# DiskBoss

## File & Disk Manager

# DiskBoss Server API

## Version 4.0

**Nov 2013**

Flexense Ltd.
www.flexense.com
info@flexense.com

# 1 DiskBoss Server SDK

## 1.1 Overview

DiskBoss is an automated, rule-based file and disk management solution allowing one to search and classify files, perform disk space utilization analysis, detect and remove duplicate files, organize files according to user-defined rules and policies, copy large amounts of files in a fault-tolerant way, synchronize disks and directories, cleanup wasted disk space and much more.

In addition to the desktop product versions, IT professionals and enterprises are provided with DiskBoss Server – a server-based product version, which runs in the background as a service and is capable of executing all types of disk analysis and file management operations in a fully automatic and unattended mode.

DiskBoss Server can be managed and configured locally or through the network using a free network client GUI application, the DiskBoss command line utility or the DiskBoss Server programming API, which provides the user with the ability to integrate DiskBoss' features and capabilities into other products and solutions.

The DiskBoss Server programming API allows one to connect to one or more DiskBoss Servers, define, configure and execute all types of disk analysis and file management operations, schedule periodic commands and save reports to a number of standard formats or export results to an SQL database.

## 1.2 SDK Components

The DiskBoss Server SDK is bundled with the standard DiskBoss Server and DiskBoss Network installations and it is located in the '**<Product Dir>\sdk**' directory. The SDK includes a C/C++ DLL library allowing one to control one or more DiskBoss Servers, a C/C++ include file defining the DiskBoss Server programming API and a number of example applications (including the full source code) showing how to use all major functions and capabilities of the DiskBoss Server programming API.

## 1.3 DiskBoss XML Format

The DiskBoss XML format is an open XML-Based format allowing one to specify different types of disk analysis and file management operations to be executed by DiskBoss. The DiskBoss Server SDK makes a heavy use of the DiskBoss XML format requiring all types of disk analysis and file management commands to be defined in the XML format. The DiskBoss XML format specification is available here: http://www.diskboss.com/diskboss_xml_format.pdf

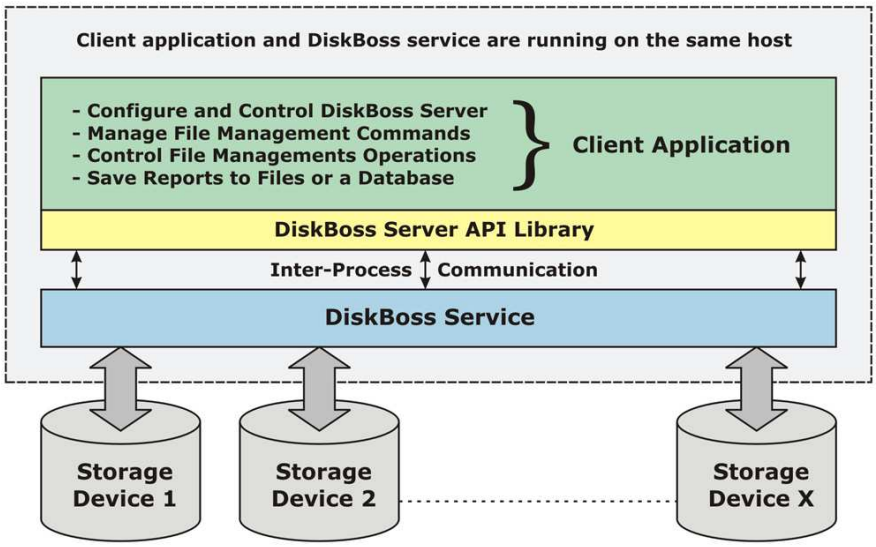
## 1.4 SDK and API License Terms

The DiskBoss Server API DLL library, C/C++ include files and source code examples can be freely redistributed to any 3rd parties, bundled with any software products and integrated into any kinds of software/hardware IT solutions without any limitations. Each instance of DiskBoss Server installed on a physical or virtual host computer and controlled by the DiskBoss Server API should be registered with an individual license and there is no option to share a single license between multiple server installations or migrate a license from one server installation to another.

