 *process)

enum J2EE_PSTATE
{
    SAPControl_J2EE_STOPPED = 1,
    SAPControl_J2EE_STARTING = 2,
    SAPControl_J2EE_CORE_RUNNING = 3,
    SAPControl_J2EE_RUNNING = 4,
    SAPControl_J2EE_STOPPING = 5,
    SAPControl_J2EE_MAINTENANCE = 6,
    SAPControl_J2EE_UNKNOWN = 7
};

class J2EEProcess2
{
    int    telnetPort;
    char *name;
    int    pid;
    char *type;
    char *restart;
    char *exitCode;
    enum J2EE_PSTATE state;
    char *statetext;
    char *startTime;
    char *elapsedTime;
    int    restartCount;
    int    errorCount;
    char *cpu;
    char *debug;
    int  clusterId;
};

class ArrayOfJ2EEProcess2
{
    J2EEProcess2 *__ptr;
    int __size;
};

Returns a list of AS Java processes controlled by jcontrol / jstart (supersedes J2EEGetProcessList).

J2EEControlProcess(char *processname,
                    char *function,
                    struct J2EEControlProcessResponse{} *out)


J2EEGetThreadList(ArrayOfJ2EEThread *thread)
enum STATE_COLOR
{
    SAPControl_GRAY  = 1,
    SAPControl_GREEN = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED    = 4
};

class J2EEThread
{
    char  *processname;
    char  *startTime;
    char  *updateTime;
    char  *taskupdateTime;
    char  *task;
    char  *subtask;
    char  *name;
    char  *classname;
    char  *user;
    char  *pool;
    char  *state;
    enum STATE_COLOR dispstatus;
};

class ArrayOfJ2EEThread
{
    J2EEThread  *__ptr;
    int  __size;
};

Returns a list of threads in the AS Java instance (superseded by J2EEGetThreadList2).

J2EEGetThreadList2(ArrayOfJ2EEThread2  *thread)

enum STATE_COLOR
{
    SAPControl_GRAY  = 1,
    SAPControl_GREEN = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED    = 4
};

class J2EEThread2
{
    char  *processname;
    char  *startTime;
    char  *updateTime;
    char  *taskupdateTime;
    char  *task;
    char  *subtask;
    char  *name;
    char  *classname;
    char  *user;
    char  *pool;
    char  *state;
    enum STATE_COLOR dispstatus;
    int  index;
};
class ArrayOfJ2EEThread2
{
    J2EEThread2 *__ptr;
    int __size;
};

Returns a list of threads in the AS Java instance (supersedes J2EEGetThreadList).

J2EEGetSessionList (ArrayOfJ2EESession *session)

class J2EESession
{
    char *processname;
    int IdHash;
    int size;
    int timeout;
    int activeRequests;
    char *startTime;
    char *updateTime;
    char *sticky;
    char *corrupt;
    char *backingStore;
};

class ArrayOfJ2EESession
{
    J2EESession *__ptr;
    int __size;
};

Returns a list of (HTTP) sessions in the AS Java instance (superseded by J2EEGetWebSessionList).

J2EEGetWebSessionList (ArrayOfJ2EEWebSession *session)

class J2EEWebSession
{
    char *processname;
    int IdHash;
    int size;
    int timeout;
    int activeRequests;
    char *startTime;
    char *updateTime;
    char *state;
    char *backingStore;
    char *user;
};

class ArrayOfJ2EEWebSession
{
    J2EEWebSession *__ptr;
    int __size;
};

Returns a list of (HTTP) sessions in the AS Java instance (supersedes J2EEGetSessionList).

J2EEGetCacheStatistic (ArrayOfJ2EECache *cache)
enum STATE_COLOR
{
    SAPControl_GRAY = 1,
    SAPControl_GREEN = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED = 4
};

class J2EECache
{
    char *cachename;
    char *processname;
    char *type;
    LONG64 size;
    LONG64 attrSize;
    LONG64 keysSize;
    int cachedObjects;
    int usedObjects;
    int puts;
    int gets;
    int hits;
    int changes;
    int removes;
    int evictions;
    int instanceInvalidations;
    int clusterInvalidations;
    char *updateTime;
    enum STATE_COLOR dispstatus;
};

class ArrayOfJ2EECache
{
    J2EECache *__ptr;
    int __size;
};

Returns a list of caches in the AS Java instance (superseded by J2EEGetCacheStatistic2).

