Sterling Selling and Fulfillment Foundation

Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide

Release 9.1.0.24

Sterling Selling and Fulfillment Foundation

Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide

Release 9.1.0.24

Note
Before using this information and the product it supports, read the information in "Notices" on page 23.

Contents

Chapter 1. Roadmap: Using the PA-DSS, Secure Deployment, and SSDCS Documentation Guides

The IBM® Sterling Selling and Fulfillment Foundation provides a strategy for secure credit card capture and protection, in accordance with the Payment Application Data Security Standard (PA-DSS) and the Payment Card Industry Data Security Standard (PCI DSS).

If your deployment captures credit cards, you can implement the IBM Sterling Sensitive Data Capture Server (SSDCS) to capture credit card numbers on behalf of the IBM applications. Doing so ensures that credit card numbers are kept outside of IBM Sterling Selling and Fulfillment Suite applications, with the added benefit that these applications are kept outside of PCI DSS auditing scope.

The following guides in the Sterling Selling and Fulfillment Foundation documentation set discuss how to implement these security strategies:

- *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* (this guide) - Describes the steps to follow for your SSDCS installation to remain in compliance with the PA-DSS. It also describes order capture and payment processing data flows, as well as showing a typical network implementation of the SSDCS. This guide explains how to keep the Sterling Selling and Fulfillment Suite applications outside of the PCI DSS auditing scope.
- *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide* - Explains how to deploy the Sterling Selling and Fulfillment Foundation securely. It covers security recommendations for applications, networks, operating systems, databases, application servers, and message queues.
- *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* - Details how to install, configure, and deploy SSDCS as a proxy service that IBM applications call to tokenize Primary Account Numbers (PANs) for credit cards and gift value cards.

Implementation Sequence

To implement these strategies, IBM suggests that you follow this sequence of steps:

1. Review all three guides in this order:
   a. *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide*
   b. *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*
   c. *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide*
2. Implement the steps suggested in the *PA-DSS Implementation Guide* to remain in compliance with PA-DSS and keep your IBM applications outside of the PCI DSS auditing scope.
3. Implement the security strategies outlined in the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*.
4. Install Sterling Selling and Fulfillment Foundation and associated applications (refer to the *Sterling Selling and Fulfillment Foundation: Installation Guide* and respective application installation guides).

5. Follow the steps in the *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* to configure your SSDCS implementation.

Chapter 2. Overview of the Sterling Sensitive Data Capture Server

The Sterling Sensitive Data Capture Server (SSDCS) is an application that integrates with the Sterling Selling and Fulfillment Suite to ensure that credit card numbers and stored value card numbers are secure by tokenizing them. SSDCS enables the Sterling Selling and Fulfillment Suite to achieve Payment Application Data Security Standard (PA-DSS) compliance.

Payment Application Data Security Standard and Payment Card Industry Data Security Standard (PCI-DSS) are standards established by the payment card industry to promote secure payment processing. PA-DSS is a standard for payment applications, which are systems that capture information pertaining to cards, such as credit cards. It mandates that sensitive information, such as a credit card number, should not be stored along with application data. PCI-DSS is a standard for the secure implementation of payment applications by customers. PA-DSS certification is awarded to third-party payment application vendors who comply with the standard. The Sterling Selling and Fulfillment Suite qualifies as a third-party payment application.

The Sterling Selling and Fulfillment Suite does not directly process credit cards, but rather provides user exits in which customers can write code to contact a payment gateway. The focus for PA-DSS compliance in the Sterling Selling and Fulfillment Suite is the tokenization of credit card information before it enters the system. This is achieved with the support of SSDCS, which interprets, validates, and tokenizes credit card numbers and stored value card numbers. Tokenization is the process of storing credit card numbers or stored value card numbers in a vault system that associates a token to a securely stored credit card number or stored value card number. The only way in which a token can be returned to its original value is by contacting the vault system.

Note: You must use your organization's vault solution.

SSDCS ensures that credit card numbers and stored value card numbers are stored only in a vault system and not in the Sterling Selling and Fulfillment Suite. As a result, SSDCS is the only application in the Sterling Selling and Fulfillment Suite that is within scope for PA-DSS compliance.

The SSDCS is invoked by the credit card capture process through the Sterling Selling and Fulfillment Foundation User Interface. No APIs were enhanced for tokenization. A token should be passed to the API instead of passing a primary account number (PAN).

