IBM® Security Access Manager for Enterprise Single Sign-On
Version 8.2.2

Serial ID SPI Guide

IBM® Security Access Manager for Enterprise Single Sign-On
Version 8.2.2

Serial ID SPI Guide

Note

Before using this information and the product it supports, read the information in "Notices" on page 17.

Edition notice

Note: This edition applies to version 8.2.2 of IBM Security Access Manager for Enterprise Single Sign-On, (product number 5724–V67) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this publication

IBM Security Access Manager for Enterprise Single Sign-On Serial ID SPI Guide describes how to integrate any device with serial numbers and use it as a second authentication factor with AccessAgent.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see "Accessibility features" in the *IBM Security Access Manager for Enterprise Single Sign-On Planning and Deployment Guide*.

Technical training

For technical training information, see the following IBM Education website at http://www.ibm.com/software/tivoli/education.

Support information

IBM Support provides assistance with code-related problems and routine, short duration installation or usage questions. You can directly access the IBM® Software Support site at http://www.ibm.com/software/support/probsub.html.

IBM Security Access Manager for Enterprise Single Sign-On Troubleshooting and Support Guide provides details about:

- What information to collect before contacting IBM Support.
- The various methods for contacting IBM Support.
- How to use IBM Support Assistant.
- Instructions and problem-determination resources to isolate and fix the problem yourself.

Note: The **Community and Support** tab on the product information center can provide additional support resources.

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES

NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

Chapter 1. About Security Access Manager for Enterprise Single Sign-On Serial ID SPI

The Serial ID Service Provider Interface (SPI) provides a generic device management framework where AccessAgent can support any new device that contains serial numbers and use it as a second authentication factor.

Examples of these devices are:

- RFID cards
- Smart cards

Serial ID Service Provider Interface (SPI) defines interfaces that:

- Detect the presence of readers
- Initialize the reader
- Set or reset reader configuration
- Read the card serial number from the card presented

To integrate a designated RFID card, reader, or any device that has a serial ID, and use it as a second factor, the vendor must:

- Develop a Windows Dynamic-link library (DLL) that implements those defined interfaces.
- Create registry values that follow the Security Access Manager for Enterprise Single Sign-On Serial ID SPI format.

AccessAgent uses these registry values as parameters passed to the interface. The registry values provide information about the SPI DLL name, the vendor-defined name of supported devices, reader configuration, and others.

Vendors are provided with an RFID DSP tool to develop and test the Serial ID Device Service Provider for AccessAgent.

Chapter 2. Serial ID specification

The Device Service Provider (DSP) configuration includes how to load the Device Service Provider, what type of devices it supports, and what interfaces these devices expose.

The Device Service Provider configuration details are stored in the `HKLM\SOFTWARE\IBM\ISAM ESSO\SOCIAccess\DSPList` registry hive.

This registry hive also contains configuration parameters that AccessAgent reads and passes on to the Device Service Provider. The implementer of the Device Service Provider provides the configuration information.

See the following registry configuration structure:

```
HKLM\SOFTWARE\IBM\ISAM ESSO\SOCIAccess\DSPList
| - <dsp name>
   | - Filename:REG_SZ: [Name of the DSP DLL]
   | - Enabled:DWORD:[0|1]
   | - Devices
       | - <device name>
            | - <DeviceTypeId:REG_SZ:[Device Type Identifier]
            | - Interfaces
                  | - <interfaces_id>:REG_SZ:[GUID]
            | - Parameters
                  | - <param_name>:REG_SZ:<param_value>
            | - Trigger
                  | - WinInterfaceClass:REG_SZ:[GUID]
```

The following names are the different identifiers in the registry configuration structure. The identifiers that are enclosed in < and > are variable names. These variables can have multiple instances, depending on the registry structure. Other identifiers are registry constants.

Name	Description
Dsp_name	Vendor-defined name of the Device Service Provider.
Filename	Name of the Device Service Provider DLL. The Device Service Provider DLL specified in the registry must be placed in the AccessAgent installation folder. The other DLL files that it depends on must be stored outside of the AccessAgent installation folder so that it does not affect the AccessAgent upgrade.
Enabled	Registry value that controls whether the Device Service Provider is used by AccessAgent. This switch can be used for troubleshooting.
Devices	Registry node that contains all the devices that are supported by the Device Service Provider.
device_name	Vendor-defined friendly name of a supported device.
DeviceTypeId	Vendor-defined identifier that represents a class of devices.
Interfaces	Registry node that contains the list of interfaces that are supported by the device.
Interface_id	Identifier (GUID) of a supported interface as defined in the Device Service Provider Interface.

Name	Description
Parameters	Registry node that contains the list of name-value pairs that can be used to configure the device.
param_name	Name of a configuration parameter.
param_value	String value of the parameters.
Trigger	Registry node that contains the device trigger.
WinInterfaceClass	Device Trigger (GUID) of supported device.

Chapter 3. Serial ID SPI API

The IBM Security Access Manager for Enterprise Single Sign-On Serial ID Service Provider Interface application programming interface (API) specifies a set of functions that the Device Service Provider implements.

See the following topics for a list of Serial ID SPI APIs:
- "Constants"
- "Error Codes"
- "Generic Service Provider Interface functions"

Constants

Name	Value Description	Header
DEVICE_SPI_VERSION_MAJOR	1	Defines the major version of the current DeviceSPI
DEVICE_SPI_VERSION_MINOR	1	Defines the minor version of the current DeviceSPI
RFID_READER_ASYNC_INTERFACE_ID	{216DE8B9-FD09-44F3-A39D-B8A6F7A078D8}	Identifier for the event-based Serial ID SPI

Error Codes

Name	Description
E_DEV_SUCCESS	The operation is successfully completed.
E_DEV_FILE_NOT_FOUND	File cannot be found.
E_DEV_DEVICE_ERROR	There is an error with device.
E_DEV_DEVICE_REMOVED	The device is no longer reachable.
E_DEV_INSUFFICIENT_BUFFER	The memory that is passed to the interface is too small.
E_DEV_INVALID_ARGUMENT	Arguments are not correct.
E_DEV_NOT_INITIALIZED	Device is not initialized.
E_DEV_FAIL	Generic error.