J2EEGetCacheStatistic2(ArrayOfJ2EECache2 *cache)

enum STATE_COLOR
{
    SAPControl_GRAY = 1,
    SAPControl_GREEN = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED = 4
};

class J2EECache2
{
    char *description;
    char *owner;
    char *processname;
    char *type;
    LONG64 size;
    LONG64 attrSize;
    LONG64 keysSize;
    int cachedObjects;
    int usedObjects;
    int puts;
    int gets;
int hits;
int changes;
int removes;
int evictions;
int instanceInvalidations;
int clusterInvalidations;
char *updateTime;
enum STATE_COLOR dispstatus;
};

class ArrayOfJ2EECache2
{
    J2EECache2 *__ptr;
    int __size;
};

Returns a list of caches in the AS Java instance (supersedes J2EEGetCacheStatistic).

J2EEGetApplicationAliasList (ArrayOfJ2EEApplicationAlias *alias)

class J2EEApplicationAlias
{
    char *AppName;
    char *Alias;
    int TotalRequests;
    char *AppActive;
    char *IgnoreCookie;
};

class ArrayOfJ2EEApplicationAlias
{
    J2EEApplicationAlias *__ptr;
    int __size;
};

Returns a list of application aliases in the AS Java instance.

J2EEGetComponentList (ArrayOfJ2EEComponentInfo *component)

enum STATE_COLOR
{
    SAPControl_GRAY   = 1,
    SAPControl_GREEN  = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED    = 4
};

class J2EEComponentInfo
{
    char *type;
    char *name;
    char *startupmode;
    char *status;
    char *expectedstatus;
    char *details;
    enum STATE_COLOR dispstatus;
};
class ArrayOfJ2EEComponentInfo
{
    J2EEComponentInfo *__ptr;
    int __size;
};

Returns a list of configured J2EE components (services and applications).

J2EEGetEJBSessionList(ArrayOfJ2EEEJBSession *ejbsession)

class J2EEEJBSession
{
    int IdHash;
    char *state;
    int size;
    int activeRequests;
    int totalRequests;
    char *backingStore;
    char *processname;
    char *startTime;
    char *updateTime;
    int responseTime;
    char *user;
    char *transaction;
    char *ejb;
    char *application;
    char *reference;
};

class ArrayOfJ2EEEJBSession
{
    J2EEEJBSession *__ptr;
    int __size;
};

Returns a list of EJB sessions in the AS Java instance.

J2EEGetRemoteObjectList(ArrayOfJ2EERemoteObject *remoteobject)

class J2EERemoteObject
{
    int IdHash;
    char *address;
    int port;
    char *protocol;
    char *direction;
    int stubs;
    int implementations;
    char *creationTime;
    char *updateTime;
    char *processname;
};

class ArrayOfJ2EERemoteObject
{
    J2EERemoteObject *__ptr;
    int __size;
};

Returns a list of remote object connections in the AS Java instance.
J2EEGetClusterMsgList(ArrayOfJ2EEClusterMsg *msg)

```cpp
class J2EEClusterMsg {
    char *service;
    char *id;
    LONG64 count;
    LONG64 length;
    LONG64 avg_length;
    LONG64 max_length;
    LONG64 count_p2p_msg;
    LONG64 count_p2p_request;
    LONG64 count_p2p_reply;
    LONG64 count_broadcast_msg;
    LONG64 count_broadcast_reply;
};

class ArrayOfJ2EEClusterMsg {
    J2EEClusterMsg *__ptr;
    int __size;
};
```

Returns a list of J2E cluster communication statistic from the message server.

J2EEGetSharedTableInfo(ArrayOfJ2EESharedTableInfo *jsf)

```cpp
class J2EESharedTableInfo {
    char *table;
    int used;
    int peak;
    int limit;
    enum STATE_COLOR dispstatus;
};

class ArrayOfJ2EESharedTableInfo {
    J2EESharedTableInfo *__ptr;
    int __size;
};
```

Returns a list of SAP startup framework shared memory table information.