For information about how to configure SSDCS securely, refer to the *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide*.

Note: SSDCS is a system-critical application. If SSDCS is unavailable, payment information cannot be captured.

Chapter 3. Installing the Sterling Sensitive Data Capture Server

About this task

This procedure assumes that you have already installed Sterling Selling and Fulfillment Foundation.

The Sterling Sensitive Data Capture Server (SSDCS) application is packaged as a compressed file with Sterling Selling and Fulfillment Foundation. The compressed file is located in <*INSTALL_DIR*>/repository/external/ssdcs.zip.

You must use the same technical stack for the Sterling Sensitive Data Capture Server (SSDCS) application as for Sterling Selling and Fulfillment Foundation. For information about the minimum requirements for installing SSDCS, refer to the table titled "Supported Application Server Tier" in the *Selling and Fulfillment Foundation: System Requirements Guide*.

Before installing the Sterling Sensitive Data Capture Server (SSDCS) application, you must read the *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* for information about how to configure SSDCS securely.

To install the Sterling Sensitive Data Capture Server (SSDCS) application:

Procedure

1. Determine the location in which the SSDCS files will reside, and give the directory a name. Define a variable, <*SSDCS_DIR*>, for this location.

   Note: Throughout this guide, <*SSDCS_DIR*> refers to the directory in which you have installed SSDCS.
2. In Sterling Selling and Fulfillment Foundation, use the customer_overrides.properties file to specify a value for the yfs.ssdcs.url property. For additional information about this property and how to override it using the customer_overrides.properties file, refer to the *Sterling Selling and Fulfillment Foundation: Properties Guide*.
3. Navigate to the <*INSTALL_DIR*>/repository/external directory and locate the ssdcs.zip file. This compressed file contains all the files that are required for SSDCS.
4. Copy ssdcs.zip from the <*INSTALL_DIR*>/repository/external directory to the <*SSDCS_DIR*> directory.
5. In the <*SSDCS_DIR*> directory, extract the ssdcs.zip file.

   The ssdcs.zip file contains the following directories:
   - bin - Contains the scripts to build the WAR file. As part of this, JAR files in jar/extn will be picked up and put in the appserver classpath each time the WAR file is built. No other scripts are necessary.
   - documentation - Contains the Javadocs compiled from the source (the implementable interfaces and user exits).
   - jar - Contains the compiled product JAR and third-party JARs.
   - jar/extn - A placeholder directory for customer extension JAR files.

- `log` - A placeholder directory for log files. It is recommended that the SSDCS_LOG_DIR environment variable is set to point to this location. For more information about log files, refer to Configuring Logging for the SSDCS.
- `properties` - Contains the property files that configure the application and logging. A JAR file of the properties in this directory will be built and placed in this directory as part of the WAR build process. For more information about the properties that you can configure for SSDCS, refer to Configuring Properties for the SSDCS.
- `resources/eardata/descriptors/**` - Contains miscellaneous WAR files that are categorized by appserver; for example, `web.xml`.
- `WebContent/jsp` - Contains `status.jsp`, which is a public JSP that is displayed if SSDCS is available. If `status.jsp` is not displayed, processing of payment information cannot continue.
- `WebContent/WEB-INF/jsp` - Contains `ssdcs_tokenize_pan.jsp`, which is a private JSP that sets the initial parameters (the authorization token and the payment type), and automatically submits to the payment server. The response from the payment server is the Payment Capture JSP. This directory also contains custom JSPs, if any.
- `external_deployments` - The location of the compiled WAR file.
- `build` - A temporary directory created as part of the build WAR process.

6. Configure the SSDCS application.

   You must configure a minimum of two properties to successfully run SSDCS. For information about how to configure SSDCS, refer to Configuring the Sterling Sensitive Data Capture Server.

7. Deploy the SSDCS application.

   For information about how to deploy SSDCS, refer to Deploying the Sterling Sensitive Data Capture Server.

Chapter 4. Configuring the Sterling Sensitive Data Capture Server

Configuration of the Sterling Sensitive Data Capture Server (SSDCS) is described in this section.

For information about configuring SSDCS securely, refer to the *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide.*

Configuring Properties for the SSDCS

Property files in Sterling Selling and Fulfillment Foundation and SSDCS contain properties that control the operation of the SSDCS application.

