Sterling Selling and Fulfillment Foundation

Secure Deployment Guide

Version 9.1.0.24

Sterling Selling and Fulfillment Foundation

IBM

Secure Deployment Guide

Version 9.1.0.24

Note
Before using this information and the product it supports, read the information in “Notices” on page 51.

Contents

Chapter 1. Introduction

Introduction

This guide provides security recommendations to guide your deployment of the IBM® Sterling Selling and Fulfillment Suite. The Suite includes the following applications (the " IBM Software"):

- IBM Sterling Distributed Order Management
- IBM Sterling Web
- IBM Sterling Warehouse Management System
- IBM Sterling Business Center
- IBM Sterling Field Sales
- IBM Sterling Call Center and IBM Sterling Store

Document Scope and Disclaimer

In this document, the terms "you," "your" or "yours" refers to the end-user or the customer. The terms "we," "our" or "ours" refers to International Business Machines Corp. (IBM).

This document provides recommendations to guide your secure deployment of IBM Sterling Selling and Fulfillment Suite. applications. Where applicable, it also documents a sample third party security configuration for use with the IBM applications. These sample configurations are not meant to be exhaustive. In some cases, there will be multiple implementation alternatives, each with its own security strengths and weaknesses. 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.

IBM strongly recommends 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 theIBM applications.

Roadmap: Using the PA-DSS, Secure Deployment, and IBM Sterling Sensitive Data Capture Server Documentation Guides

The following guides in theIBM 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,*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). The Implementation Guide describes the steps that you should follow for your Sterling Sensitive Data Capture Serverinstallation 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 Sterling Sensitive Data Capture Server. This guide explains how to keep theIBM applications outside of the PCI DSS auditing scope.

- *Sterling Selling and Fulfillment Foundation: Secure Deployment Guide* (this guide) — Explains how to deploy theSterling 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 Sterling Sensitive Data Capture Server as a proxy service that IBM applications call to tokenize Primary Account Numbers (PANs) for credit cards and gift value cards.

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

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

Chapter 2. Security Planning

Security Planning

Secure systems do not occur by chance — they have to be designed and built in.

Securing systems involves many detailed and sometimes complex steps. Systems may have to comply with strict security compliance regulations or standards. Obtaining security compliance could take months. Integrating to secured systems may require program changes. Secured networks could impede system data flows. Systems may have to be audited before they are implemented into production.

Assuming your system will be secure and will work as expected when security controls are applied in production WILL result in unscheduled redesign or rework and project delay. Not planning or considering security prior to production or "go live" will add significant risks to your project.

Scope and Caveat

We, however, want to make it absolutely clear that architecting, designing and ensuring the overall security of your system remains your responsibility. The recommendations in this document are applicable to theIBM applications. They do not take into account how the applications fit within your corporate operational environment. They may not apply to your custom code or third party components. They also do not take into account your overall security risk and threat landscape.

Further, 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, security regulations such as PA-DSS and DSS, if applicable. You may undertake activities that may affect compliance. For this reason, IBM is required to be specific only to the standard software that it provides.

Security Engineering Steps

The key message is to start your security engineering early. Start your security planning early. Secure your systems early. Develop your system with security early. Test your system early.

As the security architect, we propose you consider the following steps:

1. Work with the project's implementation team to understand the solution or system architecture.
2. Identify your security requirements including determining all security standards, laws, and/or regulations with which your system has to comply.
3. Implement your system with security features and controls enabled.
4. Ensure that all software components are integrated.
5. Develop and run end-to-end tests.
6. Based on your experience with the baseline system, develop your network deployment strategy.
7. Re-implement the system on the appropriate network zones.

8. Rerun your test cases.

These actions begin with reading, understanding, and applying the recommendations found in this guide. These recommendations are specific to theSterling Selling and Fulfillment Suite applications. Your responsibility, however, does not end here. Your system has to fit within your overall corporate operational environment. You may have very specific security requirements. As a result, YOU have the responsibility to ensure that YOUR system is secure.

Consider the following steps at a minimum:
- Understand Your System Architecture
- Build Out Your System
- Build End-to-End Test Cases
- Harden the Infrastructure and the Sterling Selling and Fulfillment Suite
- Design Your Deployment Strategy

Understand Your System Architecture

As early as possible, possibly as early as your solution definition phase, obtain and understand your system architecture. This architecture should tell you the following:
- What are the applications and software components used in the system?
- How are the applications and components connected or integrated together?
- What are the types and the sensitivity of data stored in the system?
- Who are the end-users of the system and where are they located?
- What are the expected use case scenarios?
- What are the non-functional requirements (NFRs)?

Knowing all the software components and how they are integrated or connected is crucial when you plan out your network architecture (especially the firewall and network security zone) as well as in determining what security countermeasures are needed. Knowing the type of data and their sensitivity will tell you what you have to secure and how much effort it will take.

Developing the system architecture is beyond the scope of this document. We strongly urge you, as the security professional, to seek out the system architecture from your implementation team and understand it.

As part of defining the systems architecture, you need to identify your security requirements. This could range from identifying security regulations and standards your system has to comply with to developing your security threat model.

For example, applications in your system will have to comply with the Payment Card Industry (PCI) security standards if they capture, store, process or transmit credit card. If your system accepts credit cards, we strongly urge you to read the *PCI PA-DSS Strategy* section of this document to understand the strategy around PCI DSS and PCI PA-DSS compliance.

Depending on the type of data you process or the environment in which you work, you may have to consider other regulatory requirements. For example, government projects may have to comply with Federal Information Security Management Act (FISMA). Systems that capture health information may have to comply with Health Insurance Portability and Accountability Act (HIPAA).

Understanding and building your systems to be compliant will be significantly easier than trying to retrofit and address these security concerns close to your production “go live.”

In most cases, you should also plan time and resources in your project to include an audit of the system prior to production.