J2EEEnableDbgSession(char *processname, char *flags, char *client, class J2EEEnableDbgSessionResponse*debuginfo)

```cpp
Creates a J2E debug session on a specific AS Java server process given by "processname". Use "" as processname for automatic node selection. "flags" defines a set of debug flags given as a blank separated list of keywords ("SuspendAll", "CodexIsolate", "LoadIsolate", "MigrateSessions", "KeepSession", ```
“NoDebugger”), default value is “LoadIsolate MigrateSessions”. “client” identifies the calling client “<user>@<host>” for monitoring. On success debug key and network port are returned.

**J2EEDisableDbgSession**

```c
char *key, struct J2EEDisableDbgSessionResponse{} *out
```

Removes a J2EE debug session given by “key” parameter previously created by “J2EEEnableDbgSession”.

**J2EEGetThreadCallStack**

```c
int index,
class J2EEGetThreadCallStackResponse{
    char *name;
    ArrayOfString lines;}
```

### class ArrayOfString

```c
{char **__ptr;
 int __size;
}
```

Returns the java callstack of a given java thread (“index” parameter returned by “J2EEGetThreadList2”) or all java threads (index=-1).

**J2EEGetThreadTaskStack**

```c
int index,
class J2EEGetThreadTaskStackResponse{
    char *name;
    ArrayOfString lines;}
```

### class ArrayOfString

```c
{char **__ptr;
 int __size;
}
```

Returns the J2EE taskstack of a given java thread (“index” parameter returned by J2EEGetThreadList2) or all java threads (index=-1).

**J2EEGetVMGCHistory**

```c
ArrayOfGCInfo *gc
```

### class GCInfo

```c
{char *processname;
 char *type;
 char *reason;
 char *startTime;
 int duration;
 int cpuTime;
 LONG64 objBytesBefore;
 LONG64 objBytesAfter;
 LONG64 objBytesFreed;
 LONG64 clsBytesBefore;
 LONG64 clsBytesAfter;
 LONG64 clsBytesFreed;
 LONG64 heapSize;
 int unloadedClasses;
}
class ArrayOfGCInfo
{
    GCInfo *__ptr;
    int __size;
};

Returns a list of JAVA VM garbage collections in the AS Java instance (superseded by J2EEGetVMGCHistory2).

J2EEGetVMGCHistory2(ArrayOfGCInfo2 *gc)

enum STATE_COLOR
{
    SAPControl_GRAY   = 1,
    SAPControl_GREEN  = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED    = 4
};

class GCInfo2
{
    char *processname;
    char *type;
    char *reason;
    char *startTime;
    int  duration;
    int  cpuTime;
    LONG64 objBytesBefore;
    LONG64 objBytesAfter;
    LONG64 objBytesFreed;
    LONG64 clsBytesBefore;
    LONG64 clsBytesAfter;
    LONG64 clsBytesFreed;
    LONG64 heapSize;
    int  unloadedClasses;
    LONG64 pageFaults;
    enum STATE_COLOR dispstatus;
};

class ArrayOfGCInfo2
{
    GCInfo2 *__ptr;
    int __size;
};

Returns a list of JAVA VM garbage collections in the AS Java instance (supersedes J2EEGetVMGCHistory).

J2EEGetVMHeapInfo(ArrayOfHeapInfo *heap)

enum STATE_COLOR
{
    SAPControl_GRAY   = 1,
    SAPControl_GREEN  = 2,
    SAPControl_YELLOW = 3,
    SAPControl_RED    = 4
};
class HeapInfo
{
    char *processname;
    char *type;
    LONG64 size;
    LONG64 commitSize;
    LONG64 maxUsedSize;
    LONG64 initialSize;
    LONG64 maxSize;
    enum STATE_COLOR dispstatus;
};

class ArrayOfHeapInfo
{
    HeapInfo *__ptr;
    int __size;
};

Returns a list of JAVA VM heap information.