Sterling Selling and Fulfillment Foundation Properties

Use the `customer_overrides.properties` file to configure the following Sterling Selling and Fulfillment Foundation properties for SSDCS:

- yfs.ssdcs.url
- yfs.ssdcs.servlet
- yfs.ssdcs.jsp
- yfs.ssdcs.tokenize.svc
- yfs.ssdcs.tokenize.cc
- sc.access.token.expire.in.seconds
- sc.access.token.max.allowed.expire.in.seconds

For additional information about these properties and how to override them using the `customer_overrides.properties file`, refer to the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

SSDCS Subdomain Support

Integration with the Sterling Sensitive Data Capture Server (SSDCS) is achieved in web applications via an IFRAME. This IFRAME contains the field in which the primary account number (PAN) is entered, and returns a token for storage in Sterling Selling and Fulfillment Suite's database. To retrieve the token from the IFRAME, the “Same Origin Policy” must be upheld. This means that the protocol (http or https), domain, and port number must be identical. The Same Origin Policy permits scripts running on pages originating from the same site to access each other's methods and properties with no specific restrictions. The policy also prevents access to methods and properties across different sites.

To achieve this while keeping the SSDCS—which can contact payment systems—in a separate and more secure network zone, a load balancer must be used that can rewrite the URL requests to map to the different machines. This can increase processing for the load balancer and slow network performance, an effect you can mitigate by adding the property `document.domain` to each web application. This will exempt the subdomain from the Same Origin Policy and simplify the workload for the load balancer.

For example, if you set document.domain to *yourcompany*.com, then http://smcfs.*yourcompany*.com/ and http://ssdcs.*yourcompany*.com/ would both meet adherence to the Same Origin Policy for the purpose of SSDCS.

For more information about the document.domain property, see "Configuring SSDCS Properties." For information about SSDCS deployment and Layer 7 routing considerations, refer to the *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide*.

Configuring SSDCS Properties

The ssdcs.properties file contains the SSDCS properties and is located in the <*SSDCS_DIR*>/properties directory. You can configure these properties, which are described in the following table, by editing the ssdcs.properties file:

Property	Value	Description
ssdcs.smcfs.url	Valid value = https://<*host*>:<*port*>/<*application context root*>/accessTokenServlet Default = https://<*host*>:<*port*>/smcfs/accessTokenServlet	This property points to the URL that is used to access Sterling Selling and Fulfillment Foundation. The variables <*host*> and <*port*> are placeholders that you must replace with the name of the machine (not the IP address) and with the port. You can change the default application context root (smcfs) to connect to one of the following applications: sbc, sfs, swc. Example: ssdcs.smcfs.url=https://*yourcompany*.com:8080/smcfs/accessTokenServlet
ssdcs.document.domain	A subdomain of the domain that the user is running on.	If a subdomain is specified, this setting exempts this subdomain from the Same Origin Policy. This property is read by the user interface and used as the document.domain value. For example, http://smcfs.*yourcompany*.com/ and http://ssdcs.*yourcompany*.com/ would meet the Same Origin Policy if they both set document.domain to *yourcompany*.com. Example: ssdcs.document.domain=*yourcompany*.com This property must also be set in the yfs.properties file as yfs.document.domain=*yourcompany*.com. For more information, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*. For more information about Same Origin Policy, see "SSDCS Subdomain Support" on page 7.
User Exit	**Note:** The SSDCS user exits are implemented through the ssdcs.properties file only. SSDCS user exits do not use the same user exit framework as the Sterling Selling and Fulfillment Foundation user exits and are implemented by providing a fully qualified classname for the related property, defined below. For more information about SSDCS user exits, refer to the *SSDCS Javadocs*.	

Property	Value	Description
ssdcs.ue.PanValidator	Valid value = the fully qualified classname to an implementation of ISSDCSPanValidator Default = N/A	This property identifies the Primary Account Number (PAN) validation implementation. This is an optional user exit. Example: `ssdcs.ue.PanValidator = ssdcs.ue.ISSDCSPanValidatorImpl`
ssdcs.ue.Tokenize	Valid value = the fully qualified classname to an implementation of ISSDCSTokenize Default = N/A	This property identifies the tokenization implementation. This is a required user exit for tokenization. Example: `ssdcs.ue.Tokenize = ssdcs.ue.ISSDCSTokenizeImpl`

ESAPI Properties

You can use the ESAPI.properties file to configure properties for the OWASP Enterprise Security API. This file contains validation patterns that have `Validator.ssdcs.` as a prefix. Do not modify any other properties in this file. For more information about modifying the ESAPI.properties file, refer to SSDCS Validations.