Generic Service Provider Interface functions

Table 1. GetDeviceSpiVersion

Definition	`E_DEV_Error GetDeviceSpiVersion(OUT int* puMajorVersion, OUT int* puMinorVersion);`
Description	Returns the version of the Device SPI implemented by the provider.
Parameters	`puMajorVersion` - Pointer to the location that receives the major version of the Device Service Provider. `puMinorVersion` - Pointer to the location that receives the minor version of the Device Service Provider.

Table 1. GetDeviceSpiVersion (continued)

Return Values	- E_DEV_SUCCESS if successful. - E_DEV_INVALID_ARGUMENT if any of the arguments is NULL.

Table 2. Open

Definition	E_DEV_Error Open();
Description	Initializes the Serial ID Device Service Provider. All other function returns E_DEV_NOT_INITIALIZED, if this function is not successfully called first. Call on this function again on an already initialized device does not have an effect.
Parameters	None.
Return Values	- E_DEV_SUCCESS if successful. - E_DEV_FILE_NOT_FOUND if one of the dependencies of the Device Service Provider is not installed or available.

Table 3. Close

Definition	E_DEV_Error Close();
Description	Uninitializes the interface DLL and does clean up.
Parameters	None.
Return Values	- E_DEV_SUCCESS if successful.

Table 4. RefreshDeviceStatus

Definition	E_DEV_Error RefreshDeviceStatus(IN const char szDeviceTypeId[], OUT unsigned long* pulNumDevices);
Description	Scans the status of the connected devices and return the number of devices found. The Device Framework obtains the device name from the configuration information that is associated with the Device Service Provider. When the Device Framework receives a generic device arrival or removal notification from the Operating System, it calls this function for the Device Service Provider to refresh the status of its devices. This function might be called while a device is started with StartDevice(). Hence, it is important that RefreshDeviceStatus() does not affect the operation of an existing device.
Parameters	szDeviceTypeId - String identifier of one of the devices that are supported by the Device Service Provider. pulNumDevices - Number of devices that are currently attached to the system.
Return Values	- E_DEV_SUCCESS if successful. - E_DEV_INVALID_ARGUMENT if either the szDeviceTypeId or pulNumDevices is NULL or the szDeviceTypeId is not supported by the Device Service Provider.

Table 5. GetDeviceID

Definition	`E_DEV_Error GetDeviceID(IN const char szDeviceTypeId[],`
	`IN int iDeviceIndex,`
	`OUT char* szId,`
	`IN OUT unsigned long* pulIdeLen);`

Description	Returns the identifier of a device instance. The device instance is specified with a device type identifier and an index that is based on the number of devices return in the previous RefreshDeviceStatus call. If a device remains connected to the system, the Device Service Provider must return the same identifier every time this function is called. The device instance ID returned by a Device Service Provider must be unique across all the device type that it supports.
Parameters	`szDeviceTypeId` - `NULL` terminated string that identifies the device type. `iDeviceIndex` - Index of the device in the range 0..N-1, where N is the number of connected devices as returned in the previous `RefreshDeviceStatus()` call. `szId` - `NULL` terminated device instance identifier string. If this parameter is `NULL`, the Device Service Provider returns only the required size of the buffer in `pulIdLen` parameter. `pulIdLen` - Length of the `szId` buffer that is provided as the input. The Device Service Provider sets the number of characters (including the `NULL` character) of identifier as the output.
Return Values	• `E_DEV_SUCCESS` if successful. • `E_DEV_INVALID_ARGUMENT` if `pulIdLen` is `NULL` or if `iDeviceIndex` contains a value that is not valid. • `E_DEV_INSUFFICIENT_BUFFER` if either `szId` is `NOT NULL` and the size of the `szId` buffer as provided in `pulIdLen` is insufficient to hold the ID.

Table 6. SetDeviceParam

Definition	`E_DEV_Error SetDeviceParam(IN const char szDeviceId[],`
	`IN const char szKey[],`
	`IN const char szValue[]);`

Description	Sets a configurable parameter for the device. The Device Framework obtains this parameter from the configuration information that is associated with the Device Service Provider.
Parameters	`szDeviceId` - Identifier of the device that is obtained through a `GetDeviceID` function. `szKey` - Vendor-defined configuration key. `szValue` - Configuration value.
Return Values	• `E_DEV_SUCCESS` if successful. • `E_DEV_INVALID_ARGUMENT` if any of the input parameters are `NULL` or hold values that are not valid.

Table 7. StartDevice

Definition	`E_DEV_Error StartDevice(IN const char szDeviceId[]);`

Table 7. StartDevice (continued)

Description	Initializes the device. Calling `StartDevice()` on the device that is already started has no effect.
Parameters	`szDeviceId` - Identifier of the device that is obtained through a `GetDeviceID` function.
Return Values	- `E_DEV_SUCCESS` if successful. - `E_DEV_INVALID_ARGUMENT` if any of the input parameters are `NULL` or hold values that are not valid. - `E_DEV_DEVICE_REMOVED` if the device identified by `szDeviceId` is not currently attached to host. - `E_DEV_DEVICE_ERROR` if a generic device error occurs.

Table 8. StopDevice

Definition	`E_DEV_Error StopDevice(IN const char szDeviceId[]);`
Description	Uninitializes the device. The Device Framework might call this function even after the device identified by `szDeviceId` is removed from the host.
Parameters	`szDeviceId` - Identifier of the device that is obtained through a `GetDeviceID` function.
Return Values	- `E_DEV_SUCCESS` if successful. - `E_DEV_INVALID_ARGUMENT` if any of the input parameters are `NULL` or hold values that are not valid.

