Sterling Sensitive Data Capture Server, Release 1.0

Configuration Guide

Selling and Fulfillment Foundation, Release 9.0

Last updated in HF15

January 2011

Sterling Commerce
An IBM Company

Notice and Disclaimer:

In this document, the terms "you," "your," or "yours" refer to the end user or the customer. The terms "we," "our," or "ours" refer to Sterling Commerce (America), Inc. ("Sterling Commerce," "Sterling Commerce, Inc.," or "Sterling").

Nothing herein shall be construed as limiting or reducing your obligations to comply with any applicable laws, regulations or industry standards relating to security or otherwise including, but not limited to, PA-DSS and DSS.

The customer may undertake activities that may affect compliance. For this reason, Sterling Commerce, Inc. is required to be specific to only the standard software provided by Sterling.

THE INFORMATION IN THIS DOCUMENT IS FOR INFORMATIONAL PURPOSES ONLY. STERLING COMMERCE MAKES NO REPRESENTATION OR WARRANTY AS TO THE ACCURACY OR THE COMPLETENESS OF THE INFORMATION CONTAINED HEREIN. YOU ACKNOWLEDGE AND AGREE THAT THIS INFORMATION IS PROVIDED TO YOU ON THE CONDITION THAT NEITHER STERLING NOR ANY OF ITS REPRESENTATIVES WILL HAVE ANY LIABILITY IN RESPECT OF, OR AS A RESULT OF, THE USE OF THIS INFORMATION. IN ADDITION, YOU ACKNOWLEDGE AND AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR MAKING YOUR OWN DECISIONS BASED ON THE INFORMATION HEREIN.

STERLING COMMERCE DOES NOT AND CANNOT WARRANT OR REPRESENT THAT IMPLEMENTATION OF THE RECOMMENDATIONS, GUIDELINES, AND/OR DIRECTIONS HEREIN WILL RESULT IN A SECURE SYSTEM. THE INFORMATION HEREIN IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY, AND ARE ONLY SUGGESTIONS: NOTHING HEREIN SHOULD BE CONSTRUED AS PART OF ANY USER OR CUSTOMER GUIDE OR OTHER DOCUMENTATION AGAINST WHICH ANY CONTRACTUAL WARRANTY IS MADE, OR CAN BE CONSTRUED.

THE APPLICABLE STERLING COMMERCE ENTITY RESERVES THE RIGHT TO REVISE THIS PUBLICATION FROM TIME TO TIME AND TO MAKE CHANGES IN THE CONTENT HEREOF WITHOUT THE OBLIGATION TO NOTIFY ANY PERSON OR ENTITY OF SUCH REVISIONS OR CHANGES.

You are responsible for identifying the security options most appropriate to the risks identified in your environment.

This document assumes that you are familiar with security deployment concepts, including knowledge of how to deploy these applications securely. Specifically, you must ensure that the recommendations can be used in your corporate operational environment.

Sterling Commerce strongly recommends that you work with your internal or external security teams from the initial planning stages to production deployment. Most importantly, ensure that engineers knowledgeable in security are involved in designing the system, which includes the Sterling Commerce applications.

Contents

Roadmap: Using the PA-DSS, Secure Deployment, and SSDCS Documentation Guides 5
About the *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide* . . . 6
Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide Revision Information 6
Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide Update History 7
Overview of the Sterling Sensitive Data Capture Server 8
Installing the Sterling Sensitive Data Capture Server 9
Prerequisites 9
Installation Steps 9
Configuring the Sterling Sensitive Data Capture Server 11
Configuring Properties 11
Selling and Fulfillment Foundation Properties 11
SSDCS Properties 12
ESAPI Properties 13
Configuring Logging 13
JSPs 14
Form Fields 15
Validation 18
Tokenization 19
Implementing Custom Code 20
Deploying the Sterling Sensitive Data Capture Server 21

Index **22**

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

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 Sterling Sensitive Data Capture Server (SSDCS) to capture credit card numbers on behalf of the Sterling applications. Doing so ensures that credit card numbers are kept outside of Sterling applications, with the added benefit that these applications are kept outside of PCI DSS auditing scope.

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

- *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide* - Describes the steps that you should 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 applications outside of the PCI DSS auditing scope.
- *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 Sensitive Data Capture Server, Release 1.0: Configuration Guide* (this guide) - Details how to install, configure, and deploy SSDCS as a proxy service that Sterling Commerce applications call to tokenize Primary Account Numbers (PANs) for credit cards and gift value cards.

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

1. Review all three guides in this order:
    a. *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide*
    b. *Selling and Fulfillment Foundation: Secure Deployment Guide*
    c. *Sterling Sensitive Data Capture Server, Release 1.0: Configuration Guide*
2. Implement the steps suggested in the *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide* to remain in compliance with PA-DSS and keep your Sterling applications outside of the PCI DSS auditing scope.
3. Implement the security strategies outlined in the *Selling and Fulfillment Foundation: Secure Deployment Guide*.
4. Install Selling and Fulfillment Foundation and associated applications (refer to the *Selling and Fulfillment Foundation: Installation Guide* and respective application installation guides).
5. Follow the steps in the *Sterling Sensitive Data Capture Server, Release 1.0: Configuration Guide* to configure your SSDCS implementation.