Configuring Logging for the SSDCS

SSDCS includes basic logging functionality. However, you can change logging parameters in the `log4j` configuration XML file to control the location and level of the log files.

Note: Before setting up the logging parameters, ensure that you understand the log4j utility. For detailed information about this utility, refer to the following Web site:

`http://jakarta.apache.org/log4j`

A placeholder directory, `<SSDCS_DIR>/log`, is provided for storing the SSDCS log files. It is recommended that you set up an environment variable, such as SSDCS_LOG_DIR, to point to this location.

Log Files Provided by SSDCS

SSDCS provides the following log files, which can have different configurations:

- `ssdcs.log` - Used for logging business logic issues, such as debugging and timing information. For example, the system records anything that happens in the servlet as part of the tokenization process.

  Note:

  – Sensitive information, such as credit card numbers or stored value card numbers, is not logged.

  – This is the only log file that has a TIMER logging level.

  – In order to view the log initialization entry, you must set the logging level to DEBUG.

- `ssdcs_security.log` - Used exclusively for logging security issues. The SSDCS security logger records the activity of any malicious requests that are detected, such as unauthorized users attempting to log in.
- `ssdcs_esapi.log` - Used for debugging the ESAPI setup. Both ESAPI internal classes and SSDCS extensions and implementations are logged here.

Configurable Parameters for SSDCS Logging

The following table describes the logging parameters that you can configure.

Property	Description
In the log4j Configuration XML File	
<priority> subelement of the <root> element	Specify the level of logging required. It is recommended that the value of this attribute is set to ERROR. Following are the valid values for logging levels: • FATAL • ERROR • WARN • INFO • TIMER • DEBUG
<appender> subelement	At the root level, this attribute specifies the associated name and class attribute. Select a valid log4j appender class. Each subelement can also specify the layout of the message through the <layout> subelement, and can filter for levels through the <filter> subelement. Instead of hardcoding the absolute path for the log file under the appender to be used, it is recommended that customers use a `${SSDCS_LOG_DIR}` parameter in the `log4j.properties.xml` file and invoke the JVM with `-DSSDCS_LOG_DIR=<`*application_log_directory*`>`.
<param> subelement of the <appender> element	This attribute specifies the associated name and value attributes. You can set the following variables using the <param> attribute: • maxLogSize - Specify the maximum number of write operations to be made to a log file. This is not the memory size limit of the log file. The size of the log file will depend on the size of each write operation. By default, the value of this variable is set to 100000. • rotateLogs - Determines whether the log files should be split or not. If the value of this variable is set to False, the log files will not be split, and the logger will keep writing in the same file. By default, the value of this variable is set to True.

Form Fields for SSDCS Input

The following table describes the form fields in the HTML, in the order in which they are passed into and out of the JSP. If a value is not returned in the output, the field is returned empty. The form field name and ID are identical. Some columns always passthrough; others will passthrough until a successful tokenization. For example, the servlet never modifies the ssdcsAuthenticationToken, but will recalculate a display value if a token is generated. Fields marked as In Title will

appear in the Title of the page in the order listed in the following table, separated by a |. The only mandatory field is ssdcsAuthenticationToken.

Field Name	Description	In Title	Pass-through
ssdcsAuthenticationToken	The token that is validated against Sterling Selling and Fulfillment Foundation to ensure that the request is authorized.	No	Always
ssdcsCssUrl	The URL used to include the style sheet for the JSP. It is the only attribute that will not appear in the title of the page, because it is used by only SSDCS, and there is no value in its return.	No	Always
ssdcsRedirectUrl	The URL used as the output of the servlet call. If it is not passed, the servlet will attempt to redirect to `/jsp/ssdcs_tokenize_pan.jsp`. There are two allowed values: • `/jsp/ssdcs_tokenize_pan.jsp` • `/jsp/ssdcs_xml.out`	No	Always
ssdcsDataType	The type of data being submitted to the form. Two supported values can be passed: • ssdcsCreditCardNumber • ssdcsStoredValueCardNumber If the passed value is not one of these two values, the Luhn algorithm will not be run.	Yes	Always
ssdcsDataTypeDetail	Additional information about the DataType. For ssdcsCreditCardNumber or ssdcsStoredValueCardNumber, this is expected to be the Sterling Selling and Fulfillment Foundation Payment TypeId. **Note:** ssdcsDataTypeDetail is ignored when determining if a hidden field's value has changed. ssdcsDataTypeDetail modification is allowed.	No	Always
ssdcsDataToTokenize	The data value to tokenize, for example, a credit card number. This is never returned. It is the data display value that is returned on SUCCESS; the field is empty on FAIL. This is the only field that is not a hidden field. It is the text box on the JSP that will contain the credit card number or stored value card number that requires tokenization.	No	No