Also as part of your system definition, you should perform a security threat model for your system to understand, at a minimum, potential attackers and what assets they might be interested in; and as a result, the threats for which you need to build countermeasures. For example, knowing that remote attackers may want to anonymously browse your interoperability message queues should dictate specifying countermeasures such as message queue authentication. You should ensure all security requirements are factored into your system architecture early in your project lifecycle.

Build Out Your System

Given a solid understanding of your system architecture and the security requirements, you should build out your systems with all the required security features enabled as early in the project as possible. For example, your operational infrastructure, which includes your operating systems, databases, network and applications servers, should be hardened to industry best practices or recommendations. Each integration touchpoint should be secured. For example, you should ensure that systems or programs that interact with each other through various integration technologies, such as message queues or Web services, do so with all necessary security controls enabled.

Mandating the enabling of security controls into your development and testing environments means that your system will be tested. Do not conduct all your testing without security and then expect the system to work when you turn on security during your production “go live.”

Given the complexity of the systems, you may want to take an incremental approach to building out your system. One approach is to first build out your system on a flat network where every application and component is placed in a simple single network segment. The goal of this exercise is to build up your experience in installing, configuring, and running the applications as an integrated system. You should ensure that all system components and applications are installed and integrated. In the early project phases, you may have to create some test programs or stubs to fulfill the integration. Implementing the system on a flat network means you do not have to worry about network connectivity.

Build End-to-End Test Cases

In parallel, you should also identify end-to-end test cases for the system. The goal of these test cases is to test data entering and flowing through your system. These test cases should include every integrated application. For example, one end-to-end test case could be entering or capturing orders in the IBM Sterling Web application. As part of the order capture, the test would call systems to validate and authorize payment, and possibly to conduct fraud check, validate shipping addresses, and so on. Next, the test could move the order to theIBM Sterling Distributed Order Management and the IBM Sterling Warehouse Management System for fulfillment.

A significant component of this activity will involve defining the test cases, developing the test data and configuring the applications to support the tests, and writing stub code to implement the necessary integration. The benefit in investing in this early is that the tests can be used to validate your system architecture. More importantly, these test cases can be reused as you start to incrementally harden or secure your system. Knowing that functionality worked in your baseline environment, and not in the subsequently hardened system, will help you isolate the root cause. The converse means that running these tests the first time on a hardened system will make problem diagnosis much harder.

Harden the Infrastructure and theSterling Selling and Fulfillment Suite Applications

After the successful completion of your end-to-end tests, you should next harden your infrastructure starting with the operating systems, database, and application servers. When that is completed, we recommend that you harden your network. We strongly encourage you to apply the recommendations found in the *Securing the Infrastructure* section of this document.

You should also lock down access to all the integration touchpoints into the Sterling Selling and Fulfillment Suite applications. This includes, at a minimum, all your integration calls to the Interoperability Servlet, Web Services, Enterprise JavaBeans (EJB), and so on. This is described in the *Securing Sterling Selling and Fulfillment Suite* section of this document.

You will have to change your test programs as you apply the hardening recommendations. For example, when you harden or secure the Java Messaging Services as dictated in the *JMS Services Security* and *Message Queue Security* sections, you will have to modify your test programs to authenticate into the queues.

After you have applied the hardening recommendations, you should rerun your end-to-end test cases.

You should use the knowledge and experience gained from this first exercise to validate the system architecture and to document the integration architecture. Given these two artifacts, your next step should be to build out your network architecture. This is where a good understanding of all the software components, the integration data flows, and the nonfunctional requirements (such as expected transaction volumes) will be invaluable.

Design Your Deployment Strategy

At the successful completion of this end-to-end test on a hardened infrastructure on a fully integrated system, you are now ready to design and then deploy your system into secure network architecture. The key to the hardened network is the development of your network architecture (specifically, network security zones) and determining which software components goes into which zone.

Network zones are used to allow authorized network communication to support your business needs and at the same time restrict all other unauthorized network access. See Network Security for more detail. See the *Deployment Scenarios* section of this document for guidance on planning your deployment strategy.

Chapter 3. PCI PA-DSS Strategy

PCI PA-DSS Strategy Introduction

Applicability

This strategy only applies to you if you plan to capture, process, store, or transmit credit card information in any of the Sterling Selling and Fulfillment Suite applications.

PCI DSS and PCI PA-DSS Standards

Organizations that handle credit cards should already be familiar with the PCI DSS because compliance to PCI DSS is mandatory. At a high level, the PCI DSS standard mandates how payment applications (both commercial off-the-shelf and custom built) are deployed into your operational environment. Organizations that fall into this category have to be PCI DSS audited, at a minimum.

The PCI PA-DSS standard is relatively new and applies to software vendors like IBM. It governs the steps that commercial off-the-shelf payment applications must adhere to, such as documentation, developer training, and patching. Customers should be aware of PCI PA-DSS; for example, Visa® has mandated that merchants should only use PA-DSS validated applications after July 1, 2010.

PCI PA-DSS Key Concepts

The IBM PCI PA-DSS strategy is based on two key concepts. The first is the use of a well-known and accepted practice called "tokenization" to transform sensitive credit card Primary Account Numbers (PANs) into unique and cryptographically strong random string tokens, storing PANs only in your central corporate credit card vault and storing tokens only in the Sterling Selling and Fulfillment Suite applications.

The second key concept is to architect the Sterling Selling and Fulfillment Suite applications in such a way that PANs are captured through a proxy service called the Sterling Sensitive Data Capture Server, and not by theIBM applications.

With these two concepts, you should be able to offload PCI DSS security concerns surrounding the handling of PAN such as encryption, secure PAN storage, key management, and logging to your central vault. In addition, by keeping PAN outside of the Sterling Selling and Fulfillment Suite applications, you should be able to work with your PCI DSS QSA to keep the IBM applications outside of PCI DSS auditing scope.

The*Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* and the*Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* provide more detail on this strategy.

Customer Responsibility

To use the IBM PA-DSS strategy, you, the customer, must comply with the following requirements.