Table 9. PF_ID_NOTIFY

Definition	`typedef E_DEV_Error (*PF_ID_NOTIFY)(IN void *pVoid,`
	`IN const char szDeviceId[],`
	`IN const unsigned char* pucId,`
	`IN unsigned long ulIdLen);`

Description	Callback function that allows a Device Service Provider to notify AccessAgent that a serial ID is read.
Parameters	`pVoid [in]` - Callback parameter that is registered with the DSP. `szDeviceId [in]` - Identifier of the reader that received the UID data. `pucId [in]` - Buffer containing the UID of the card. `ulIdLen [in]` - Length of the UID buffer in bytes.
Return Values	- `E_DEV_SUCCESSFUL` if successful.

Table 10. RegisterReaderCallback

Definition	`E_DEV_Error RegisterReaderCallback(IN PF_ID_NOTIFY pfNotify,`
	`IN void* pVoid);`

Description	Registers the reader callback function with the Device Service Provider. This function can be called with `NULL` argument to clear the previously registered callback.
Parameters	`pfNotify` - Reader callback function. `pVoid` - Callback parameter. This value is passed to the callback function when it is called by the Device Service Provider.
Return Values	- `E_DEV_SUCCESS` if successful.

Table 11. PF_Log

Definition	`typedef void (*PF_Log)(IN void *pCallbackParam,`
	`E_DEV_LogLevel level,`
	`IN const char sFunctionName[],`
	`IN const char szDesc[]);`

Description	Callback function that allows a Device Service Provider to write log statement. This function is implemented by the application (through the AccessAgent Device Framework) and called by the Device Service Provider. The following log levels are defined: • `DEV_LOG_SEVERE - 1` • `DEV_LOG_BASEINFO - 2` • `DEV_LOG_MOREINFO - 3` • `DEV_LOG_DEBUG - 4`
Parameters	`pCallbackParam` - Callback parameter that is registered with the Device Service Provider. `level` - Log Level. `szFunctionName` - Function from with the log is being written. `szDesc` - Descriptive text.
Return Values	

Table 12. RegisterLogCallback

Definition	`E_DEV_Error RegisterLogCallback(IN PF_Log pfLog,`
	`IN void* pVoid);`

Description	Registers a logging callback function with the Device Service Provider. Through this function, the Device Framework allows a Device Service Provider to write an error or debug information in AccessAgent log files. This function can be called with NULL argument to clear the previously registered callback.
Parameters	`pfLog` - Logging callback function. `pVoid` - Callback parameter. This value is passed to the callback function when it is called by the Device Service Provider.
Return Values	• `E_DEV_SUCCESS` if successful.

Chapter 4. Implementing and testing the Serial ID Device Service Provider

Know how you can implement and test if the Serial ID Device Service Provider (DSP) is working properly.

Implementing a Serial ID Device Service Provider

Follow this procedure to implement the Serial ID Device Service Provider (DSP).

Procedure

1. Implement each of the Service Provider Interface functions.

   See "Generic Service Provider Interface functions" on page 5.
2. Use the RFID DSP tool to test the Device Service Provider.

   See "Testing Serial ID Device Service Provider."
3. Deploy the RFID Device Service Provider with AccessAgent.

Testing Serial ID Device Service Provider

Vendors can use the RFID DSP tool to verify or test the implementation of Serial ID Service Provider Interface.

Procedure

1. Create the registry configuration for the Device Service Provider.

   See Chapter 2, "Serial ID specification," on page 3.
2. Copy the Device Service Provider DLL to the same folder where the tool is located.
3. Run the tool and select a Device Service Provider from the list.

   The tool automatically loads the Device Service Provider and detects the attached devices that are recognized by the selected Device Service Provider. The attached device type IDs and device IDs are displayed.
4. Present the authentication factor to the reader, if applicable.

   The graphical user interface displays the device serial number and log statements of the Device Service Provider and the graphical user interface tool.

Verifying with AccessAgent

Verify with AccessAgent if the Serial ID Service Provider Interface is working properly.

Procedure

1. Copy the Serial ID Device Service Provider DLL to `<AccessAgent installation folder>/Config`.
2. Edit **DeploymentOptions.reg** to include the IBM Security Access Manager for Enterprise Single Sign-On Serial ID Device Service Provider registry configuration. See *IBM Security Access Manager for Enterprise Single Sign-On Configuration Guide*.
3. Install AccessAgent. See *IBM Security Access Manager for Enterprise Single Sign-On Installation Guide*.

4. Configure AccessAgent to use RFID as an authentication factor. See *IBM Security Access Manager for Enterprise Single Sign-On Configuration Guide*.
5. Use the second authentication factors that you added and see whether it works with the typical AccessAgent workflows.