## 1.5 DiskBoss API Library Usage Models

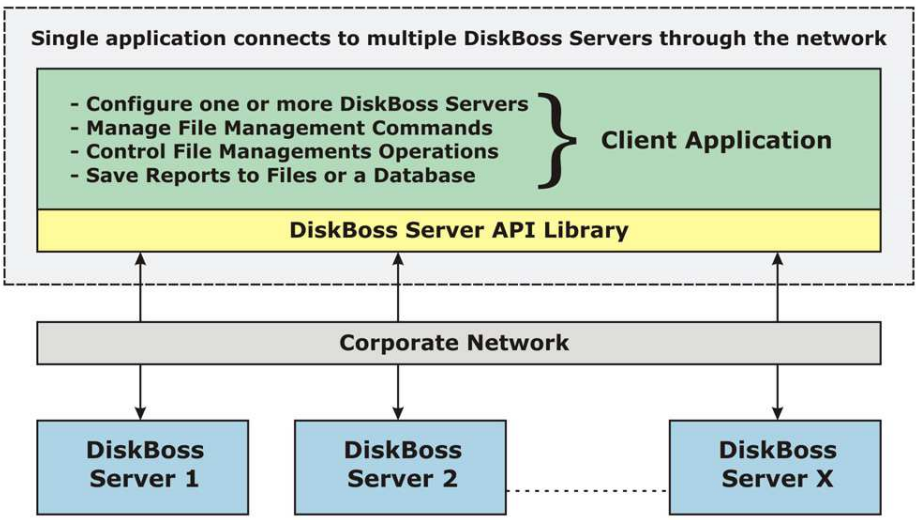
The DiskBoss Server API library allows one to control a single DiskBoss Server on the same host where the client application is running on or multiple DiskBoss servers running on a number of hosts connected to the same local network.

**Managing Multiple Storage Devices Using a Single DiskBoss Server**




In a single-host setup, the DiskBoss API library connects locally to the DiskBoss server, which runs in the background as a service. The DiskBoss API library provides the user with the ability to configure the server, setup user-defined file management commands, control file management operations and save results to file reports or an SQL database.

**Managing Multiple DiskBoss Servers Using a Single Client Application**




In a multi-host configuration, the DiskBoss API library connects to one or more DiskBoss Servers through the network allowing one to configure, manage and control multiple DiskBoss servers using a single client application.

# 2 DiskBoss Server API Reference

## 2.1 SCA_DiskBossInit

This function initializes the DiskBoss API library. The function should be called once before any other function calls.

**Prototype:**

ULONG **SCA_DiskBossInit**(ULONG Mode);

**Parameters:**

- **Mode** - DiskBoss API init mode, reserved, must be SCA_DISKBOSS_DEFAULT

**Returns:**

- **SCA_DISKBOSS_OK** - DiskBoss API init successful
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error

## 2.2 SCA_DiskBossFree

This function releases the DiskBoss API library. The function releases all allocated resources and disconnects from all connected DiskBoss Servers destroying all previously allocated server IDs. This function affects the client process only and all DiskBoss servers that were controlled by the client process will continue to operate and execute previously started file management operations. In order to stop previously started file management operations, the client process should explicitly stop these commands using the SCA_DiskBossStop or SCA_DiskBossStopAll API functions.

**Prototype:**

ULONG **SCA_DiskBossFree**(void);

**Parameters:**

- **None**

**Returns:**

- **SCA_DISKBOSS_OK** - DiskBoss API library released
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error

## 2.3 SCA_DiskBossSetErrorCallback

This function sets a custom error callback allowing one to intercept error messages issued by the DiskBoss API library. The function receives a pointer to the error callback function and a user data pointer, which may be used to access user-specific objects from the error callback function.

**Prototype:**

```
ULONG SCA_DiskBossSetErrorCallback(
                        SCA_DISKBOSS_ERROR_CALLBACK ErrorCallback,
                        void *UserData);
```

**Error Callback Function:**

```
typedef ULONG (*SCA_DISKBOSS_ERROR_CALLBACK)(
                        SCA_DISKBOSS_ERROR_INFO *ErrorInfo,
                        void *UserData);
```

**Error Info Structure:**

```
typedef struct tagSCA_DISKBOSS_ERROR_INFO {
        char            Message[SCA_STRING_LENGTH];
        char            Date[SCA_STRING_LENGTH];
        char            Time[SCA_STRING_LENGTH];
}SCA_DISKBOSS_ERROR_INFO;
```

**Parameters:**

- **ErrorCallback** - Error callback function for DiskBoss API library to call when an error occur. May be set to NULL if the user needs to disable a previously set error callback.
- **UserData** - an optional pointer to a user data object, which will be passed in the specified error callback with each error message.