Adherence to the Sterling Sensitive Data Capture Server PA-DSS Implementation Guide

You must, as required by PCI, read and understand the*Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* and implement theSterling Sensitive Data Capture Server in accordance to its instructions. If you do not follow the steps outlined in the *Implementation Guide*, yourSterling Sensitive Data Capture Server installation could become non-compliant with PA-DSS, which could place the Sterling Selling and Fulfillment Suiteapplications within PCI DSS auditing scope.

The *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* and this *Secure Deployment Guide* document the configuration of one reference implementation. Your system will likely be different. You should be able to maintain the Sterling Sensitive Data Capture Server's PA-DSS compliance if your DSS Qualified Security Assessor (QSA) approves the variances from the *Implementation Guide*.

Credit Card Vault

You, the customer, are expected to provide the credit card vault. We do not provide a credit card vault because vaults are typically centralized corporate assets rather than resources implemented by projects or departments.

Internally, the PCI PA-DSS Strategy was tested with the nuBridges Protect™ credit card vault (see http://nubridges.com/solutions/tokenization/).

Payment Abstraction Layer

Our PA-DSS strategy relies on the fact that sensitive Primary Account Numbers (PANs) are stored in your credit card vault and its surrogate token stored in the Sterling Selling and Fulfillment Suite applications. When those applications need clear text for processing, we expect you, the customer, to convert the tokens to clear text (which we will refer to as detokenization) outside of theSterling Selling and Fulfillment Suite applications.

For example, prior to tokenization, theIBM payments agents were typically configured to send PAN directly to payment gateways for authorization. With our tokenization strategy, the Sterling Selling and Fulfillment Suite applications only store tokens and never the PAN (either in encrypted or cleartext form). To facilitate payment transactions that require the clear text PAN, we rely on you to detokenize outside of the Sterling Selling and Fulfillment Suite applications.

Typically, customers would implement a proxy service, which we will refer to as a Payment Abstraction Layer (PAL). The PAL will receive payment transactions, de-tokenize the token, replace the token with the clear PAN, then proxy the request to the payment gateway. The major benefit of performing the detokenization in the PAL is to keep theSterling Selling and Fulfillment Suite applications outside of PCI DSS auditing scope.

Upgrading from Previous Application Versions

If you are upgrading from previous versions of theSterling Selling and Fulfillment Suite applications that captured and stored PAN, you must consider the following:

- During the upgrade, you must tokenize PAN (even if they were stored encrypted) so that theSterling Selling and Fulfillment Suite applications only store PAN. PCI PA-DSS mandates this. Please see the *Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* for more guidance.

- The use of facilities in the Sterling Selling and Fulfillment Suite applications previously used to handle credit card information (for example, APIs used to encrypt credit cards) has been deprecated. You should not use these facilities if you want to maintain the PA-DSS validation for the Sterling Sensitive Data Capture Server .

Chapter 4. Deployment Scenarios

Deployment Scenario Overview

How you deploy your applications in the network security zones depends upon the applications you need to install. Answering the following questions will refer you to deployment scenarios for your applications.

1. Do any of these applications need to capture credit card numbers?

   Your application—and, as a result, your system—have to comply to PCI DSS security standards if they capture, process, store or transmit credit card information. If this applies to you, you should read IBM PCI PA-DSS Strategy. The Sterling Selling and Fulfillment Suite applications come with a software component called theSterling Sensitive Data Capture Server that was purposely built to keep credit card data outside of theSterling Selling and Fulfillment Suite applications. The benefit of this approach is to potentially isolate the PCI auditing scope to the Sterling Sensitive Data Capture Server component only, outside of the Sterling Selling and Fulfillment Suite applications.

   This strategy relies on you, the customer, to provide the following:

   - Credit card vault
   - Payment Abstraction Layer (PAL)
2. If you are implementing theSterling Web application only, see the topic Sterling Web" Only Deployment Scenario."
3. If you are implementing theSterling Distributed Order Management application only, or with optional components such as Sterling Business Intelligence, Sterling Warehouse Management System, Sterling Call Centerand Sterling Store, see Sterling Distributed Order Management with Optional Applications Deployment Scenario.
4. If you are implementingSterling Web and the Sterling Distributed Order Management applications in a combined integrated fashion, see Sterling Distributed Order Management and Sterling Web Deployment Scenario."

Note that these combinations are not exhaustive. For more information on how Sterling Selling and Fulfillment Suite applications can be combined, refer to the *Product Overview* guide.

Securing the Infrastructure

Infrastructure Security Introduction

Infrastructure hardening is a vital step in making your operational environment and system more secure. The hardening process, especially for systems as complex as an operating system, database management systems, or application servers, involve many small but detailed steps. These steps range from removing or disabling unneeded or insecure services and restricting access to critical resources to applying the latest software patches.

Since the infrastructure components are not IBM products, we defer to industry best practices, especially when the guidance and recommendations are open, consensus developed, and peer reviewed. Fortunately, the Center for Internet Security (CIS) and the Defense Information Systems Agency (DISA) have developed comprehensive hardening checklists.

Operating System Security

The Center for Internet Security (CIS) has produced hardening checklists or benchmarks that are specific to each operating system. The CIS also produces tools to help assess the compliance of systems to CIS benchmarks.

We strongly recommend that you review and consider the hardening guidelines found in the CIS Benchmarks. The choices you make should be in line with your corporate standards.

Database Management System Security

The Center for Internet Security (CIS) has produced hardening checklists or benchmarks that are specific to each database management system.

We strongly recommend you review and consider the hardening guidelines found in the CIS Benchmarks. The choices you make should be in line with your corporate standards.

Caveats for DB2® for Linux, UNIX, and Windows Hardening Recommendations

The following are caveats or deviations from the recommendations in CIS IBM DB2 Benchmark v1.1.0 that was released on December 31, 2009.

In Recommendation 3.1.14 — Set maximum connection limits, the CIS recommends setting the parameters **max_coordagents** and **max_connections** to 100. Depending on how many IBM agents and application servers you start up, you may need more. Consult the for steps to estimate your connection requirements.

In Chapter 4, the CIS recommends the use of Label-Based Access Control (LBAC), which is a separately licensed DB2 component. The Sterling Selling and Fulfillment Suite applications are not certified for use with LBAC.