2.4 ICM Specific Methods

ICMGetThreadList(ArrayOfICMThread *thread)

class ICMThread
{
    char *name;
    char *id;
    LONG64 requests;
    char *status;
    char *requesttype;
};

class ArrayOfICMThread
{
    ICMThread *__ptr;
    int __size;
};

Returns a list of threads used by ICM.

ICMGetConnectionList(ArrayOfICMConnection *connection)

class ICMConnection
{
    char *conid;
    char *protocol;
    char *role;
    char *requesttype;
    char *peer_address;
    int peer_port;
    char *local_address;
    int local_port;
    int proc_timeout;
    int keepalive_timeout;
    char *connection_time;
    int nihdl;
};
class ArrayOfICMConnection
{
    ICMConnection *__ptr;
    int __size;
};

Returns a list of incoming network connections handled by ICM.

ICMGetCacheEntries(ArrayOfICMCacheEntry *entry)

class ICMCacheEntry
{
    char *name;
    int version;
    LONG64 size;
    bool valid;
    char *cache;
    char *creation_time;
    char *last_access_time;
    char *expiration_time;
    char *cacheurl;
};

class ArrayOfICMCacheEntry
{
    ICMCacheEntry *__ptr;
    int __size;
};

Returns a list of objects cached by ICM. This list contains entries of all ICM caches, “cache” identifies the
    cache name. “cacheurl” can be used to read the object directly from the cache.

ICMGetProxyConnectionList(ArrayOfICMProxyConnection *connection)

class ICMProxyConnection
{
    char *conid;
    char *role;
    char *peer_address;
    int peer_port;
    char *local_address;
    int local_port;
    char *status;
    int nihdl;
    int socket;
    char *partner;

    char *internal_convid 0:1;
    char *external_convid 0:1;
    int *snc_context_length 0:1;
    char *failover_status 0:1;
    char *disconnect_time 0:1;
    char *objectid 0:1;
    char *tid_uid_mode 0:1;
};
class ArrayOfICMProxyConnection
{
    ICMProxyConnection *__ptr;
    int __size;
};

Returns a list of outgoing network proxy connections handled by ICM. The list contains JCo and VM container proxy connections. “partner” identifies the actual connection type. The remaining fields are only set for the matching connection type.

3 Error Handling

The interface is using SOAP Faults and HTTP error handling. General failures are reported via “HTTP/1.1 500 Internal Server Error” and SOAP Fault, <faultstring> contains the error details, e.g.:

    <SOAP-ENV:Fault>
        <faultcode>SOAP-ENV:Server</faultcode>
        <faultstring>DpIPCInit failed</faultstring>
    </SOAP-ENV:Fault>

Missing user credentials are reported via “HTTP/1.1 401 Unauthorized” error code and SOAP Fault:

    <SOAP-ENV:Fault>
        <faultcode>SOAP-ENV:Client</faultcode>
        <faultstring>HTTP Error: 'Unauthorized'</faultstring>
    </SOAP-ENV:Fault>

Invalid user credentials are reported via HTTP/1.1 500 Internal Server Error” and SOAP Fault:

    <SOAP-ENV:Fault>
        <faultcode>SOAP-ENV:Server</faultcode>
        <faultstring>Invalid Credentials</faultstring>
    </SOAP-ENV:Fault>

Insufficient user privileges are reported via HTTP/1.1 500 Internal Server Error” and SOAP Fault:

    <SOAP-ENV:Fault>
        <faultcode>SOAP-ENV:Server</faultcode>
        <faultstring>Permission denied</faultstring>
    </SOAP-ENV:Fault>
4 Web Service Clients

SAP offers 3 graphical user interfaces using of the SAPControl Web service interface:

- SAP Microsoft Management Console SnapIn (SAP MMC, Microsoft platforms only)
- SAP Java Management Console (SAP MC)
- SAP NetWeaver Administrator

In addition a command line client "sapcontrol" is available, offering easy access to all webmethods:

usage: sapcontrol

-prot GSOAP_HTTP | HTTP using gsoap build in sockets (default)
| NI_HTTP | HTTP using SAP NI sockets (prefer Unix domain sockets)
| NI_HTTPS | HTTPS using SAP NI sockets (prefer Unix domain sockets)
| WINHTTP | HTTP using Windows winhttp
| WINHTTPS | HTTPS using Windows winhttp
| PIPE | Windows named pipes