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized

## 2.4 SCA_DiskBossConnect

This function connects the API library to a DiskBoss server locally or through the network. The server should be running and the port should be open in the host's firewall. The function returns a server ID, which will be later used to communicate with the server. Multiple DiskBoss servers may be connected and controlled simultaneously.

**Prototype:**

```
ULONG SCA_DiskBossConnect(  const char *HostName,
                            unsigned short Port,
                            ULONG TimeOut,
                            ULONG *pServerId);
```

**Parameters:**

- **HostName** - the name or the IP address of the host to connect to
- **Port** - the port number (default 8094) to connect to
- **TimeOut** - the connect timeout in milliseconds
- **ServerId** - a pointer to a variable to receive the server ID

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_TIMEOUT** - Connect timeout
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error

## 2.5 SCA_DiskBossDisconnect

This function disconnects from the specified DiskBoss Server. The function receives the server ID to disconnect from. The function just closes the current communication session and does not affect previously started file management operations.

**Prototype:**

```
ULONG SCA_DiskBossDisconnect( ULONG ServerId);
```

**Parameters:**

- **ServerId** - The ID of the server to disconnect from

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid

## 2.6 SCA_DiskBossServerLogin

This function logs into the DiskBoss server using the specified user name and password. The authentication procedure activates the API session and it must be performed before any other function calls for each specific session with the DiskBoss Server.

**Prototype:**

```
ULONG SCA_DiskBossServerLogin(  ULONG ServerId,
                                const char *UserName,
                                const char *Password);
```

**Parameters:**

- **ServerId** - The ID of the server to login to
- **UserName** - The DiskBoss Server user name (default 'diskboss')
- **Password** - The DiskBoss Server password (default 'diskboss')

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error

## 2.7 SCA_DiskBossGetServerInfo

This function gets detailed information about the DiskBoss server. The function receives a server ID and returns detailed information about the server in the server info structure.

**Prototype:**

```
ULONG SCA_DiskBossGetServerInfo( ULONG ServerId,
                        SCA_DISKBOSS_SERVER_INFO *pServerInfo);


typedef struct tagSCA_DISKBOSS_SERVER_INFO {
        char                HostName[SCA_STRING_LENGTH];
        char                Version[SCA_STRING_LENGTH];
        ULONG               Status;
        ULONG               TotalCommands;
        ULONG               RunningCommands;
        ULONG               CompletedCommands;
        ULONG               FailedCommands;
}SCA_DISKBOSS_SERVER_INFO;
```

**Parameters:**

- **ServerId** - The ID of the DiskBoss Server to inquire information from
- **pServerInfo** - A pointer to the server info structure to return information

**Server Info Structure:**

- **HostName** - The name or the IP address of the host (as used in connect)
- **Version** - The version of the DiskBoss Server
- **Status** - The status of the DiskBoss Server (default SCA_DISKBOSS_OK)
- **TotalCommands** - The total number of commands defined in the server
- **RunningCommands** - The number of running commands
- **CompletedCommands** - The number of completed commands
- **FailedCommands** - The number of failed commands

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error

## 2.8 SCA_DiskBossGetParam

This function gets a configuration parameter from the server. The function receives a server ID, the type of the parameter to inquire and returns the current parameter value in the supplied memory buffer in the form of an ASCII text string.

**Prototype:**

```
ULONG SCA_DiskBossGetParam( ULONG ServerId,
                            ULONG ParamType,
                            char *Buffer,
                            ULONG Length);
```

**Parameters:**

- **ServerId** - The ID of the server to inquire the parameter from
- **ParamType** - The type of the parameter to inquire
- **Buffer** - A buffer to receive the parameter value in
- **Length** - The length of the buffer

**Parameter Types:**

- **SCA_DISKBOSS_CFG_REPORT_TITLE** - Current report title
- **SCA_DISKBOSS_CFG_REPORT_LABEL** - Current report label
- **SCA_DISKBOSS_CFG_REPORT_HOST_NAME** - Current report host name
- **SCA_DISKBOSS_CFG_REPORT_LEVELS** - Get the maximum number of levels to export
- **SCA_DISKBOSS_CFG_REPORT_FILES_PER_CLASS** - Get the maximum number of files per class to export
- **SCA_DISKBOSS_CFG_REPORT_FILES_PER_DIR** - Get the maximum number of files per directory to export

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error

## 2.9 SCA_DiskBossSetParam

This function sets a server configuration parameter. The function receives a server ID, the type of the parameter to set and the parameter value in the form of an ASCII string.