About the *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide*

The *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide* describes the steps that you must follow for your Sterling Sensitive Data Capture Server (SSDCS) installation to remain in compliance with the Payment Application Data Security Standard (PA-DSS). It also describes how to keep the Sterling Commerce applications outside of the PCI Data Security Standard (PCI DSS) auditing scope.

The information in this document is based on the PCI Security Standards Council (PCI SSC) PA-DSS program (version 1.2, dated October, 2008). Sterling Commerce recommends that its customers deploy the SSDCS application in a manner that adheres to the PCI DSS and the PCI PA-DSS (version 1.2).

Subsequent to this, best practices and hardening methods such as those referenced by the Center for Internet Security (CIS), including their various benchmarks, should be followed to enhance system logging, reduce the chance of intrusion, and increase the ability to detect intrusion. Other general recommendations to secure networking environments should be followed, as well. Such methods include, but are not limited to, enabling operating system auditing subsystems, system logging of individual servers to a centralized logging server, disabling of infrequently used or frequently vulnerable networking protocols, and the implementation of certificate-based protocols for access to servers by users and vendors.

Note: If you do not follow the steps outlined here, your Sensitive Data Capture Server installations will not be PA-DSS compliant and the Sterling applications could be considered to be within PCI DSS auditing scope.

Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide Revision Information

Revision Information	
Author	Bernie Wong, Performance Engineering Director
Approving Authority	Steven Aulds, Senior Vice President, Engineering
Revision Date	March 31, 2010
Next Review Date	March 31, 2011 or whenever the underlying application changes or whenever the PA-DSS requirements change
Exclusions	Applies to all Engineering employees who develop or maintain the SSDCS
Standard Number	ESP201

Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide Update History

Name	Title	Date	Summary of Changes
Bernie Wong	Performance Engineering Director	Dec 14, 2009	Initial version

Note: The *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide* will be reviewed on a yearly basis, whenever the underlying application changes or whenever the PA-DSS requirements change. Updates will be tracked and reasonable accommodations will be made to distribute the updated guide to users.

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 Selling and Fulfillment Foundation User Interface. No APIs were enhanced for tokenization. A token should be passed to the API instead of passing a payment application number (PAN).

For information about how to configure SSDCS securely, refer to the *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide*.

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

Installing the Sterling Sensitive Data Capture Server

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

Note: <*INSTALL_DIR*> refers to the directory in which you have installed Selling and Fulfillment Foundation.

Prerequisites

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

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

Note: Before installing the Sterling Sensitive Data Capture Server (SSDCS) application, you must read the *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide* for information about how to configure SSDCS securely.

Installation Steps

After you have installed Selling and Fulfillment Foundation, perform the following steps to install the SSDCS application:

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 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 *Selling and Fulfillment Foundation: Properties Guide*.

3. Navigate to the <*INSTALL_DIR*>/repository/external directory and locate the ssdcs.zip file. This zip 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, unzip 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.
- `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.
- `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.

Configuring the Sterling Sensitive Data Capture Server

This section describes how to configure the Sterling Sensitive Data Capture Server (SSDCS). It contains the following information:

- Configuring Properties
- Configuring Logging
- JSPs
- Implementing Custom Code

For information about configuring SSDCS securely, refer to the *Sterling Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide.*

Configuring Properties

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

Selling and Fulfillment Foundation Properties

Use the `customer_overrides.properties` file to configure the following 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 *Selling and Fulfillment Foundation: Properties Guide*.

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 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://sterlingcommerce.com:8080/smcfs/accessTokenServlet`
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 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*.	
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 = com.stercomm.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=com.stercomm.ssdcs.ue.ISSDCSTokenizeImpl`

ESAPI Properties

You can use the ESAPI.properites 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 Validation.

Configuring Logging

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.

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.

  Notes:

  - 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.

- `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.

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

Property	Description
<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.

JSPs

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.