[-trace <filename>] Trace SOAP request/response
[-user <user> <password>] OS user and password for Web service authentication
[-repeat <N> <D>] Repeat Webmethod call <N> times (-1=forever) with <D> sec delay
[-format list] List output format (default)
| Script] Script output format
[-host <hostname>] Host to connect to (default: localhost)
- nr <instance nr.> SAP Instance number to connect to
- function <Webmethod> [parameter list]

Exitcode: 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

Security: Trusted connects without user and password check are possible through Unix domain socket or Windows named pipes.
Protected webmethods like Start or Stop require a trusted connection or OS user and password authentication.

Webmethods:

Start
Stop [softtimeout sec]
Shutdown
RestartInstance [softtimeout sec]
StopService
StartService <SID>
RestartService
ParameterValue [parameter]
GetStartProfile
GetTraceFile
GetAlertTree
GetAlerts
GetEnvironment
GetVersionInfo
GetQueueStatistic
GetProcessList
getInstanceProperties
ListDeveloperTraces
ReadDeveloperTrace <filename> <filesize>
ListLogFiles
ReadLogFile <filename> [<filter> [<language> [<maxentries> [<cookie>]]]]
 AnalyseLogFiles [<severity 0-2>] [<maxentries>] [<starttime YYYY MM DD HH:MM:SS> <endtime YYYY MM DD HH:MM:SS>]
GetAccessPointList
OExecute <command> <async> <timeout> <protocolfile>
SendSignal <pid> <signal>
GetSystemInstanceList
StartSystem [ALL|SCS|DIALOG|ABAP|J2EE |TREX|ENQREP |LEVEL <level> [<waittimeout sec>]]
StopSystem [ALL|SCS|DIALOG|ABAP|J2EE |TREX|ENQREP |LEVEL <level> [ [<waittimeout sec>] [softtimeout sec]]
RestartSystem [ALL|SCS|DIALOG|ABAP|J2EE |TREX|ENQREP |LEVEL <level> [ [<waittimeout sec>] [softtimeout sec]]
AccessCheck <function>
GetSecNetworkId <service_ip> <service_port> [<version> [<challenge>]]
GetNetworkId <service_ip> <service_port> [<version>]
ABAPReadSyslog
ABAPReadRawSyslog
ABAPGetWPTable
J2EEControlProcess <processname> <function>
J2EEGetProcessList
J2EEGetProcessList2
J2EEGetThreadList
J2EEGetThreadList2
J2EEGetThreadCallStack [<threadindex>]
J2EEGetThreadTaskStack [<threadindex>]
J2EEGetSessionList
J2EEGetCacheStatistic
J2EEGetCacheStatistic2
J2EEGetApplicationAliasList
J2EEGetComponentList
J2EEGetWebSessionList
J2EEGetEJBSessionList
J2EEGetRemoteObjectList
J2EEGetVMGCHistory
J2EEGetVMGCHistory2
J2EEGetVMHeapInfo
J2EEGetClusterMsgList
J2EEGetSharedTableInfo
ICMGetThreadList
ICMGetConnectionList
ICMGetProxyConnectionList
ICMGetCacheEntries
EnqGetStatistic
EnqGetLockTable
StartWait <timeout sec> <delay sec>
StopWait <timeout sec> <delay sec>
WaitforStarted <timeout sec> <delay sec>
WaitforStopped <timeout sec> <delay sec>
RestartServiceWait <timeout sec> <delay sec>
CheckHostAgent

5 Web Server Functionality

In addition to offering Web service functionality sapstartsrv acts as a very simple Web server on the same TCP/IP ports. A web browser can download files by using HTTP get from sapstartsrv. The root directory of the Web server is $(DIR_EXECUTABLE)/servicehttp. If no path is given in the request, the client will be redirected to $(DIR_EXECUTABLE)/servicehttp/sapmc/sapmc.html which is intended to download a SAP Java management client.