**Prototype:**

```
ULONG SCA_DiskBossSetParam( ULONG ServerId,
                            ULONG ParamType,
                            const char *Value);
```

**Parameters:**

- **ServerId** - The ID of the server to set the parameter on
- **ParamType** - The type of the parameter to set
- **Value** - The parameter value in the form of an ASCII string

**Parameter Types:**

- **SCA_DISKBOSS_CFG_REPORT_TITLE** - Set a custom report title
- **SCA_DISKBOSS_CFG_REPORT_LABEL** - Set a custom report label
- **SCA_DISKBOSS_CFG_REPORT_HOST_NAME** - Set a custom report host name
- **SCA_DISKBOSS_CFG_REPORT_LEVELS** - Set the maximum number of levels to export
- **SCA_DISKBOSS_CFG_REPORT_FILES_PER_CLASS** - Set the maximum number of files per class to export
- **SCA_DISKBOSS_CFG_REPORT_FILES_PER_DIR** - Set the maximum number of files per directory to export

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error

## 2.10 SCA_DiskBossGetConfig

This function gets configuration from the DiskBoss server and saves it to the specified XML file. The function receives a server ID and the name of the XML file to save the server configuration to.

**Prototype:**

ULONG **SCA_DiskBossGetConfig**( ULONG ServerId,
const char *XmlFileName);

**Parameters:**

- **ServerId** - The ID of the server to get the configuration from
- **XmlFileName** - The name of the XML file to save the configuration to

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error

## 2.11 SCA_DiskBossSetConfig

This function sets the DiskBoss server configuration from the specified XML file. The function reads the XML file, sends the specified configuration options to the DiskBoss server and updates the server's configuration file.

**Prototype:**

ULONG **SCA_DiskBossSetConfig**( ULONG ServerId,
const char *XmlFileName);

**Parameters:**

- **ServerId** - The ID of the server to set the configuration to
- **XmlFileName** - The name of the XML file to load the configuration from

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error

## 2.12 SCA_DiskBossAddCommand

This function loads a file management command from the specified XML file and adds it to the server. The function receives a server ID and the name of the XML file containing the file management command to add. If another command with the same command name already defined in the server, the function will fail with an appropriate error code.

**Prototype:**

ULONG **SCA_DiskBossAddCommand**( ULONG ServerId,
const char *XmlFileName);

**Parameters:**

- **ServerId** - The ID of the server to add the command to
- **XmlFileName** - The name of the XML file to load the command from

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_ALREADY_EXISTS** - The command already exists

## 2.13 SCA_DiskBossUpdateCommand

This function updates an existing file management command from the specified XML file. The function receives a server ID and the name of the XML file containing the file management command to update. If the command does not exist, the function will fail with an appropriate error code.

**Prototype:**

ULONG **SCA_DiskBossUpdateCommand**( ULONG ServerId,
const char *XmlFileName);

**Parameters:**

- **ServerId** - The ID of the server to update the command on
- **XmlFileName** - The name of the XML file to load the command from

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.14 SCA_DiskBossGetCommand

This function gets the specified file management command from the DiskBoss server and saves it to an XML file. The function receives a server ID, the command name and the name of the XML file to save the command to.

**Prototype:**

```
ULONG SCA_DiskBossGetCommand(   ULONG ServerId,
                                const char *CommandName,
                                const char *XmlFileName);
```

**Parameters:**

- **ServerId** - The ID of the server to get the command from
- **CommandName** - The name of the file management command to get
- **XmlFileName** - The name of the XML file to save the command to

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist
- **SCA_DISKBOSS_ERR_FILE_IO** - Error saving the XML file

## 2.15 SCA_DiskBossGetCommandInfo

This function gets extended info about a file management command by the command name. The function receives a server ID, the name of the command and returns extended information about the file management command in the command info structure.