Field Name	Description	In Title	Pass-through
ssdcsToken	The tokenized value. This will return its input unless there is a successful tokenization call. For example, if the data validation fails in an edit scenario, the previous token value is returned.	Yes	Until success
ssdcsDisplayValue	The value to display to the user instead of the token or sensitive data (credit card number or stored value card number). If ssdcsDataToTokenize is not passed, this is a passthrough. By default, it is computed as the last four digits of ssdcsDataToTokenize, but it can be manually set in the Tokenize user exit.	Yes	Until success
ssdcsResultCode	The result of the servlet invocation: • <BLANK> - Not a returnable status. It is used during initial form loading to understand that the only processing performed is authentication. No tokenization is occurring. • INITIAL - The result code returned when the form is initially loaded (that is, the output code of <BLANK>), before the credit card number or stored value card number is entered. • FAIL - The result code returned on failure. This can bypass the INITIAL state if failure occurs in the data validation, or if authentication fails. • SUCCESS - The result code returned on success.	Yes	No
ssdcsFailReason	Explains why the tokenization fails: • INVALID_SESSION - Authentication with Sterling Selling and Fulfillment Foundation failed. • INVALID_DATA - ssdcsDataToTokenize was incorrect. It did not pass data validation, or, in the case of a credit card number or stored value card number, it did not pass the primary account number validation (that is, the Luhn algorithm). • TOKENIZATION_FAILED - Tokenization implementation did not return a token.	Yes	No
ssdcsResultDescription	A free-form field that can be used to provide a readable description of the result of a call. It is reset for each call. It may be returned when some internal errors occur.	Yes	No

Field Name	Description	In Title	Pass-through
ssdcsTabIndex	Set as the HTML tab index attribute. It is used to control the tab stops on the JSP.	No	Always
ssdcsAdditionalResultData	Additional result data to be understood by the calling UI. In the provided JSP, it is a list in the form Name:Value; containing the following data: CardType:<VISA,MASTERCARD,etc>; This is reset for each call.	Yes	No
ssdcsAutoSubmit	A flag. When set to yes, this instructs the JSP that the UI Framework will not submit the form, and to provide automatic submission, such as an onBlur method.	No	Always
ssdcsDbg	A honeypot parameter. There is no business logic on this parameter. Instead, the value is hardcoded to N on the initial load. If it is ever modified, a security log is generated.	No	Always

The title will contain the fields described in the previous table (in the order shown in the table), separated by the character |. An example output for the title:

```
ssdcsCreditCardNumber|||FAIL|INVALID_DATA|This is not a valid credit card number|
ssdcsCreditCardNumber|400000013246|3246|SUCCESS|||CardType:VISA;
```

Note: There is a trailing | at the end of the first line.

SSDCS Validations

SSDCS provides input validation, session validation, and primary account number (that is, credit card number or stored value card number) validation.

Input Validation

Input validation occurs on the fields that are passed in, as described in the following table. All alphabetical characters accept uppercase and lowercase letters. If a field is listed as N/A, its input is discarded immediately after invoking the servlet because its result is always computed and its previous result is not desired. Unless otherwise specified, numeric validation does not validate negative numbers.

Field Name	Extensible	Validation
ssdcsAuthenticationToken	Yes	Alphanumeric
ssdcsCssUrl	Yes	This must be the same domain name as the request, with an optional port. The remainder of the URL must consist of alphanumeric characters, a period, a slash, an underscore, or a hyphen, and it must end with .css.
ssdcsRedirectUrl	Yes	This must begin with /jsp/, end with .jsp, and contain only alphabetical characters or underscores in between.