Caveats for Oracle Hardening Recommendations

The following are caveats or deviations from the recommendations in CIS Oracle Database 11g Benchmark v1.0.1 that was released on January 10, 2010.

Some of the hardening recommendations (such as 8.09 — Set CPU_PER_SESSION as appropriate and 8.11 — Set LOGICAL_READS_ PER_SESSION as appropriate) restrict the amount of CPU and I/O resources that a session or a user can consume. Setting the limits too low can cause application transactions to be abnormally terminated. Please use these controls with caution.

Some of the recommendations (such as 8.13 — Set CONNECT_TIME as appropriate and 8.14 — Set IDLE_TIME as appropriate) limit the amount of time that connections are opened, or the amount of time connections are idle. Setting these thresholds could cause Oracle to close connections that are managed by the application's connection pools. Please consult the*Sterling Selling and Fulfillment Foundation: Performance Management Guide* for more information on how connections inSterling Selling and Fulfillment Suite applications are managed by connection pools.

In Requirement 14.01, the CIS recommends enabling and applying Oracle Label Security (OLS), which is a separately licensed Oracle component. The Sterling Selling and Fulfillment Suite applications are not certified for use with OLS.

Application Server Security

Application Server Security Introduction: The application server is a critical system component upon which theSterling Selling and Fulfillment Suite applications run. Unfortunately, unlike the operating systems and database management systems, the Center for Internet Security (CIS) currently does not have hardening benchmarks for application servers such as Oracle WebLogic or IBM® JBoss. Security Technical Implementation Guides (STIG) or hardening checklists produced by the Defense Information Systems Agency (DISA) are written for a generic application server. Given the lack of consensus-built and publicly peer reviewed hardening checklists,IBM at this time recommends that you work with your application server vendor for the overall hardening recommendations.

The following topics provide guidance on how to secure specific critical resources in application servers upon which Sterling Selling and Fulfillment Suite applications rely:
- “Securing Access to Java Messaging Services (JMS)”
- “Securing Oracle WebLogic Messaging” on page 14
- “Securing JBoss Messaging” on page 14

JMS Services Security:

Securing Access to Java Messaging Services (JMS): TheSterling Selling and Fulfillment Suite applications use Java Messaging Services (JMS) for many reasons, including:
- Intermediate message queue for interoperability or integration reasons. For example, Sterling Distributed Order Managementcould send orders, as messages, to an integration queue to an external system for processing. Similarly, a legacy order capture system could send orders toSterling Selling and Fulfillment Suite for order fulfillment.
- Work-in-progress messages that are used byIBM Sterling B2B Integrator servers. The fulfillment application uses these agent queues as a means to store and dispatch task messages to agent processes.

By default, JMS vendors create queues to allow complete unrestricted access. As a result, by default, unauthenticated attackers can, at a minimum, do the following:
- Browse messages in queues, which could lead to loss of confidentiality
- Inject new messages or alter existing messages, which could lead to loss of integrity
- Flood message queues with bogus messages, which could lead to loss of availability

Depending on your security threat models, you may want to secure your message queues so that only authenticated users can access the queues and messages in flight are encrypted to ensure confidentiality.

The following sections provide sample configuration guidance on securing these application server-based JMS resources:
- “Securing Oracle WebLogic Messaging” on page 14
- “Securing JBoss Messaging” on page 14

Note: See “Securing TIBCO Messaging” on page 19 and “Securing IBM® WebSphere® MQ Messaging” on page 21 for sample configuration guidance for those standalone systems.

Client Side Configuration

After you have secured your JMS resources, see “Securing JMS Integration” on page 44 for instructions on how to configure your IBMapplication to access those resources.

Securing Oracle WebLogic Messaging:

Note: The sample configuration provided below, using Oracle WebLogic 10.3.2, is an example for illustration purposes only. The sample is by no means exhaustive or optimal. There are other approaches to securing message queues, each with their own strengths and weaknesses. Since message queuing software is provided by third party vendors, we strongly encourage you to discuss your approach to hardening or securing message queues with Oracle WebLogic messaging specialists. Products can evolve; hence, processes for hardening them might change over time. If you have any problems with the processes discussed here, we ask you work with Oracle directly. Use the following recommendations as a starting point and customize the configuration for your specific operating environment.

By default, the Oracle WebLogic JNDI and JMS resources are owned and managed by an “everyone” Group to which anonymous users (Anonymous Roles) are assigned. As a result, anonymous users are, by default, able to access JNDI and JMS resources.

To secure access to the WebLogic JMS, you need to do the following:
- Define a user and group that will be given access rights to the JMS
- Add that user to the JMS Roles and Policies

To perform these functions in Oracle WebLogic 10.3.2, navigate to your WebLogic Administration Console and perform the following steps:
1. Select the tab **Users and Groups**. Create a group and add users to the group.
2. Go to **Services > Messaging > JMS Servers** and create a new JMS Server.
3. Go to **Services > Messaging > JMS Modules** and create a new JMS Module.
4. Click on the newly created JMS Module and click on the **New** button. Create a Connection Factory.
5. Repeat the step given above to create a Queue.
6. Go to the queue **Security** tab.
7. Add Group as **JMS Queue Scoped Roles**.
8. Under **Policies**, add the user.

This restricts queue access to your defined users and groups.

For more details, refer to the *WebLogic Administration and Console Guide*, especially the section "Securing WebLogic Resources Using Roles and Policies."

Securing JBoss Messaging:

Note: The sample configuration provided, using JBoss Messaging version 4.3, is an example for illustration purposes only. The sample is by no means exhaustive or optimal. There are other approaches to securing message queue, each with its own strengths and weaknesses. Since message queueing software is provided by third party vendors, we strongly encourage you to discuss your approach to hardening or securing message queues with JBoss messaging specialists. Products can evolve; hence, processes for hardening them might change over time. If you have any problems with the processes discussed here, we ask you work with JBoss directly. You should use the following recommendations as a starting point and customize the configuration for your specific operating environment.