**Prototype:**

```
ULONG SCA_DiskBossGetCommandInfo(       ULONG ServerId,
                                        const char *CommandName,
                                SCA_DISKBOSS_COMMAND_INFO *pCommandInfo);
```

```
typedef struct tagSCA_DISKBOSS_COMMAND_INFO {
        char            Name[SCA_STRING_LENGTH];
        ULONG           Type;
        ULONG           Status;
        ULONG           Progress;
        ULONG           ErrorCount;
        ULONG           TotalFiles;
        SCA_UINT64      TotalSpace;
        ULONG           OutputFiles;
        SCA_UINT64      OutputSpace;
        char            NextStart[SCA_STRING_LENGTH];
}SCA_DISKBOSS_COMMAND_INFO;
```

**Parameters:**

- **ServerId** - The ID of the server to get the command info from
- **CommandName** - The name of the command to get info for
- **pCommandInfo** - A pointer to the command info structure

**Command Info Structure:**

- **Name** - The name of the file management command
- **Type** - The type of the command (refer to the list of command types)
- **Status** - The command status (refer to the list of command statuses)
- **Progress** - The command progress (if supported by the command)
- **ErrorCount** - The number of non-critical errors
- **TotalFiles** - The total number of files processed by the command
- **TotalSpace** - The total amount of disk space in Bytes
- **OutputFiles** - The number of output files (operation specific)
- **OutputSpace** - The amount of output disk space (operation specific)
- **NextStart** - The next start time for scheduled commands

**Command Types:**

- **SCA_DISKBOSS_CMD_SEARCH** - File search command
- **SCA_DISKBOSS_CMD_ORGANIZE** - File organizing command
- **SCA_DISKBOSS_CMD_CLASSIFY** - File classification command
- **SCA_DISKBOSS_CMD_ANALYZE** - Disk space analysis command
- **SCA_DISKBOSS_CMD_DUPLICATES** - Duplicate files detection command
- **SCA_DISKBOSS_CMD_COPY** - File copy command
- **SCA_DISKBOSS_CMD_MOVE** - File move command
- **SCA_DISKBOSS_CMD_DELETE** - File delete command
- **SCA_DISKBOSS_CMD_SYNC** - File synchronization command
- **SCA_DISKBOSS_CMD_MONITOR** - Disk change monitoring command

**Command Statuses:**

- **SCA_DISKBOSS_CMD_IDLE** - The command is idle
- **SCA_DISKBOSS_CMD_PENDING** - The command is start pending
- **SCA_DISKBOSS_CMD_RUNNING** - The command is running
- **SCA_DISKBOSS_CMD_PAUSED** - The command is paused
- **SCA_DISKBOSS_CMD_COMPLETED** - The command completed
- **SCA_DISKBOSS_CMD_CANCELED** - The command canceled by the user
- **SCA_DISKBOSS_CMD_FAILED** - The command failed

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.16 SCA_DiskBossGetCommandInfoByIndex

This function gets extended info about a file management command by the command index. The function receives a server ID, the command index and returns extended information about the file management command in the command info structure. This function is intended to be used in conjunction with the SCA_DiskBossGetServerInfo API function, which returns the total number commands defined in a server, to discover, list and manage file management commands defined in the DiskBoss server.

**Prototype:**

```
ULONG SCA_DiskBossGetCommandInfoByIndex( ULONG ServerId,
                                         ULONG CommandIndex,
                                         SCA_DISKBOSS_COMMAND_INFO *pCommandInfo);
```

For detailed info about the command info structure, fields descriptions and return results refer to the **SCA_DiskBossGetCommandInfo** function.

## 2.17 SCA_DiskBossCopyCommand

This function copies the specified file management command. The function receives the name of the file management command to copy and the name of the command copy. If the new command name is already in use, the function will fail with an appropriate error code.

**Prototype:**

```
ULONG SCA_DiskBossCopyCommand( ULONG ServerId,
                               const char *CommandName,
                               const char *NewCommandName);
```

**Parameters:**

**ServerId** - The ID of the server to copy the command on
**CommandName** - The name of the command to copy
**NewCommandName** - The name of the copy

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist
- **SCA_DISKBOSS_ERR_ALREADY_EXISTS** - The copy name is in use

## 2.18 SCA_DiskBossRenameCommand

This function renames the specified file management command. The function receives the name of the file management command to rename and a new command name. If the new command name is already in use, the function will fail with an appropriate error code.