6 Logfiles

- Major problems are reported in the Windows application eventlog and Unix Syslog.
- SAP developer traces are directed to $(DIR_HOME)/sapstartsrv.log, default trace level is 0 (see service/trace profile parameter)!
- SAP instance start/stop is traced in $(DIR_HOME)/sapstart.log, stdout/stderr of started programs is redirected to stderr0-N.
- SLD registration is traced in $(DIR/Home)/dev_sldregs
- LDAP registration is traced in $(DIR_HOME)/dev_ldaps

7 Profile Parameters

- Autostart: 1: auto start of SAP instance during service start 0: no auto start (default)
- service/trace: 0-3: trace level for sapstartsrv.log (default 0)
- service/protectedwebmethods: Blank separated list of protected webmethods
- service/max_dia_queue_time: ABAP dialog queue time threshold (in sec) for yellow rating (default 5)
- service/j2eethreadtasktime/yellow: thread task time threshold (in sec) for yellow rating (default: 10)
- service/j2eethreadtasktime/red: thread task time threshold (in sec) for yellow rating (default: 20)
- service/j2eeccachehitratio/yellow: 0-100: cache hit ratio threshold (percentage) for yellow rating (default: 80)
- service/j2eeccachehitratio/red: 0-100: cache hit ratio threshold (percentage) for red rating (default: 90)
- service/j2eemheapusage/yellow: 0-100: java vm heap usage ratio threshold (percentage) for red rating (default: 80)
- service/j2eemheapusage/red: 0-100: java vm heap usage ratio threshold (percentage) for red rating (default: 90)
service/startpriority: Instance start priority (default: calculated from instance type, e.g. “1”: SCS, “2”: ABAP +Enque WP, “3”: Other)

service/hostname: Hostname or IP address for binding the Web service interface

service/halib: HA shared library to load for controlling clustered instances (default: Windows: sapNThalib.dll, Unix: None)

ldap/autoregister: 1: register service in LDAP directory during service start 0: no registration (default)

8 C# Sample Client Using the SAPControl Interface

The following section describes how to create a small self written sample application using the Web service interface. The description assumes Microsoft Visual .NET 2005 and a local SAP system using instance number 61 to be installed.

- Use “File->New->Project…” to crate a new project. Select “Other Languages->Visual C#” project type and the “Console Application” template. Enter a project name and press “OK” to create the project.
- Open the solution explorer using “View->Solution Explorer”. Select the “References” node in the solution explorer tree and choose context menu “Add Web References…”. Enter the SAPControl WSDL URL http://localhost:56013/?wsdl and press “Go”. The documentation of SAPControl will be displayed with all Web service methods. Select “Add Reference”. Visual Studio now generates a Web service proxy using namespace “localhost”.
- Modify the empty “Main” function of the template code like this:

```csharp
static void Main(string[] args)
{
    // Create a proxy object for the SAPControl interface
    localhost.SAPControl myservice = new localhost.SAPControl();

    // Declare an array of processes returned by GetProcessList()
    localhost.OSProcess[] osprocs;

    // Set the Url to connect to sapstartsrv of the SAP instance
    myservice.Url = "http://localhost:56113";

    // Get the list of running processes
    osprocs = myservice.GetProcessList();

    // Break into debugger to inspect ‘osprocs’ value
}
```

- Start the SAP instance if not yet done. Use “Debug->Start Debugging” to start the sample client in the Visual Studio debugger.
- The client will get the process list from sapstartsrv and break into the debugger afterwards. “osprocs” contains the same information as you can see in the MMC in the “Process List” node.

9 References

- SAP note 142100, Windows: Problems with new SAP service as of Rel. 4.5B
- SAP note 823941, SAP Start Service on Unix
- SAP note 877795, Problems with SAP start service sapstartsrv as of Release 7.00
- SAP note 927637, Web service authentication in sapstartsrv as of Release 7.00
- SAP note 936273, sapstartsrv for all platforms
- SAP note 995116, Backward porting of sapstartsrv for earlier releases
- [http://service.sap.com/~form/sapnet?_FRAME=CONTAINER&_OBJECT=011000358700008126612002E](http://service.sap.com/~form/sapnet?_FRAME=CONTAINER&_OBJECT=011000358700008126612002E), How to Use the ISAPControl Interface