Chapter 5. Device Service Provider Interface headers

The example code can be used for implementing and testing device Service Provider Interface (SPI) headers.

```
||822S1||    /*
||822S1||    * ============================================================================
||822S1||    * (c) Copyright IBM Corp. 2008, 2014
||822S1||    * ============================================================================
||822S1||    */
||822S1||
||822S1||    ///**
||822S1||    //* @file      DeviceSPI.h
||822S1||    //* @brief     Defines Device SPI
||822S1||    //* @version   1.2
||822S1||    //*/
||822S1||
||822S1||    ///**
||822S1||    //* Change History
||822S1||    //* [20081008]: From v1.0 to v1.1
||822S1||    //* 1. Removed GetRfidSpiVersion() and added GetDeviceSpiVersion(). Replaced the
||822S1||    corresponding version constants.
||822S1||    //* 2. Added the interface ID string for asynchronous RFID reader interface.
||822S1||    //* 3. Added const qualifier for parameters wherever applicable
||822S1||    //* 4. Added a void* parameter to the RFID reader callback function.
||822S1||    //* 5. Added utility functions for writing log statements.
||822S1||    //*
||822S1||    //* [20081024]: From v1.1 to v1.2
||822S1||    //* 1. Added pre-processor instructions to allow multiple includes of this file
||822S1||    //* 2. Changes to the E_DEV_LogLevel enum.
||822S1||    //* 3. Changes to GetDeviceID() function: Added szDeviceTypeId parameter.
||822S1||    //*/
||822S1||    #ifndef _DEVICESPI_H_
||822S1||    #define _DEVICESPI_H_
||822S1||
||822S1||    #ifndef IN
||822S1||    #define IN
||822S1||    #endif
||822S1||    #ifndef OUT
||822S1||    #define OUT
||822S1||    #endif
||822S1||
||822S1||    //Constants
||822S1||    #define DEVICE_SPI_VERSION_MAJOR 1
||822S1||    #define DEVICE_SPI_VERSION_MINOR 0
||822S1||
||822S1||    #ifndef DEVICE_SPI
||822S1||    #ifdef DEVICE_SPI_IMPORTS
||822S1||    #define DEVICE_SPI __declspec(dllimport)
||822S1||    #else
||822S1||    #define DEVICE_SPI __declspec(dllexport)
||822S1||    #endif
||822S1||    #endif
||822S1||
||822S1||    #ifdef __cplusplus
||822S1||    extern "C" {
||822S1||    #endif
||822S1||
||822S1||    //Error Codes
||822S1||    typedef enum
||822S1||    {
||822S1||     //SPI Error Codes------
||822S1||     //More Error codes to be defined in the future
||822S1||     //...
```

```
 E_DEV_FILE_NOT_FOUND = -7,
 E_DEV_DEVICE_ERROR = -6,
 E_DEV_DEVICE_REMOVED = -5,
 E_DEV_INSUFFICIENT_BUFFER = -4,
 E_DEV_INVALID_ARGUMENT = -3,
 E_DEV_NOT_INITIALIZED = -2,
 E_DEV_FAIL = -1,
 //---------------------

 //SPI Success codes----
 E_DEV_SUCCESS = 0,
 //More success codes to be defined in the future
 //...
 //---------------------
}E_DEV_Error;

//Utility functions: START -----------------------------------------------------
typedef enum
{
 DEVSPI_LOG_SEVERE = 1,
 DEVSPI_LOG_BASEINFO,
 DEVSPI_LOG_MOREINFO,
 DEVSPI_LOG_DEBUG
}E_DEV_LogLevel;

/**
 * \brief Callback function that allows a DSP to write log statements.
    This function is actually implemented by the application and invoked
by the DSP.
 * \param pCallbackParam [in] Callback parameter registered with the DSP.
 * \param level [in] Log level
 * \param szFunctionName [in] NULL-terminated function name
 * \param szDesc [in] NULL-terminated descriptive text
 */
typedef void (*PF_Log)(IN void* pCallbackParam, E_DEV_LogLevel level,
IN const char szFunctionName[], IN const char szDesc[]);


/**
 * \brief Register a logging callback function with the DSP.
   Through this function, the Device Framework allows a DSP
   to write error or debug information in AccessAgent log files.
   This function can be called with NULL argument to clear
   the previously registered callback.
 * \param pfLogCallback [in] Pointer to logging function
 * \param pVoid [in] Callback parameter.
 * \return E_DEV_SUCCESS if successful
 */
DEVICE_SPI E_DEV_Error RegisterLogCallback(IN PF_Log pfLogCallback,
IN void* pVoid);
//Utility functions: END -------------------------------------------------------

//Device Services Provider (DSP) Interface: START------------------------------

/**
 * \brief Returns the version of the Device SPI implemented by the DSP
 * \param puMajorVersion [out] The current major version is 1
 * \param puMinorVersion [out] The current minor version is 0
 * \return E_DEV_SUCCESSFUL if successful.
 *   E_DEV_INVALID_ARGUMENT if any of the arguments is NULL.
 */
DEVICE_SPI E_DEV_Error GetDeviceSpiVersion(OUT int* puMajorVersion, OUT
int* puMinorVersion);

/**
 * \brief Initializes the interface DLL. All other function may return
E_DEV_NOT_INITIALIZED, if this function
```

```
||822S1||           hasn't been successfully invoked.
||822S1||        * \return E_DEV_SUCCESS if successful
||822S1||        */
||822S1||       DEVICE_SPI E_DEV_Error Open();
||822S1||
||822S1||       /**
||822S1||        * \brief Uninitializes the interface DLL and performs cleans up.
||822S1||        * \return E_DEV_SUCCESS if successful
||822S1||        */
||822S1||       DEVICE_SPI E_DEV_Error Close();
||822S1||
||822S1||       /**
||822S1||        * \brief Scans the status of connected devices of the given type and
||822S1||       returns the number of devices found.
||822S1||        * \param szDeviceTypeId [in] NULL terminated string that identifies the
||822S1||       type of the device
||822S1||               e.g. the reader name "B-Net 91 07"
||822S1||        * \param ulNumDevices [out] Number of devices currently attached to the system
||822S1||        * \return E_DEV_SUCCESS if successful
||822S1||        * \remark A DSP may support devices of multiple types.
||822S1||        */
||822S1||       DEVICE_SPI E_DEV_Error RefreshDeviceStatus(IN const char szDeviceTypeId[],
||822S1||       OUT unsigned long* pulNumDevices);
||822S1||
||822S1||       /**
||822S1||        * \brief Returns the device instance identifier.
||822S1||        * \param szDeviceTypeId [in] NULL terminated string that identifies the type
||822S1||       of the device.
||822S1||        * \param iDeviceIndex [in] Index of the device in the range 0..N-1 where
||822S1||       N is the number of connected devices
||822S1||              as returned in the previous GetNumberOfConnectedDevices() call.
||822S1||        * \param szId [out] NULL terminated device identifier string. If this parameter
||822S1||       is NULL,
||822S1||             the DSP only returns the required size of the buffer in pulIdLen parameter.
||822S1||        * \param pulIdLen [inout] Length of the szId buffer provided as the input.
||822S1||              The DSP sets the number of characters (including the NULL character)
||822S1||       of identifier as the output.
||822S1||        * \return E_DEV_SUCCESS if successful
||822S1||        * \remark As long as a device remains connected to the system, a DSP must
||822S1||       return the same identifier every time this function is invoked.
||822S1||        */
||822S1||       DEVICE_SPI E_DEV_Error GetDeviceID(IN const char szDeviceTypeId[], IN int
||822S1||       iDeviceIndex, OUT char* szId, IN OUT unsigned long* pulIdLen);
||822S1||       //Device Services Provider Interface: END---------------------------------
||822S1||
||822S1||
||822S1||       //Generic Device Interface: START------------------------------------------
||822S1||       /**
||822S1||        * \brief Sets a configurable parameter for the device
||822S1||        * \param szDeviceId [in] Identifier of the device
||822S1||        * \param szKey [in] Configuration key
||822S1||        * \param szValue [in] Configuration value
||822S1||        * \return E_DEV_SUCCESS if successful
||822S1||        * \remark The device parameters should be set before calling StartDevice().
||822S1||          The configured parameters will take effect when StartDevice() is called next.
||822S1||        */
||822S1||       DEVICE_SPI E_DEV_Error SetDeviceParam(IN const char szDeviceId[], IN const
||822S1||       char szKey[], IN const char szValue[]);
||822S1||
||822S1||       /**
||822S1||        * \brief Initializes a device
||822S1||        * \param szDeviceId [in] Device Identifier
||822S1||        * \return E_DEV_SUCCESS if successful
||822S1||        */
||822S1||       DEVICE_SPI E_DEV_Error StartDevice(IN const char szDeviceId[]);
||822S1||
||822S1||       /**
```

```
 * \brief Uninitializes the device and performs clean up.
 * \param szDeviceId [in] Device Identifier
 * \return E_DEV_SUCCESS if successful
 * \remark A client may call this function even after the device identified
by szDeviceId is removed from the host.
 */
DEVICE_SPI E_DEV_Error StopDevice(IN const char szDeviceId[]);

//Generic Device Interface: END-------------------------------------------

//RFID Reader interface: START--------------------------------------------

//GUID representing the current asynchronous RFID reader interface
#define RFID_READER_ASYNC_INTERFACE_ID "{216DE8B9-FD09-44f3-A39D-B8A6F7A078D8}"

/**
 * \brief Notifies the client of the card UID read by the RFID Reader
 * \param pVoid [in] Callback parameter registered with the DSP.
 * \param szDeviceId [in] Identifier of the reader that received the UID data
 * \param pucId [in] Buffer containing the UID of the card
 * \param ulIdLen [in] Length of the UID buffer in bytes
 * \return E_DEV_SUCCESSFUL if succesful
 * \remark The application won't hold on to the ID buffer after callback
function
 *  returns. As such, a DSP may free the buffer pointed to by pucId after
the function returns.
 */
typedef E_DEV_Error (*PF_ID_NOTIFY)(IN void* pVoid, IN const char szDeviceId[],
IN const unsigned char* pucId, IN unsigned long ulIdLen);

/**
 * \brief Registers a identifier callback function with the DSP.
 * \param fNotify [in] Callback function
 * \param pVoid [in] Callback parameter. This value is passed to the callback
function when it is invoked by the DSP.
 * \return E_DEV_SUCCESS if successful
 */
DEVICE_SPI E_DEV_Error RegisterReaderCallback(IN PF_ID_NOTIFY fNotify,
IN void* pVoid);

//RFID Reader interface: END----------------------------------------------


#ifdef __cplusplus
}
#endif

#endif //#ifndef _DEVICESPI_H_
```