**Prototype:**

```
ULONG SCA_DiskBossRenameCommand(    ULONG ServerId,
                                    const char *CommandName,
                                    const char *NewCommandName);
```

**Parameters:**

**ServerId** - The ID of the server to rename the command on
**CommandName** - The name of the command to rename
**NewCommandName** - The new command name

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.19 SCA_DiskBossDeleteCommand

This function deletes the specified file management command. The function receives a server ID and the name of the file management command to delete. If the command is running, it will be canceled and all results discarded.

**Prototype:**

```
ULONG SCA_DiskBossDeleteCommand(    ULONG ServerId,
                                    const char *CommandName);
```

**Parameters:**

**ServerId** - The ID of the server to delete the command from
**CommandName** - The name of the command to delete

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.20 SCA_DiskBossStartCommand

This function starts the specified file management command. The function receives a server ID, the start mode and the name of the file management command to start. If the start mode is set to 'Start-Now', the command will be started immediately. If the start mode is set to 'Start-Pending', the command will be submitted to the process queue and executed according to the submission order. If the specified command is already active, the function will fail with an appropriate error code. The function is asynchronous and it does not wait for the command to complete. In order to monitor the status of a previously started command, use the SCA_DiskBossGetCommandInfo API function.

**Prototype:**

```
ULONG SCA_DiskBossStartCommand( ULONG ServerId,
                                const char *CommandName,
                                ULONG Mode);
```

**Parameters:**

- **ServerId** - The ID of the server to start the command on
- **CommandName** - The name of the command to start
- **Mode** - The command start mode (refer to the list of start modes)

**Command Start Modes:**

- **SCA_DISKBOSS_CMD_START_PENDING** - Submit the command to the process queue. Commands submitted to the process queue are started and executed sequentially one after one by the DiskBoss task scheduler.
- **SCA_DISKBOSS_CMD_START_NOW** - Immediately start the command

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.21 SCA_DiskBossWaitCommand

This function waits of a previously started file management command to complete. The function receives a server ID, the command name to wait for, the timeout period and returns the command completion status. If the function reaches the specified timeout period, it returns an appropriate error code and the last command status.

**Prototype:**

```
ULONG SCA_DiskBossWaitCommand(  ULONG ServerId,
                                const char *CommandName,
                                ULONG TimeOut,
                                ULONG *pCommandStatus);
```

**Parameters:**

- **ServerId** - The ID of the server to wait for the command on
- **CommandName** - The name of the command to wait for
- **TimeOut** - The wait timeout in milliseconds
- **pCommandStatus** - A pointer to a variable to receive the last status

**Command Statuses:**

- **SCA_DISKBOSS_CMD_IDLE** - The command is idle
- **SCA_DISKBOSS_CMD_PENDING** - The command is start pending
- **SCA_DISKBOSS_CMD_RUNNING** - The command is running
- **SCA_DISKBOSS_CMD_PAUSED** - The command is paused
- **SCA_DISKBOSS_CMD_COMPLETED** - The command completed
- **SCA_DISKBOSS_CMD_CANCELED** - The command canceled by the user
- **SCA_DISKBOSS_CMD_FAILED** - The command failed

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.22 SCA_DiskBossPauseCommand

This function suspends a previously started file management command. The function receives a server ID and the name of the command to pause. If the command is not running, the function will fail and return an appropriate error code.

**Prototype:**

ULONG **SCA_DiskBossPauseCommand**( ULONG ServerId,
const char *CommandName);

**Parameters:**

**ServerId** - The ID of the server to pause the command on
**CommandName** - The name of the command to pause

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.23 SCA_DiskBossResumeCommand

This function resumes a previously suspended file management command. The function receives a server ID and the name of the command to resume. If the command is not paused, the function will fail and return an appropriate error code.

**Prototype:**

ULONG **SCA_DiskBossResumeCommand**( ULONG ServerId,
const char *CommandName);

**Parameters:**

**ServerId** - The ID of the server to resume the command on
**CommandName** - The name of the command to resume

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.24 SCA_DiskBossStopCommand

This function stops a previously started file management command. The function receives a server ID and the name of the command to stop. If the command is not active, the function will fail and return an appropriate error code.