### 10 Interface Version History

<table>
<thead>
<tr>
<th>Function</th>
<th>6.40</th>
<th>7.00</th>
<th>7.10</th>
<th>7.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Stop</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Shutdown</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ParameterValue</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetProcessList</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetProcessList2</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetStartProfile</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetTraceFile</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetAlertTree</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetAlerts</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RestartService</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>StopService</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetEnvironment</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ListDeveloperTraces</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ListLogFile</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ReadDeveloperTrace</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ReadLogFile</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AnalyseLogFile</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RestartInstance</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SendSignal</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetVersionInfo</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetQueueStatistic</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetInstanceProperties</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OSEexecute</td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AnalyseLogFiles</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetAccessPointList</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetSystemInstanceList</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>StartSystem</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>-------------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>StopSystem</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RestartSystem</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AccessCheck</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetProcessParameter</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetProcessParameter</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShmDetach</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetNetworkId</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetSecConnectionId</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABAPReadSyslog</td>
<td>&gt;= PATCH 169</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ABAPReadRawSyslog</td>
<td>&gt;= PATCH 169</td>
<td>&gt;= PATCH 96</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ABAPGetWPTable</td>
<td>&gt;= PATCH 169</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ABAPAcknowledgeAlerts</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetProcessList</td>
<td></td>
<td>&gt;= PATCH 169</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>J2EEGetProcessList2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetThreadList</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetThreadList2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetSessionList</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetWebSessionList</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetCacheStatistic</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetCacheStatistic2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetApplicationAliasList</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>J2EEGetComponentList</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetVMCHistory</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetVMCHistory2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetVMHeapInfo</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetEJBSessionList</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetRemoteObjectList</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetClusterMagList</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetSharedTableInfo</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEEnableDbgSession</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEDisableDbgSession</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetThreadCallStack</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EEGetTaskStack</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICMGetThreadList</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICMGetConnectionList</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICMGetCacheEntries</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICMGetProxyConnectionList</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICMGetProxyConnectionList</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EnqGetLockTable</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EnqRemoveLocks</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EnqGetStatistic</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11 Index

ABAPAcknowledgeAlerts, 18
ABAPGetWPTable, 17
ABAPReadRawSyslog, 17
ABAPReadSyslog, 16
AccessCheck, 14
AnalyseLogFiles, 13
GetAccessPointList, 13
GetAlerts, 9
GetAlertTree, 8
GetEnvironment, 8
GetInstanceProperties, 11
GetNetworkId, 16
GetProcessList, 6
GetProcessParameter, 14
GetQueueStatistic, 10
GetSecNetworkId, 16
GetStartProfile, 7
GetSystemInstanceList, 14
GetTraceFile, 7
GetVersionInfo, 10
ICMGetCacheEntries, 29
ICMGetConnectionList, 28
ICMGetProxyConnectionList, 29
ICMGetThreadList, 28
J2EEControlProcess, 19
J2EEDisableDbgSession, 26
J2EEEnableDbgSession, 25
J2EEGetApplicationAliasList, 23
J2EEGetCacheStatistic, 21
J2EEGetCacheStatistic2, 22
J2EEGetClusterMagList, 25
J2EEGetComponentList, 23
J2EEGetEJBSessionList, 24
J2EEGetProcessList, 18
J2EEGetProcessList2, 19
J2EEGetRemoteObjectList, 24
J2EEGetSessionList, 21
J2EEGetSharedTableInfo, 25
J2EEGetThreadCallStack, 26
J2EEGetThreadList, 19
J2EEGetThreadList2, 20
J2EEGetThreadTaskStack, 26
J2EEGetVMGCHistory, 26
J2EEGetVMGCHistory2, 27
J2EEGetVMHeapInfo, 27
ListDeveloperTraces, 7
ListLogFiles, 13
OSExecute, 10
ParameterValue, 6
ReadDeveloperTrace, 8
ReadLogFile, 12
RestartInstance, 5
RestartService, 6
RestartSystem, 7
SendSignal, 9
SetProcessParameter, 15
Shutdown, 5
Start, 5
StartSystem, 6
Stop, 5
StopService, 6
StopSystem, 6