Configuring JBoss Queue Security

JBoss has two different JMS options. In version 4.2 of JBoss, the server software shipped with a JMS feature known as JBoss MQ. However, in Release 4.3, the JMS offering was revamped to improve performance and to provide additional features. The new JMS feature is called JBoss Messaging.

Note: This document assumes you have installed the JBoss Messaging package. These instructions will not work with JBoss MQ.

Install JBoss Messaging with the following ant release script to create a queue called testQueue. You can look at `<`*`JBOSS_HOME`*`>/server/messaging/deploy/jboss-messaging.sar/destination-service.xml` to verify that the messaging installation was successful.

```
<mbean code="org.jboss.jms.server.destination.QueueService"
  name="jboss.messaging.destination:service=Queue,name=testQueue" xmbean-
  dd="xmdesc/Queue-xmbean.xml">
  <depends optional-attribute-
    name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
  <depends>jboss.messaging:service=PostOffice</depends>
  <attribute name="SecurityConfig">
    <security>
      <role name="guest" read="true" write="true"/>
      <role name="publisher" read="true" write="true" create="false"/>
      <role name="noacc" read="false" write="false" create="false"/>
    </security>
  </attribute>
</mbean>
```

If you are using the out-of-the-box JBOSS configuration that was installed from these steps, you can start the new server:

```
<JBOSS_HOME>/bin/run.sh -b 0 -c messaging
```

Configuring Queue Security in JBoss

As shown in the XML snippet above, the destination-service.xml Queue definition contains a SecurityConfig element that defines what activities can be performed by a role or group. The users and groups are defined in `<JBOSS_HOME>/server/messaging/conf/props/messaging-roles.properties`.

Note: If you chose to configure messaging for a different server name, replace "messaging" in the user and role property file paths with the name of the server you configured.

The following is an example:

```
#
# user=role1,role2,...
guest=guest
vinay=vinay
noacc=noacc
```

The user and passwords are defined in <*JBOSS_HOME*>/server/messaging/conf/props/messaging-users.properties:

```
#
# user=password
guest=guest
vinay=vinay
noacc=noacc
```

At this point, you have defined user authentication to the JBoss queues.

Configuring JBoss Messaging to Use SSL

The first step is to set up the keystore. The messaging package includes a sample keystore and truststore. You can use this keystore or you can create your own. In either case, copy the keystore to the <*JBOSS_HOME*>/server/<*servername*>/conf/ directory.

These will be referenced from the SSL Bisocket configuration xml (see *Enabling the SSL Bisocket Transport* in this topic).

Adding a Secure Connection Factory

In the directory <*JBOSS_HOME*>/server/<*servername*>/deploy/, create a new xml file called messaging-secure-socket-service.xml. A sample of this file is included with the JBoss messaging installation package in the location examples/secure-socket/etc/messaging-secure-socket-service.xml.