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law :

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information; at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Privacy Policy Considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering uses other technologies that collect each user's user name, password or other personally identifiable information for purposes of session management, authentication, single sign-on configuration or other usage tracking or functional purposes. These technologies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details/us/en sections entitled "Cookies, Web Beacons and Other Technologies" and "Software Products and Software-as-a Service".

Glossary

This glossary includes terms and definitions for IBM Security Access Manager for Enterprise Single Sign-On.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology (opens in new window).

A

account data
The logon information required to verify an authentication service. It can be the user name, password, and the authentication service which the logon information is stored.

account data bag
A data structure that holds user credentials in memory while single sign-on is performed on an application.

account data item
The user credentials required for logon.

account data item template
A template that defines the properties of an account data item.

account data template
A template that defines the format of account data to be stored for credentials captured using a specific AccessProfile.

action
In profiling, an act that can be performed in response to a trigger. For example, automatic filling of user name and password details as soon as a sign-on window displays.

Active Directory (AD)
A hierarchical directory service that enables centralized, secure management of an entire network, which is a central component of the Microsoft Windows platform.

Active Directory credential
The Active Directory user name and password.

Active Directory password synchronization
An IBM Security Access Manager for Enterprise Single Sign-On feature that synchronizes the ISAM ESSO password with the Active Directory password.

active radio frequency identification (active RFID)
A second authentication factor and presence detector. See also radio frequency identification.

active RFID
See active radio frequency identification.

AD
See Active Directory.

administrator
A person responsible for administrative tasks such as access authorization and content management. Administrators can also grant levels of authority to users.

API
See application programming interface.

application
A system that provides the user interface for reading or entering the authentication credentials.

application policy
A collection of policies and attributes governing access to applications.

application programming interface (API)
An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

audit
A process that logs the user, Administrator, and Helpdesk activities.

authentication factor
The device, biometrics, or secrets required as a credentials for validating digital identities. Examples of authentication

factors are passwords, smart card, RFID, biometrics, and one-time password tokens.

authentication service
A service that verifies the validity of an account; applications authenticate against their own user store or against a corporate directory.

authorization code
An alphanumeric code generated for administrative functions, such as password resets or two-factor authentication bypass.

auto-capture
A process that allows a system to collect and reuse user credentials for different applications. These credentials are captured when the user enters information for the first time, and then stored and secured for future use.

automatic sign-on
A feature where users can log on to the sign-on automation system and the system logs on the user to all other applications.