**Prototype:**

ULONG **SCA_DiskBossStopCommand**( ULONG ServerId,
const char *CommandName);

**Parameters:**

**ServerId** - The ID of the server to stop the command on
**CommandName** - The name of the command to stop

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.25 SCA_DiskBossStopAll

This function stops all running and/or paused commands on the specified DiskBoss server. The function receives the ID of the DiskBoss server to stop commands on.

**Prototype:**

ULONG **SCA_DiskBossStopAll**( ULONG ServerId);

**Parameters:**

**ServerId** - The ID of the server to stop all running commands on

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error

## 2.26 SCA_DiskBossScheduleCommand

This function schedules a file management command to be executed periodically according to the specified time period. The function receives a server ID, the name of the command to schedule and the time period. Scheduled commands are automatically executed by the DiskBoss server according to the specified time period.

**Prototype:**

ULONG **SCA_DiskBossScheduleCommand**(ULONG ServerId,
const char *CommandName,
ULONG TimeUnits, ULONG TimeValue);

**Parameters:**

- **ServerId** - The ID of the server to schedule the command on
- **CommandName** - The name of the command to schedule
- **TimeUnits** - SCA_DISKBOSS_CMD_MINUTES or SCA_DISKBOSS_CMD_HOURS
- **TimeValue** - The time period value

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.27 SCA_DiskBossUnscheduleCommand

This function disables periodic execution of a previously scheduled file management command. The function receives a server ID and the command name to disable periodic execution for.

**Prototype:**

ULONG **SCA_DiskBossUnscheduleCommand**( ULONG ServerId,
const char *CommandName);

**Parameters:**

- **ServerId** - The ID of the server to unschedule the command on
- **CommandName** - The name of the command to unschedule

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist

## 2.28 SCA_DiskBossSaveReport

This function saves file classification, disk space analysis, duplicate files, file search or disk change monitoring results to a report file. The function receives a server ID, the command name to save the report for, a report file name and the report format, which may be set to HTML, XML, text, Excel CSV or FLR (DiskBoss' native report format).

**Prototype:**

```
ULONG SCA_DiskBossSaveReport( ULONG ServerId,
                              const char *CommandName,
                              const char *FileName,
                              ULONG Format);
```

**Parameters:**

- **ServerId** - The ID of the server to save the report from
- **CommandName** - The name of the command to save the report for
- **FileName** - The name of the report file
- **Format** - The report format (Refer to the list of report formats)

**Report Formats:**

- **SCA_DISKBOSS_FORMAT_HTML** - Saves an HTML report
- **SCA_DISKBOSS_FORMAT_TEXT** - Saves a text report
- **SCA_DISKBOSS_FORMAT_CSV** - Saves an Excel CSV report
- **SCA_DISKBOSS_FORMAT_XML** - Saves an XML report
- **SCA_DISKBOSS_FORMAT_PDF** - Saves a PDF report
- **SCA_DISKBOSS_FORMAT_FLR** - Saves a DiskBoss' native report

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist
- **SCA_DISKBOSS_ERR_FILE_IO** - Error saving the report file

## 2.29 SCA_DiskBossSaveToDatabase

This function saves file classification, disk space analysis, duplicate files, file search or disk change monitoring results to an SQL database. The function receives a server ID, the command name to save the report for and the ODBC data source, user name and password.

**Prototype:**

```
ULONG SCA_DiskBossSaveToDatabase( ULONG ServerId,
                                  const char *CommandName,
                                  const char *DataSource,
                                  const char *UserName,
                                  const char *Password);
```

**Parameters:**

- **ServerId** - The ID of the server to save results from
- **CommandName** - The name of the command to save results for
- **DataSource** - The name of the ODBC data source
- **UserName** - The ODBC user name
- **Password** - The ODBC Password