This file should have the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Secure Socket Transport Example: the deployment descriptor for the
secure socket factory service, secure connector and secure connection factory.
$Id: messaging-secure-socket-service.xml 2773 2007-06-12 13:31:30Z <cvs_login> $
 -->
<server>
  <mbean code="org.jboss.jms.server.connectionfactory.ConnectionFactory"
    name="jboss.messaging.destination:service=SecureConnectionFactory" xmbean-
    dd="xmdesc/ConnectionFactory-xmbean.xml">
  <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
  <depends optional-attribute-
name="Connector">jboss.messaging:service=Connector,transport=sslbisocke
  t</depends>
  <attribute name="JNDIBindings"> <bindings>
  <binding>/SecureConnectionFactory</binding> </bindings>
  </attribute>
  </mbean>
</server>
```

Enabling the SSL Bisocket Transport

In the directory <*JBOSS_HOME*>/server/<*servername*>/deploy/, create a new xml file called remoting-sslbisocket-service.xml. A sample of this file is included with the JBoss messaging installation package in the location examples/config/remoting-sslbisocket-service.xml.

In the event that you are using your own keystore with a different filename or password, edit the file to point to the keystore and password you have chosen. The path of the keystore appears to be relative to the <*JBOSS_HOME*>/server/<*servername*>/conf directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  The deployment descriptor for the secure socket factory service and secure
    connector.
    $Id: remoting-sslbisocket-service.xml 3409 2007-12-04 21:32:54Z <cvs_login> $
-->
<server>
  <mbean code="org.jboss.remoting.transport.Connector"
    name="jboss.messaging:service=Connector,transport=sslbisocket" display-
    name="SSL Bisocket Transport Connector">
  <attribute name="Configuration">
    <config>
      <invoker transport="sslbisocket">
        <!-- There should be no reason to change these parameters - warning!
          Changing them may stop JBoss Messaging working correctly -->
  <attribute name="marshaller"
    isParam="true">org.jboss.jms.wireformat.JMSWireFormat</attribute>
  <attribute name="unmarshaller"
    isParam="true">org.jboss.jms.wireformat.JMSWireFormat</attribute>
  <attribute name="dataType" isParam="true">jms</attribute>
  <attribute name="socket.check_connection" isParam="true">false</attribute>
  <attribute name="timeout" isParam="true">0</attribute>
  <attribute
  name="serverBindAddress">${jboss.bind.address}</attribute>
  <attribute name="serverBindPort">5457</attribute>
  <attribute name="clientSocketClass"
    isParam="true">org.jboss.jms.client.remoting.ClientSocketWrapper</attribu
    te>
  <attribute
    name="serverSocketClass">org.jboss.jms.server.remoting.ServerSocketWrap
    per</attribute>
  <attribute
    name="serverSocketFactory">jboss.messaging:service=ServerSocketFactory,
    type=SSL</attribute>
  <attribute name="numberOfCallRetries" isParam="true">1</attribute>
  <attribute name="pingFrequency" isParam="true">214748364</attribute>
  <attribute name="pingWindowFactor"
    isParam="true">10</attribute>
  <attribute
    name="onewayThreadPool">org.jboss.jms.server.remoting.DirectThreadPool<
    /attribute>
  <!-- End immutable parameters -->

  <!-- Periodicity of client pings. Server window by default is twice this
    figure -->
  <attribute name="clientLeasePeriod" isParam="true">10000</attribute>
  <!-- Number of seconds to wait for a connection in the client pool to
    become free -->
  <attribute name="numberOfRetries" isParam="true">10</attribute>
  <!-- Max Number of connections in client pool. This should be
    significantly higher than
  the max number of sessions/consumers you expect -->
  <attribute name="JBM_clientMaxPoolSize" isParam="true">200</attribute>
  <!-- Use these parameters to specify values for binding and connecting
    control connections to work with your firewall/NAT configuration
  <attribute name="secondaryBindPort">xyz</attribute> <attribute
    name="secondaryConnectPort">abc</attribute> -->
 </invoker>
 <handlers>
    <handler
      subsystem="JMS">org.jboss.jms.server.remoting.JMSServerInvocationHandle
      r</handler>
    </handlers>
    </config>
</attribute>
  <depends>jboss.messaging:service=ServerSocketFactory,type=SSL</depends>
</mbean>
<!-- This section is for custom (SSL) server socket factory  -->
<!--
    The server socket factory mbean to be used as attribute to socket invoker
      (see serverSocketFactory attribute above for where it is used). This
      service provides the exact same API as the ServerSocketFactory, so can be
      set as an attribute of that type on any
    MBean requiring an ServerSocketFactory.
   -->
   <mbean code="org.jboss.remoting.security.SSLServerSocketFactoryService"
     name="jboss.messaging:service=ServerSocketFactory,type=SSL" display-name="SSL
     Server Socket Factory">
     <depends optional-attribute-name="SSLSocketBuilder" proxy-
       type="attribute">jboss.messaging:service=SocketBuilder,type=SSL</depends>
   </mbean>
   <!--
     This service is used to build the SSL Server socket factory. This will be
       where all the store/trust information will be set. If do not need to make
       any custom configurations, no extra attributes need to be set for the
       SSLSocketBuilder and just need to set the javax.net.ssl.keyStore and
       javax.net.ssl.keyStorePassword system properties. This can be done by just
       adding something like the following to the run script for JBoss (this one is
       for run.bat):
     set JAVA_OPTS=-Djavax.net.ssl.keyStore=.keystore -
       Djavax.net.ssl.keyStorePassword=opensource %JAVA_OPTS%
     Otherwise, if want to customize the attributes for SSLSocketBuilder, will need
       to uncomment them below.
   -->
   <mbean code="org.jboss.remoting.security.SSLSocketBuilder"
     name="jboss.messaging:service=SocketBuilder,type=SSL" display-name="SSL Server
     Socket Factory Builder">
   <!-- IMPORTANT - If making ANY customizations, this MUST be set to false.
     Otherwise, will used default settings and the following attributes will be
     ignored. -->
   <attribute name="UseSSLServerSocketFactory">false</attribute>
   <!-- This is the url string to the key store to use -->
   <attribute name="KeyStoreURL">messaging.keystore</attribute>
   <!-- The password for the key store -->
   <attribute name="KeyStorePassword">secureexample</attribute>
   <!-- The password for the keys (will use KeystorePassword if this is not set
     explicitly. -->
   <attribute name="KeyPassword">secureexample</attribute>
   <!-- The protocol for the SSLContext. Default is TLS. -->
   <attribute name="SecureSocketProtocol">TLS</attribute>
      <!-- The algorithm for the key manager factory.  Default is SunX509. -->
      <attribute name="KeyStoreAlgorithm">SunX509</attribute>
      <!-- The type to be used for the key store.
        Defaults to JKS. Some acceptable values are JKS (Sun's keystore format),
          JCEKS (Java Cryptography Extension keystore - More secure JKS), and PKCS12
          (Public-Key Cryptography Standards #12 keystore - Keystore - version of
          RSA's Personal Information Exchange Syntax Standard). These are not case sensitive.
      -->
      <attribute name="KeyStoreType">JKS</attribute>
   </mbean>
</server>
```

Once you have completed these steps, you can restart your server. Thereafter, you can use the connection factory SecureConnectionFactory. It will require you to set up your client to use SSL to connect to the JMS queues.

Message Queue Security

Securing TIBCO Messaging: The sample configuration provided below, using TIBCO Enterprise Messaging Server version 5.0, is an example for illustration purposes only. The sample is by no means exhaustive or optimal. There are other approaches to securing message queue, each with its own strengths and weaknesses. Since message queues are provided by third party vendors, we strongly encourage you to discuss your approach to hardening or securing message queues with TIBCO messaging specialists. Products can evolve; hence, processes for hardening them might change over time. If you have any problems with the processes discussed here, we ask you work with TIBCO directly. You should use the following recommendations as a starting point and customize the configuration for your specific operating environment.

Configure JMS Objects

Use the TIBCO command line utility, tibemsadmin, to create your JMS objects and to configure user and group permissions. This utility can be found in the directory `<TIBCO>/ems/5.0/bin`.

1. Start the TIBCO server, enter the tibmsd command.
2.

   To start the TIBCO admin tool, enter the tibemsadmin command.
3. At the prompt, type: `connect`

   Note: If this is the first time you are starting the administration tool or if you have not set the password for the admin user, type connect and hit enter twice to authenticate as the admin user.
4. To create a Queue Connection Factory, enter:

   `create factory <Connection Factory Name> <Connection Factory Type>`

   Example: `create factory secureqcf queue`
5. To make the QCF accessible from other hosts, use the addprop command to set the URL to listen on an external address:

   `addprop factory <Connection Factory Name> url=<url-string>`

   Example: `addprop factory secureqcf url=tcp://devhost:7222`
6. Create a queue:

   `create queue <queue-name>`

   Example: `create queue securequeue`

Configure Users, Groups, and Permissions

After creating the queue and the QCF, create users and groups, and then grant queue permissions to the groups.