B

base distinguished name
A name that indicates the starting point for searches in the directory server.

base image
A template for a virtual desktop.

bidirectional language
A language that uses a script, such as Arabic and Hebrew, whose general flow of text proceeds horizontally from right to left, but numbers, English, and other left-to-right language text are written from left to right.

bind distinguished name
A name that specifies the credentials for the application server to use when connecting to a directory service. The distinguished name uniquely identifies an entry in a directory.

biometrics
The identification of a user based on a physical characteristic of the user, such as a fingerprint, iris, face, voice, or handwriting.

C

CA See certificate authority.

CAPI See cryptographic application programming interface.

Card Serial Number (CSN)
A unique data item that identifies a hybrid smart card. It has no relation to the certificates installed in the smart card

CCOW
See Clinical Context Object Workgroup.

cell A group of managed processes that are federated to the same deployment manager and can include high-availability core groups.

certificate
In computer security, a digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority and is digitally signed by that authority. See also certificate authority.

certificate authority (CA)
A trusted third-party organization or company that issues the digital certificates. The certificate authority typically verifies the identity of the individuals who are granted the unique certificate. See also certificate.

CLI See command-line interface.

Clinical Context Object Workgroup (CCOW)
A vendor independent standard, for the interchange of information between clinical applications in the healthcare industry.

cluster
A group of application servers that collaborate for the purposes of workload balancing and failover.

command-line interface (CLI)
A computer interface in which the input and output are text based.

credential
Information acquired during authentication that describes a user, group associations, or other security-related identity attributes, and that is used to perform services such as authorization, auditing, or delegation. For example, a

user ID and password are credentials that allow access to network and system resources.

cryptographic application programming interface (CAPI)
An application programming interface that provides services to enable developers to secure applications using cryptography. It is a set of dynamically-linked libraries that provides an abstraction layer which isolates programmers from the code used to encrypt the data.

cryptographic service provider (CSP)
A feature of the i5/OS™ operating system that provides APIs. The CCA Cryptographic Service Provider enables a user to run functions on the 4758 Coprocessor.

CSN See Card Serial Number.

CSP See cryptographic service provider.

D

dashboard
An interface that integrates data from a variety of sources and provides a unified display of relevant and in-context information.

database server
A software program that uses a database manager to provide database services to other software programs or computers.

data source
The means by which an application accesses data from a database.

deployment manager
A server that manages and configures operations for a logical group or cell of other servers.

deployment manager profile
A WebSphere® Application Server runtime environment that manages operations for a logical group, or cell, of other servers.

deprovision
To remove a service or component. For example, to deprovision an account means to delete an account from a resource. See also provision.

desktop pool
A collection of virtual desktops of similar configuration intended to be used by a designated group of users.

directory
A file that contains the names and controlling information for objects or other directories.

directory service
A directory of names, profile information, and machine addresses of every user and resource on the network. It manages user accounts and network permissions. When a user name is sent, it returns the attributes of that individual, which might include a telephone number, as well as an email address. Directory services use highly specialized databases that are typically hierarchical in design and provide fast lookups.

disaster recovery
The process of restoring a database, system, policies after a partial or complete site failure that was caused by a catastrophic event such as an earthquake or fire. Typically, disaster recovery requires a full backup at another location.

disaster recovery site
A secondary location for the production environment in case of a disaster.

distinguished name (DN)
The name that uniquely identifies an entry in a directory. A distinguished name is made up of attribute:value pairs, separated by commas. For example, CN=person name and C=country or region.

DLL See dynamic link library.

DN See distinguished name.

DNS See domain name server.

domain name server (DNS)
A server program that supplies name-to-address conversion by mapping domain names to IP addresses.

dynamic link library (DLL)
A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a DLL can be shared by several applications simultaneously.

E

enterprise directory
A directory of user accounts that define IBM Security Access Manager for Enterprise Single Sign-On users. It validates user credentials during sign-up and logon, if the password is synchronized with the enterprise directory password. An example of an enterprise directory is Active Directory.

enterprise single sign-on (ESSO)
A mechanism that allows users to log on to all applications deployed in the enterprise by entering a user ID and other credentials, such as a password.

ESSO See enterprise single sign-on.

event code
A code that represents a specific event that is tracked and logged into the audit log tables.

F

failover
An automatic operation that switches to a redundant or standby system or node in the event of a software, hardware, or network interruption.

fast user switching
A feature that allows users to switch between user accounts on a single workstation without quitting and logging out of applications.

Federal Information Processing Standard (FIPS)
A standard produced by the National Institute of Standards and Technology when national and international standards are nonexistent or inadequate to satisfy the U.S. government requirements.

FIPS See Federal Information Processing Standard.

fix pack
A cumulative collection of fixes that is released between scheduled refresh packs, manufacturing refreshes, or releases. A fix pack updates the system to a specific maintenance level.

FQDN
See fully qualified domain name.

fully qualified domain name (FQDN)
In Internet communications, the name of a host system that includes all of the subnames of the domain name. An example of a fully qualified domain name is rchland.vnet.ibm.com. See also host name.

G

GINA See graphical identification and authentication.

GPO See group policy object.

graphical identification and authentication (GINA)
A dynamic link library that provides a user interface that is tightly integrated with authentication factors and provides password resets and second factor bypass options.

group policy object (GPO)
A collection of group policy settings. Group policy objects are the documents created by the group policy snap-in. Group policy objects are stored at the domain level, and they affect users and computers contained in sites, domains, and organizational units.

H

HA See high availability.

high availability (HA)
The ability of IT services to withstand all outages and continue providing processing capability according to some predefined service level. Covered outages include both planned events, such as maintenance and backups, and unplanned events, such as software failures, hardware failures, power failures, and disasters.

host name
In Internet communication, the name given to a computer. The host name might be a fully qualified domain name such as mycomputer.city.company.com, or it might be a specific subname such as mycomputer. See also fully qualified domain name, IP address.

hot key
A key sequence used to shift operations

between different applications or between different functions of an application.

hybrid smart card
An ISO-7816 compliant smart card which contains a public key cryptography chip and an RFID chip. The cryptographic chip is accessible through contact interface. The RFID chip is accessible through contactless (RF) interface.