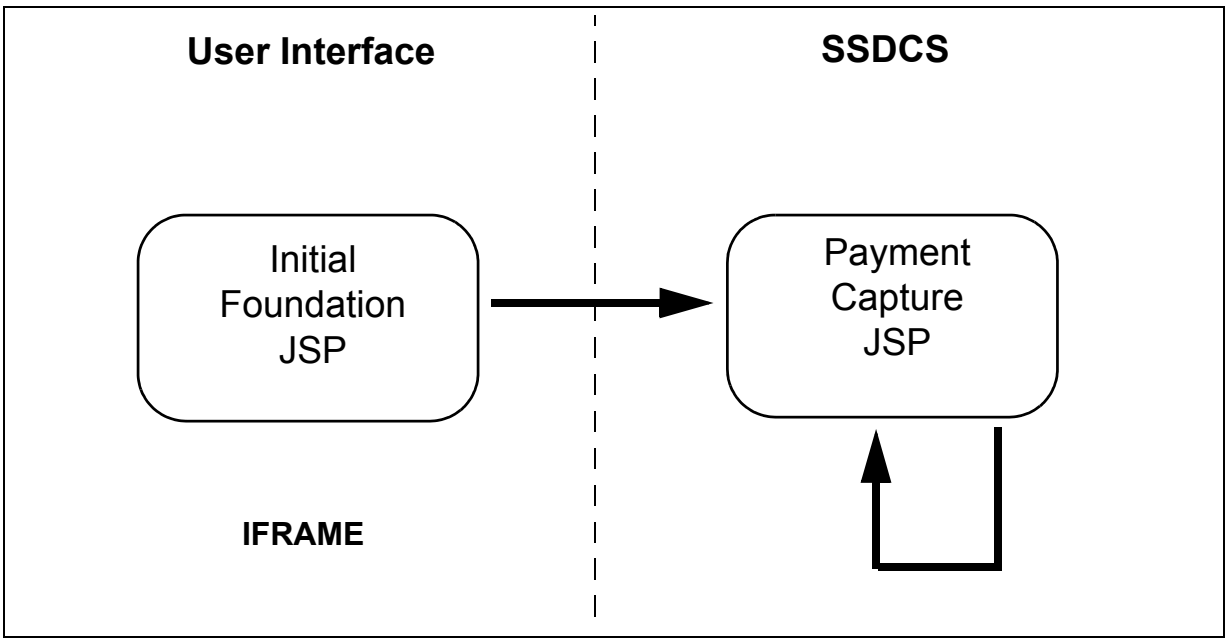
The following process occurs when a credit card number or stored value card number is entered in the text field of the JSP:

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 and Validation.
2. SSDCS validates whether the session token passed to it from Selling and Fulfillment Foundation is a valid session. If the session is not valid, 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 Validation.
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.

Form Fields

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	Passthrough
ssdcsAuthenticationToken	The token that is validated against 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`. This should be passed as a relative path to the context root, for example, `/jsp/ssdcs_tokenize_pan.jsp`.	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 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
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

Field Name	Description		In Title	Passthrough
ssdcsResultCode	The result of the servlet invocation:		Yes	No
	<BLANK>	Not a returnable status. It is used during initial form loading to understand that the only processing performed is authentication.		
	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.		
ssdcsFailReason	Explains why the tokenization fails:		Yes	No
	INVALID_SESSION	Authentication with 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.		
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
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.

Validation

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.
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 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 *Selling and Fulfillment Foundation Javadocs*. If the session is not valid, the 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 Sterling payment type 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. Sterling Commerce recommends that if the payment type is an unknown type, you revert to a default logic that will still provide validation.

Tokenization

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.

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 Sterling payment type 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. Sterling Commerce recommends that if the payment type is an unknown type, you revert to a default logic that will still provide tokenization.

Implementing 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 Sensitive Data Capture Server, Release 1.0: PA-DSS Implementation Guide*.

Deploying the Sterling Sensitive Data Capture Server

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.

Note: 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 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>
```

To deploy the Sterling Sensitive Data Capture Server application:

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

   ```
   unzip ssdcs.zip
   ```

   Note: Running the unzip in command line mode displays 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`.

   Note: The `-v` (verbose) option displays 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.

   Notes:

   - 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 appserver classpath; otherwise, errors may occur if your appserver uses a different `log4j` version.

Index

C

Center for Internet Security (CIS) 6

configuring
  logging 13
  properties 11

credit card validation 19

custom code, implementing 19

E

ESAPI properties 13

F

form fields 15

I

input validation 18

INVALID_SESSION error 19

J

JSPs, overview 14

L

log files
  ssdcs.log 13
  ssdcs_esapi.log 13
  ssdcs_security.log 13

log4j configuration file 13

logging, configuring 13

P

PA-DSS, definition 8

Payment Card Industry Data Security Standard (PCI DSS)
  auditing scope 6

Payment Card Industry Payment Application - Data Security Standard (PCI PA-DSS)
  compliance 6

PCI-DSS, definition 8

properties
  configuring 11
  ESAPI 13
  Selling and Fulfillment Foundation 11
  SSDCS 12

R

revision information 6

S

sc.access.token.expire.in.seconds 11

sc.access.token.max.allowed.expire.in.seconds 11

security
  general recommendations 6

Selling and Fulfillment Foundation properties 11

session validation 19

SSDCS
  configuration 11
  deploying 21
  installation 9
  overview 8
  properties 12

ssdcs.log 13

ssdcs.smcfs.url property 12

ssdcs.ue.PanValidator user exit 12

ssdcs.ue.Tokenize user exit 12

ssdcs.zip
  contents 9
  location 9

ssdcs_esapi.log 13

SSDCS_LOG_DIR variable 10
ssdcs_security.log 13
stored value card validation 19

T

tokenization 8, 19

U

update history 7
user exits
  ssdcs.ue.PanValidator 12
  ssdcs.ue.Tokenize 12

V

validation
  credit card validation 19
  input validation 18
  session validation 19
  stored value card validation 19

Y

yfs.ssdcs.jsp property 11
yfs.ssdcs.servlet property 11
yfs.ssdcs.tokenize.cc property 11
yfs.ssdcs.tokenize.svc property 11
yfs.ssdcs.url property 11