To do this task:	Enter this command:	Example
Create a group	`create group <groupname>`	`create group securegroup`
Create a user	`create user <username>`	`create user secureuser`
Change the password for a user	`set password <username> <new password>`	`set password secureuser securepassword`
Add a user to a group	`add member <group name> <user to be added>`	`add member securegroup secureuser`

To do this task:	Enter this command:	Example

Grant permissions on a queue to a given group	`grant queue <queuename> group=<groupname> <permission>`	`grant queue securequeue group=securegroup send`
		
		`grant queue securequeue group=securegroup receive`
		
		`grant queue securequeue group=securegroup browse`

Enable authorization to a queue	`addprop queue <queuename> secure`	`addprop queue securequeue secure`

Next, configure the client side.

Configuring the JMS Client to Work with TIBCO EMS

You will need the following three items to connect to an unsecured queue on TIBCO:

- URL — `tcp://<hostname>:<port>`

  Example: `tcp://tibserver:7222`
- ICF — com.tibco.tibjms.naming.TibjmsInitialContextFactory
- QCF — Name configured in the connection factory setup steps

To connect to the secured queues you created above, you will need to pass a username and password for both JNDI and queue-based security.

1. In the SDF, configure these four properties:
   - sci.queuebasedsecurity.userid
   - sci.queuebasedsecurity.password
   - java.naming.security.principal
   - java.naming.security.credentials
2. Add the following JARs to the client CLASSPATH:
   - `<TIBCO>/ems/5.0/lib/jms.jar`
   - `<TIBCO>/ems/5.0/lib/tibjms.jar`

SSL and TIBCO

TIBCO supports registering with multiple different JNDIs, such as JBoss and IBM® WebSphere®. It also ships with a built-in JNDI.

Note: This document assumes you are using the built-in JNDI. Other configurations are beyond the scope of this document.

TIBCO ships with a default SSL configuration file. To start the server up with the default SSL configuration, perform the following steps:

1. To start the tibco daemon, enter:
   ```
   ./tibemsd -config ../samples/config/tibemsdssl.conf -ssl_trace
   -ssl_debug_trace
   ```

   Note: This pathname applies to Version 5.1.2 V5. The pathname for other versions may vary.

2. To start the tibemsadmin program and connect to the SSL JNDI, enter these commands:

   ```
   $ ./tibemsadmin
   > connect ssl://www.acme.com:7243
   Login name (admin):
   Password:
   ```

   The system displays this response:

   ```
   Connected to: ssl:// www.acme.com:7243
   ssl:// www.acme.com:7243>
   ```
3. To create a Queue Connection Factory (QCF) on the SSL JNDI, enter this command:

   ```
   ssl://www.acme.com:7243> create factory SSL_QCF queue url=ssl://7243
   ```

   The system displays this response:

   ```
   QueueConnectionFactory 'SSL_QCF' has been created
   ```
4. To achieve an SSL connection in a test program, add the following parameters before creating the initial context:

   ```
   com.tibco.tibjms.naming.security_protocol=ssl
   java.naming.factory.url.pkgs=com.tibco.tibjms.naming
   com.tibco.tibjms.naming.ssl_trusted_certs=/u01/home/<dir>/apps/
     tibco/ems/5.0/bin/certs/server_root.cert.pem
   com.tibco.tibjms.naming.ssl_enable_verify_hostname=false
   ```

   Note: The value for the com.tibco.tibjms.naming.ssl_trusted_certs context parameter depends upon the setup given in your tibems configuration file. Out of the box with Version 5.1.2. V5, for example, use `<TIBCO>/samples/certs/server_root.cert.pem`.

   Note: The com.tibco.tibjms.naming.ssl_enable_verify_hostname context parameter must be set to false if you are using the out-of-the-box default configurations and the host name referenced in the certificate will not match the host name where your server is running.
5. Add the following JARs in order to get your TIBCO JMS client to connect with SSL:
   - `<TIBCO>/5.0/lib/tibjms.jar`
   - `<TIBCO>/5.0/lib/jms.jar`
   - `<TIBCO>/5.0/lib/tibcrypt.jar`
   - `<TIBCO>/5.0/lib/slf4j-api-1.4.2.jar`
   - `<TIBCO>/5.0/lib/slf4j-simple-1.4.2.jar`

Securing IBM® WebSphere® MQ Messaging: The sample configuration provided below, using IBM® WebSphere® MQ version 7.0.1, is an example for illustration purposes only. The sample is by no means exhaustive or optimal. There are other approaches to securing message queue, each with its own strengths and weaknesses. Since message queuing software is provided by third party vendors, we strongly encourage you to discuss your approach to hardening or securing message queues with IBM WebSphere MQ specialists. Products can evolve; hence, processes for hardening them might change over time. If you have any problems with the processes discussed here, we ask you work with IBM® directly. You should use the following recommendations as a starting point and customize the configuration for your specific operating environment.

Prerequisites

Install the correct version of WebSphere MQ and all patches up to the latest fix pack. WebSphere MQ is not a JMS, but it can be accessed via JMS implementation classes. To do so, you may need to install additional WebSphere MQ packages for Java.

The Java packages for WebSphere MQ provide a tool known as JMSAdmin that can be used to bind JNDI entries for interfacing to WebSphere MQ as a JMS.

Configure WebSphere MQ Messaging for Security

Normally, WebSphere MQ is installed as user mqm and the installation directory is `/opt/mqm`, referred to here as WMQ_HOME. All commands should be run as user mqm unless otherwise noted.

1. Create a queue manager.

   A queue manager is the process with which message clients interact to create and retrieve messages, and where administrators manage queues.

   The crtmqm command line utility is used to create a queue manager for your WebSphere MQ installation. It takes one argument: the queue manager name. You will reference this name when starting or stopping the queue manager and when binding JNDI entries. In this example, we name the queue manager sciqm:

   ```
   $ cd <WMQ_HOME>/bin
   bin $ ./crtmqm sciqm
   ```

2. To start the queue manager, enter this command:

   ```
   bin $ ./strmqm sciqm
   ```

3. Verify that the queue manager is running with this command:

   ```
   bin $ ./dspmq
   QMNAME (sciqm)          STATUS (Running)
   ```

Configure Queues

Next, configure two agent and two integration queues.