I

interactive graphical mode
A series of panels that prompts for information to complete the installation.

IP address
A unique address for a device or logical unit on a network that uses the Internet Protocol standard. See also host name.

J

Java Management Extensions (JMX)
A means of doing management of and through Java technology. JMX is a universal, open extension of the Java programming language for management that can be deployed across all industries, wherever management is needed.

Java runtime environment (JRE)
A subset of a Java developer kit that contains the core executable programs and files that constitute the standard Java platform. The JRE includes the Java virtual machine (JVM), core classes, and supporting files.

Java virtual machine (JVM)
A software implementation of a processor that runs compiled Java code (applets and applications).

JMX See Java Management Extensions.

JRE See Java runtime environment.

JVM See Java virtual machine.

K

keystore
In security, a file or a hardware cryptographic card where identities and private keys are stored, for authentication and encryption purposes. Some keystores also contain trusted or public keys. See also truststore.

L

LDAP See Lightweight Directory Access Protocol.

Lightweight Directory Access Protocol (LDAP)
An open protocol that uses TCP/IP to provide access to directories that support an X.500 model. An LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory.

lightweight mode
A Server AccessAgent mode. Running in lightweight mode reduces the memory footprint of AccessAgent on a Terminal or Citrix Server and improves the single sign-on startup duration.

linked clone
A copy of a virtual machine that shares virtual disks with the parent virtual machine in an ongoing manner.

load balancing
The monitoring of application servers and management of the workload on servers. If one server exceeds its workload, requests are forwarded to another server with more capacity.

lookup user
A user who is authenticated in the Enterprise Directory and searches for other users. IBM Security Access Manager for Enterprise Single Sign-On uses the lookup user to retrieve user attributes from the Active Directory or LDAP enterprise repository.

M

managed node
A node that is federated to a deployment manager and contains a node agent and can contain managed servers. See also node.

mobile authentication
An authentication factor which allows mobile users to sign-on securely to corporate resources from anywhere on the network.

N

network deployment
The deployment of an IMS™ Server on a WebSphere Application Server cluster.

node A logical group of managed servers. See also managed node.

node agent
An administrative agent that manages all application servers on a node and represents the node in the management cell.

O

one-time password (OTP)
A one-use password that is generated for an authentication event, and is sometimes communicated between the client and the server through a secure channel.

OTP See one-time password.

OTP token
A small, highly portable hardware device that the owner carries to authorize access to digital systems and physical assets, or both.

P

password aging
A security feature by which the superuser can specify how often users must change their passwords.

password complexity policy
A policy that specifies the minimum and maximum length of the password, the minimum number of numeric and alphabetic characters, and whether to allow mixed uppercase and lowercase characters.

personal identification number (PIN)
In Cryptographic Support, a unique number assigned by an organization to an individual and used as proof of identity. PINs are commonly assigned by financial institutions to their customers.

PIN See personal identification number.

pinnable state
A state from an AccessProfile widget that can be combined to the main AccessProfile to reuse the AccessProfile widget function.

PKCS See Public Key Cryptography Standards.

policy template
A predefined policy form that helps users define a policy by providing the fixed policy elements that cannot be changed and the variable policy elements that can be changed.

portal A single, secure point of access to diverse information, applications, and people that can be customized and personalized.

presence detector
A device that, when fixed to a computer, detects when a person moves away from it. This device eliminates manually locking the computer upon leaving it for a short time.

primary authentication factor
The IBM Security Access Manager for Enterprise Single Sign-On password or directory server credentials.

private key
In computer security, the secret half of a cryptographic key pair that is used with a public key algorithm. The private key is known only to its owner. Private keys are typically used to digitally sign data and to decrypt data that has been encrypted with the corresponding public key.

provision
To provide, deploy, and track a service, component, application, or resource. See also deprovision.

provisioning API
An interface that allows IBM Security Access Manager for Enterprise Single Sign-On to integrate with user provisioning systems.

provisioning bridge
An automatic IMS Server credential distribution process with third party provisioning systems that uses API libraries with a SOAP connection.

provisioning system
A system that provides identity lifecycle management for application users in enterprises and manages their credentials.

Public Key Cryptography Standards (PKCS)
A set of industry-standard protocols used for secure information exchange on the Internet. Domino® Certificate Authority and Server Certificate Administration applications can accept certificates in PKCS format.

published application
An application installed on Citrix XenApp server that can be accessed from Citrix ICA Clients.

published desktop
A Citrix XenApp feature where users have remote access to a full Windows desktop from any device, anywhere, at any time.

R

radio frequency identification (RFID)
An automatic identification and data capture technology that identifies unique items and transmits data using radio waves. See also active radio frequency identification.

random password
An arbitrarily generated password used to increase authentication security between clients and servers.

RDP See remote desktop protocol.

registry
A repository that contains access and configuration information for users, systems, and software.

registry hive
In Windows systems, the structure of the data stored in the registry.

remote desktop protocol (RDP)
A protocol that facilitates remote display and input over network connections for Windows-based server applications. RDP supports different network topologies and multiple connections.

replication
The process of maintaining a defined set of data in more than one location. Replication involves copying designated changes for one location (a source) to another (a target) and synchronizing the data in both locations.

revoke
To remove a privilege or an authority from an authorization identifier.

RFID See radio frequency identification.

root CA
See root certificate authority.

root certificate authority (root CA)
The certificate authority at the top of the hierarchy of authorities by which the identity of a certificate holder can be verified.

S

scope A reference to the applicability of a policy, at the system, user, or machine level.

secret question
A question whose answer is known only to the user. A secret question is used as a security feature to verify the identity of a user.

secure remote access
The solution that provides web browser-based single sign-on to all applications from outside the firewall.

Secure Sockets Layer (SSL)
A security protocol that provides communication privacy. With SSL, client/server applications can communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

Secure Sockets Layer virtual private network (SSL VPN)
A form of VPN that can be used with a standard web browser.