**Returns:**

- **SCA_DISKBOSS_OK** - Operation completed successfully
- **SCA_DISKBOSS_ERR_INTERNAL** - Internal error
- **SCA_DISKBOSS_ERR_NOT_INITIALIZED** - DiskBoss API not initialized
- **SCA_DISKBOSS_ERR_PARAMETER** - One or more parameters are invalid
- **SCA_DISKBOSS_ERR_NOT_CONNECTED** - DiskBoss Server is not connected
- **SCA_DISKBOSS_ERR_NOT_REGISTERED** - DiskBoss Server is not registered
- **SCA_DISKBOSS_ERR_NETWORK** - Unexpected network error
- **SCA_DISKBOSS_ERR_NOT_EXIST** - The command does not exist
- **SCA_DISKBOSS_ERR_SQL_CONNECT** - Cannot connect to the database
- **SCA_DISKBOSS_ERR_SQL_EXPORT** - SQL export error

# 3 DiskBoss Server SDK Examples

The DiskBoss Server SDK includes a number of source code examples showing how to use all major functions and capabilities of the DiskBoss file management server. The user is provided with a number of example applications (including the full source code) allowing one to configure the DiskBoss server, import, export, manage and execute file management commands and export results to a number of standard formats or an SQL database. The examples explained in this section are located in the '**<Product Dir>\sdk\examples**' directory. For each example, the user is provided with a C/C++ source file and a Visual Studio workspace file.

## 3.1 SRVSTATUS - Shows the status of the DiskBoss Server

This example shows how to connect to a DiskBoss server locally or through the network, login into the server with a user name and password, inquire the server's status information and display the server's status on the standard output. The example opens a simple communication session with the server, authenticates the user and performs a number of simple operations allowing one to learn basic features of the DiskBoss Server API.

## 3.2 SRVCONFIG - Configures the DiskBoss Server

This example application shows how to connect to a DiskBoss server, export the server's configuration to an XML file and then import the configuration back to the server. The major goal of the example is to show how to automate server's configuration using XML configuration files allowing one to upload user-specific configuration options to one or more DiskBoss servers using the DiskBoss Server API.

## 3.3 SRVCOMMANDS - Shows File Management Commands

This example shows how to display file management commands defined in one or more DiskBoss Servers. The example connects to a DiskBoss server locally or through the network, inquires extended information about file management commands defined in the server and shows the list of commands on the standard output. The major goal of the example is to show how to discover file management commands defined in a DiskBoss server using the DiskBoss Server API.

## 3.4 CMDIMPORT - Imports a File Management Command

This example shows how to import a DiskBoss file management command, defined in the DiskBoss XML format to a DiskBoss server locally or through the network. The example connects to the server and uploads a new file management command loaded from a local XML file to the server. The major goal of the example is to show how to create and configure file management commands defined in the DiskBoss Server using the DiskBoss Server API.

### 3.5 CMDEXPORT - Exports a File Management Command

This example application shows how to export a file management command defined in a DiskBoss server to a local XML file. The example connects to the DiskBoss server, downloads the specified file management command and saves it in a local XML file. The major purpose of the example is to show how to backup file management commands defined in a DiskBoss server using the DiskBoss Server API.

### 3.6 CMDEXECUTE - Executes a File Management Command

This example shows how to execute file management commands defined in a DiskBoss Server. The example connects to a DiskBoss Server locally or through the network, starts the specified file management command, waits for the command to complete and displays the command's completion status. The major goal of the example is to show how to execute, control and monitor file management commands using the DiskBoss Server API.

### 3.7 CMDDELETE - Deletes a File Management Command

This example application shows how to delete a file management command from a DiskBoss Server. The application connects to a DiskBoss server locally or through the network and deletes the specified file management command. The major goal of the example is to show how to manage file management commands using the DiskBoss Server API.

### 3.8 SAVEREPORT - Saves Results to Report Files

This example application shows how to export file classification, disk space analysis, duplicate files, file search or disk change monitoring results into HTML, XML, PDF, text or Excel CSV reports. The example application connects to a DiskBoss server, downloads results for the specified file management command and saves the results into a local file. The major goal of the example is to show how to access results for all types of disk analysis, search and monitoring operations.

### 3.9 SAVETOSQL - Saves Results to an SQL Database

This example application shows how to export file classification, disk space analysis, duplicate files, file search or disk change monitoring results to an SQL database. The example application connects to a DiskBoss server, downloads results for the specified file management command and saves the results into an SQL data using the ODBC database interface. The major goal of the example is to show how to export results for all types of disk analysis, search and monitoring operations to an SQL database.