Field Name	Extensible	Validation
ssdcsDataType	Yes	Alphanumeric plus underscore.
ssdcsDataTypeDetail	By data type	Known DataType: Alphanumeric plus underscore, space, and hyphen Unknown DataType: No match allowed.
ssdcsDataToTokenize	By data type	Known DataType: Numeric. Prior to validation, all special characters are stripped. Unknown DataType: No match allowed.
ssdcsToken	Yes	Alphanumeric
ssdcsDisplayValue	By data type	Known DataType: Numeric Unknown DataType: No match allowed.
ssdcsResultCode	No	INITIAL, FAIL, or SUCCESS
ssdcsFailReason	No	N/A
ssdcsResultDescription	No	N/A
ssdcsTabIndex	Yes	Numeric, including negative numbers
ssdcsAdditionalResultData	No	N/A
ssdcsAutoSubmit	No	Y or N
ssdcsDbg	Yes	Y or N

Validation patterns are stored in the `ESAPI.properties` file, and have `Validator.ssdcs.` as a prefix. The ESAPI reference implementation will internally use the patterns that have `Validator.` as a prefix. If the property is not configured, the defaults listed in the previous table are used. The Extensible column in the previous table indicates the fields that support extensibility by data type by additionally prefixing the value of ssdcsDataType in the Validation property. The ssdcsDataType will be checked first to see if the validation is extended. For example, to set a different validator for credit card numbers:

```
Validator.ssdcs.ssdcsCreditCardNumber.ssdcsDataToTokenize
Validator.ssdcs.ssdcsDataToTokenize
```

Session Validation

Before SSDCS processes any credit card number or stored value card number, it validates with Sterling Selling and Fulfillment Foundation that the session token passed to it in the ssdcsAuthenticationToken field is a valid session. This token is created using the createAccessToken API call. For information about the createAccessToken API, refer to the *Sterling Selling and Fulfillment Foundation Javadocs*. If the session is not valid, the Sterling Selling and Fulfillment Foundation servlet returns an INVALID_SESSION error.

Credit Card and Stored Value Card Validation

SSDCS validates if the credit card number has been entered correctly by running the Luhn algorithm. An interface is also exposed to enable the implementation of an alternative validation algorithm (for example, to validate a stored value card). The ssdcs.ue.PanValidator user exit property is used to configure the implementation of this interface. When this is implemented, the built-in algorithm will be skipped and the implementation will return the result. This user exit may defer validation to the internal Luhn algorithm by not returning a validation value

for credit cards. Stored value cards may use the user exit to validate them. However, only credit cards will run the internal Luhn algorithm, both after or instead of the user exit call. By default, all the stored value card numbers are valid.

Both the interface and the default logic will also compute the Credit Card Type as part of the validation, if possible. For example, all Visa credit cards begin with a 4, so if the credit card number passes validation, then the card type will be returned as Visa.

Note: The ssdcs.ue.PanValidator user exit contains the name of the payment type configured in the Sterling Selling and Fulfillment Suite in its input, which is populated from the ssdcsDataTypeDetail field. This field is provided to customize branching logic, based on the payment type being tokenized. IBM recommends that if the payment type is an unknown type, you revert to a default logic that will still provide validation.

Tokenization Process

The tokenization process occurs after the validation of the credit card or stored value card is completed. Tokenization is implemented through the SSDCS user exits, which are configured in the ssdcs.properties file. The ssdcs.ue.Tokenize user exit property must be configured in order to implement tokenization. For more information about the SSDCS user exits, refer to Configuring Properties for the SSDCS.

The return of the tokenization call will include the token as well as the display number. If it is not provided or is longer than four characters, the last four digits of the credit card number or stored value card number will be returned as the display number.

Note: The ssdcs.ue.Tokenize user exit contains the name of the payment type configured in the Sterling Selling and Fulfillment Suite in its input, which is populated from the ssdcsDataTypeDetail field. This field is provided to customize branching logic, based on the payment type being tokenized. IBM recommends that if the payment type is an unknown type, you revert to a default logic that will still provide tokenization.

SSDCS Output Formats

Sterling Sensitive Data Capture Server provides two output formats for data: JSP and XML.

JSP Output for SSDCS

The JSPs for SSDCS are embedded within an application screen (for example, within an IFRAME) that captures credit card numbers and stored value card numbers. The body of a JSP consists of a single form. The form displays a single text field for the entry of a credit card number or a stored value card number, and also contains hidden form fields that represent all possible inputs and outputs.