Security Token Service (STS)
A web service that is used for issuing and exchanging security tokens.

security trust service chain
A group of module instances that are configured for use together. Each module instance in the chain is called in turn to perform a specific function as part of the overall processing of a request.

serial ID service provider interface
A programmatic interface intended for integrating AccessAgent with third-party Serial ID devices used for two-factor authentication.

serial number
A unique number embedded in the IBM Security Access Manager for Enterprise Single Sign-On keys, which is unique to each key and cannot be changed.

server locator
A locator that groups a related set of web applications that require authentication by the same authentication service. In AccessStudio, server locators identify the authentication service with which an application screen is associated.

service provider interface (SPI)
An interface through which vendors can integrate any device with serial numbers with IBM Security Access Manager for Enterprise Single Sign-On and use the device as a second factor in AccessAgent.

signature
In profiling, unique identification information for any application, window, or field.

sign-on automation
A technology that works with application user interfaces to automate the sign-on process for users.

sign up
To request a resource.

silent mode
A method for installing or uninstalling a product component from the command line with no GUI display. When using silent mode, you specify the data required by the installation or uninstallation program directly on the command line or in a file (called an option file or response file).

Simple Mail Transfer Protocol (SMTP)
An Internet application protocol for transferring mail among users of the Internet.

single sign-on (SSO)
An authentication process in which a user can access more than one system or application by entering a single user ID and password.

smart card
An intelligent token that is embedded with an integrated circuit chip that provides memory capacity and computational capabilities.

smart card middleware
Software that acts as an interface between smart card applications and the smart card hardware. Typically the software consists of libraries that implement PKCS#11 and CAPI interfaces to smart cards.

SMTP See Simple Mail Transfer Protocol.

snapshot
A captured state, data, and hardware configuration of a running virtual machine.

SOAP A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet. See also web service.

SPI See service provider interface.

SSL See Secure Sockets Layer.

SSL VPN
See Secure Sockets Layer virtual private network.

SSO See single sign-on.

stand-alone deployment
A deployment where the IMS Server is deployed on an independent WebSphere Application Server profile.

stand-alone server
A fully operational server that is managed independently of all other servers, using its own administrative console.

strong authentication
A solution that uses multifactor authentication devices to prevent unauthorized access to confidential corporate information and IT networks, both inside and outside the corporate perimeter.

strong digital identity
An online persona that is difficult to impersonate, possibly secured by private keys on a smart card.

STS See Security Token Service.

system modal message
A system dialog box that is typically used to display important messages. When a system modal message is displayed,

nothing else can be selected on the screen until the message is closed.

T

terminal emulator
A program that allows a device such as a microcomputer or personal computer to enter and receive data from a computer system as if it were a particular type of attached terminal.

terminal type (tty)
A generic device driver for a text display. A tty typically performs input and output on a character-by-character basis.

thin client
A client that has little or no installed software but has access to software that is managed and delivered by network servers that are attached to it. A thin client is an alternative to a full-function client such as a workstation.

transparent screen lock
An feature that, when enabled, permits users to lock their desktop screens but still see the contents of their desktop.

trigger
In profiling, an event that causes transitions between states in a states engine, such as, the loading of a web page or the appearance of a window on the desktop.

trust service chain
A chain of modules that operate in different modes such as validate, map, and issue truststore.

truststore
In security, a storage object, either a file or a hardware cryptographic card, where public keys are stored in the form of trusted certificates, for authentication purposes in web transactions. In some applications, these trusted certificates are moved into the application keystore to be stored with the private keys. See also keystore.

tty See terminal type.

two-factor authentication
The use of two factors to authenticate a user. For example, the use of password and an RFID card to log on to AccessAgent.

U

uniform resource identifier
A compact string of characters for identifying an abstract or physical resource.

user credential
Information acquired during authentication that describes a user, group associations, or other security-related identity attributes, and that is used to perform services such as authorization, auditing, or delegation. For example, a user ID and password are credentials that allow access to network and system resources.

user deprovisioning
The process of removing a user account from IBM Security Access Manager for Enterprise Single Sign-On.

user provisioning
The process of signing up a user to use IBM Security Access Manager for Enterprise Single Sign-On.

V

VB See Visual Basic.

virtual appliance
A virtual machine image with a specific application purpose that is deployed to virtualization platforms.

virtual channel connector
A connector that is used in a terminal services environment. The virtual channel connector establishes a virtual communication channel to manage the remote sessions between the Client AccessAgent component and the Server AccessAgent.

virtual desktop
A user interface in a virtualized environment, stored on a remote server.

virtual desktop infrastructure
An infrastructure that consists of desktop operating systems hosted within virtual machines on a centralized server.

Virtual Member Manager (VMM)
A WebSphere Application Server component that provides applications with a secure facility to access basic

organizational entity data such as people, logon accounts, and security roles.

virtual private network (VPN)
An extension of a company intranet over the existing framework of either a public or private network. A VPN ensures that the data that is sent between the two endpoints of its connection remains secure.

Visual Basic (VB)
An event-driven programming language and integrated development environment (IDE) from Microsoft.

VMM See Virtual Member Manager.

VPN See virtual private network.

W

wallet A secured data store of access credentials of a user and related information, which includes user IDs, passwords, certificates, encryption keys.

wallet caching
The process during single sign-on for an application whereby AccessAgent retrieves the logon credentials from the user credential wallet. The user credential wallet is downloaded on the user machine and stored securely on the IMS Server.

wallet manager
The IBM Security Access Manager for Enterprise Single Sign-On GUI component that lets users manage application credentials in the personal identity wallet.

web server
A software program that is capable of servicing Hypertext Transfer Protocol (HTTP) requests.

web service
A self-contained, self-describing modular application that can be published, discovered, and invoked over a network using standard network protocols. Typically, XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available. See also SOAP.

WS-Trust
A web services security specification that defines a framework for trust models to establish trust between web services.

Index

Printed in USA