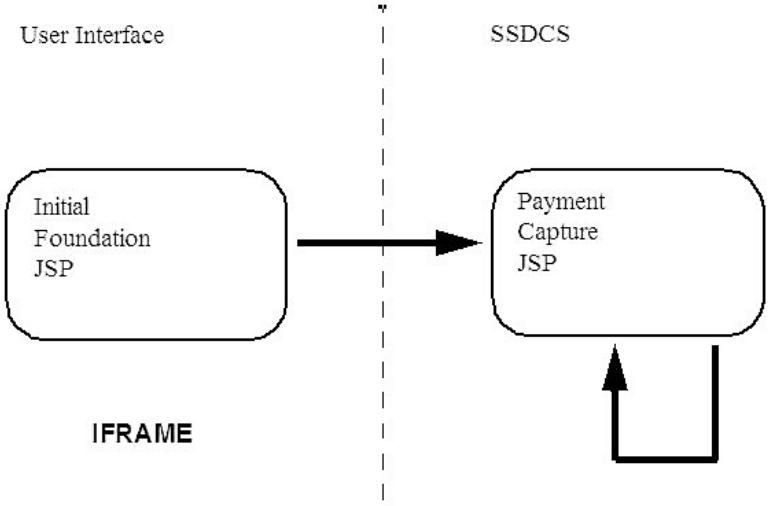
The following process occurs when a credit card number or stored value card number is entered in the text field of the JSP:

1. SSDCS validates whether the data entered in the form fields is valid data. For example, the ssdcsAuthenticationToken field must consist of only alphanumeric characters. For details about the form fields, refer to Form Fields for SSCDS Input and SSDCS Validations.

2. SSDCS validates whether the session token passed to it from Sterling Selling and Fulfillment Foundation is a valid session. If the session is not valid, Sterling Selling and Fulfillment Foundation returns an error.
3. SSDCS validates the credit card number or stored value card number that is entered. For credit card numbers, this involves running the Luhn algorithm. An interface is also provided to implement an alternative validation algorithm. If the number is not valid, SSDCS returns an error. For additional information about the validation of credit card numbers and stored value card numbers, refer to SSDCS Validations.
4. After the credit card number or stored value card number is validated, SSDCS makes an API call to the vault to tokenize the number.
5. The vault returns a token as well as a display value. The display value is what is displayed in the JSP text field instead of the credit card number or stored value card number that was entered.

Before SSDCS is invoked, the IFRAME is populated with an initial, blank JSP that is local to the user's application. This JSP is responsible for setting the initial parameters (that is, the authorization token and the payment type) and auto submitting immediately to SSDCS. When SSDCS is invoked to tokenize a credit card number or stored value card number, the response from SSDCS is the Payment Capture JSP, which, upon submission, will return itself. The Payment Capture JSP is the same as the initial JSP, with the exception of the following: in the Payment Capture JSP, the hidden form fields are populated with computed data, and the text entry field is populated with the returned display number instead of the credit card number or stored value card number. If a submission error occurs, the same form is used to correct the credit card number based on the error. Similarly, if a user decides to use a different credit card, and therefore has to enter a new credit card number, the same form is used.

The following figure depicts how the Payment Capture JSP submits back to itself every time SSDCS is invoked.



The path to the JSP is `/jsp/ssdcs_tokenize_pan.jsp`. The form is `ssdcs`. The text field has two possible values for the class attribute for CSS styling: ssdcsPan and ssdcsPanError, which allow you to set an alternative styling for situations when tokenization fails. For information about how to customize the style of JSP pages, refer to the applicable customization guide for your application.

Note: You must customize the style of JSP pages for SSDCS from your application. You cannot customize the style of JSPs from SSDCS.

Direct access to all the JSPs is prevented, except for `status.jsp`, which is located in the `/jsp` directory and can be used to determine whether or not the server is available.

XML Output for SSDCS

Applications such as IBM Sterling Store Associate are PA-DSS compliant and integrate with the Sterling Sensitive Data Capture Server. Rather than using a browser widget to implement the entry field, the PAN is captured by the application and submitted to SSDCS for tokenization.

Tokenization on the first request to SSDCS is accomplished by posting the `ssdcsResultCode` parameter as INITIAL. This tokenizes the `ssdcsDataToTokenize` field on the first request, rather than only painting a UI for subsequent requests.

To simplify the output for applications, the `/jsp/ssdcs_xml_out.jsp` has the following format, while ssdcsRedirectURL is used as the output of the servlet call:

```
<Tokenize>
<TokenInformation DataType="" Token="" DisplayValue="" AdditionalResultData=""/>
<ResultInformation ResultCode="" FailReason="" ResultDescription=""/>
</Tokenize>
```

All attributes in the XML are those exposed in the title for Rich Client Platform (RCP) applications. They are grouped into two elements: the first for the token and related information, and the second for the result and response codes.

Implementing SSDCS Custom Code

To implement your own custom code, create a JAR file in the `<SSDCS_DIR>/jar/extn` directory. The SSDCS deployment script automatically includes this JAR file when it is located in this directory, and your custom code will be picked up by the `ant` command.

For more information about adding your own custom code, see the *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide*.

Chapter 5. Deploying the Sterling Sensitive Data Capture Server

About this task

You must deploy the Sterling Sensitive Data Capture Server (SSDCS) application in secure mode, otherwise an INVALID_SESSION error occurs when Sterling Sensitive Data Capture Server tries to validate the session token.

To deploy the Sterling Sensitive Data Capture Server application:

Procedure

1. Open a command window and navigate to <*SSDCS_DIR*>, the directory in which you extracted the `ssdcs.zip` file. The command is: `unzip ssdcs.zip`

   Note: Running the unzip in command line mode gives you a list of all the files that are created as part of the installation.
2. Navigate to the `/bin` subdirectory.
3. Run the following command after ensuring that `ant` is in your path:

   ```
   ant -v -f build_deployment.xml -Dappserver_type=<weblogic|websphere|jboss>
   ```

   This command creates an external deployments directory that contains a file named `ssdcs.war`. The `-v` (verbose) option gives you a list of all the files that are created as part of the installation.
4. Deploy the `ssdcs.war` file with the application server.

   Note:
   - When deploying the `ssdcs.war` file, ensure that the environment variable SSDCS_LOG_DIR is set to the absolute path you want the log files to go into.
   - When deploying on IBM WebSphere®, you must provide the context root `/ssdcs` when prompted.
   - The `log4j` JAR file provided in <*SSDCS_DIR*>`/jar` must appear first in the application server classpath; otherwise, errors may occur if your application server uses a different `log4j` version.

Deploying the Sterling Sensitive Data Capture Server as a Demo

You must deploy the Sterling Sensitive Data Capture Server (SSDCS) application in secure mode, otherwise an INVALID_SESSION error occurs when Sterling Sensitive Data Capture Server tries to validate the session token.

If you are deploying Sterling Sensitive Data Capture Server for demonstration purposes only, you can suppress the INVALID_SESSION error. To suppress this error and allow HTTP connections to validate access tokens, you must modify the `web.xml` file in Sterling Selling and Fulfillment Foundation by changing the value from TRUE to FALSE, as follows:

```
<context-param>
   <param-name>scui-access-token-validation-secure</param-name>
   <param-value>FALSE</param-value>
</context-param>
```

Chapter 6. Upgrading the Sterling Sensitive Data Capture Server

About this task

The upgrade process for the Sterling Sensitive Data Capture Server assumes that you have already installed Sterling Selling and Fulfillment Foundation and SSDCS and that you have followed all of the prerequisite requirements in Installing the Sterling Sensitive Data Capture Server.

To upgrade SSDCS:

Procedure

1. Install SSDCS Release 1.1 into a new directory that is different from the SSDCS Release 1.0 directory.
2. Perform a file comparison and merge the files from the Release 1.0/`properties` directory with the Release 1.1/`properties` directory. This will copy your configurations from Release 1.0 to Release 1.1.
3. Copy over the Release 1.0 `jar/extn` folder to replace the Release 1.1 `jar/extn` folder.
4. Configure any additional new functionality as needed.
5. Rebuild the SSDCS WAR file and deploy the SSDCS application. For information about how to deploy SSDCS, refer to Deploying the Sterling Sensitive Data Capture Server.

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 grant 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.

1623-14, Shimotsuruma, Yamato-shi

Kanagawa 242-8502 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 may 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

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

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 measurements 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© IBM 2011. Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2011.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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 http://www.ibm.com/legal/copytrade.shtml.

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

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 registered 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.

Connect Control Center®, Connect:Direct®, Connect:Enterprise®, Gentran®, Gentran®:Basic®, Gentran:Control®, Gentran:Director®, Gentran:Plus®, Gentran:Realtime®, Gentran:Server®, Gentran:Viewpoint®, Sterling Commerce™, Sterling Information Broker®, and Sterling Integrator® are trademarks or registered trademarks of Sterling Commerce™, Inc., an IBM Company.

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

Index

Printed in USA