The runmqsc command is a command line utility used to create objects in WebSphere MQ. The tool is found in the directory <*WMQ_HOME*>/bin. You need to pass the queue manager name you used when you issued the crtmqm command to identify the queue manager with which you want to interact.

```
bin $ ./runmqsc sciqm
define qlocal(SCHEDULE)
define qlocal (RELEASE)
define qlocal(CREATEORDER)
define qlocal(SHIPNOTICES)
```

It is beyond the scope of this document to explain all of the commands and options available through the runmqsc command line utility. For example, define commands are used to define objects for the queue manager, including defining a queue and creating channels, processes, remote queues and other objects.

It is possible to override the attributes of the queue at the time that you create it. For instance, you can override the maximum queue depth at the time that you create the queue.

For details on the define and other commands, refer to the WebSphere MQ documentation.

Configure Authentication to the Queue Manager and to Queues

Next, configure authentication to the queue manager and to queues to allow a user to connect to the queue manager and put messages into a queue. To do this, start the WebSphere MQ Listener.

1. Create the user on the system where the queue manager is running.
2. Use the setmquat command to allow the user to connect to connect to the queue manager.

   ```
   bin $ ./setmqaut -m sciqm -t qmgr -p quser +all
   ```

   The following commands authorize user "quser" to perform any action on the four queues:

   ```
   bin $ ./setmqaut -m sciqm -n SCHEDULE -t q -p quser +all
   bin $ ./setmqaut -m sciqm -n RELEASE -t q -p quser +all
   bin $ ./setmqaut -m sciqm -n CREATEORDER -t q -p quser +all
   bin $ ./setmqaut -m sciqm -n SHIPNOTICES -t q -p quser +all
   ```
3. Issue the following command to start the MQ and establish a listener:

   ```
   bin $ ./runmqlsr -m sciqm -t TCP
   ```

   This starts the listener on the default port of 1414 for the given queue manager. If an alternate port is required, use the -p option. For example:

   ```
   bin $ ./runmqlsr -m sciqm -t TCP -p 1415
   ```

   Note: runmqlsr ships with WebSphere MQ. You can also use inetd or xinetd as an alternate method of establishing a listener.

Binding the MQ Queues to JNDI

After you create the queue manager and queues and have started the MQ listener, you need to bind the queues to the JNDI so that they can be located by your client applications.

The Java packages for WebSphere MQ provide a utility known as JMSAdmin that can be used to bind JNDI entries for WebSphere MQ. The tool can be used to bind queues into virtually any JNDI you wish to use. This document will focus on file bindings.

In a normal install, if you have installed the Java packages for WebSphere MQ, the JMSAdmin tool and related files will be in <*WMQ_HOME*>/java/bin. There are a few key files that you must back up and configure before you start.

Configuring JMSAdmin.config

The JMSAdmin.config file defines the ICF and URL the tool will connect to. To use file bindings, configure the file with values similar to what is shown here:

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
PROVIDER_URL=file:/opt/mqm/filebindings
```

The INITIAL_CONTEXT_FACTORY tells JMSAdmin that you want to use file bindings. File bindings uses a regular file on the operating system to perform lookups to objects.

The PROVIDER_URL tells JMSAdmin where it should create the file bindings. In the example above, JMSAdmin will create /opt/mqm/filebindings/.bindings. This is a regular file that you can open with a text editor. Users or programs that need to read the binding file will need read access to the file.

JMSAdmin Shell Script

The file JMSAdmin is a sample shell script used to invoke the underlying java program JMSAdmin. To use the script, define the following environment variables:

- Define JAVA_HOME and add the $JAVA_HOME/bin to the PATH in the shell script.
- Add the following files to the CLASSPATH:
  - ${WMQ_HOME}/java/lib/fscontext.jar
  - ${WMQ_HOME}/java/lib/com.ibm.mq.jar
  - ${WMQ_HOME}/java/lib/com.ibm.mqjms.jar
  - ${WMQ_HOME}/java/lib/jms.jar

Note: The above CLASSPATH allows JMSAdmin to bind message queue objects into the file bindings (fscontext) when JMSAdmin is running with the Java that is shipped with the WebSphere MQ Java client. Java clients that do not use the MQ Java JDK may require the following additional JAR files:

- ${WMQ_HOME}/java/lib/providerutil.jar
- ${WMQ_HOME}/java/lib/dhbcore.jar
- ${WMQ_HOME}/java/lib/com.ibm.mq.jmqi.jar

For instance, those JAR files are needed if a WebLogic server running on a Sun JDK is used to invoke synchronous services that put messages in the MQ queues.

You can now run the JMS Admin tool, JMSAdmin. Start it in interactive mode by issuing a JMSAdmin command:

```
bin $ ./JMSAdmin
```

If successful, a prompt displays:

```
5724-H72, 5655-L82, 5724-L26 (c) Copyright IBM Corp. 2002,2005. All Rights
   Reserved.
Starting Websphere MQ classes for Java(tm) Message Service Administration
InitCtx>
```

It is a common practice to put all the JMSAdmin commands into a script file with an scp file extension to define the objects in the JNDI. Instead, the document intends to explain how to create a QCF and JNDI entries for queues. The entries in a typical scp file are shown here:

```
InitCtx> def qcf(AGENT_QCF) qmgr(sciqm) tran(client) chan(SYSTEM.DEF.SVRCONN)
  host(localhost) port(1414)
InitCtx> def q(SCHEDULE) qu(SCHEDULE) qmgr(sciqm)
InitCtx> def q(RELEASE) qu(RELEASE) qmgr(sciqm)
InitCtx> def q(CREATEORDER) qu(CREATEORDER) qmgr(sciqm)
InitCtx> def q(SHIPNOTICES) qu(SHIPNOTICES) qmgr(sciqm)
```

The first line in this example shows how to define a QCF called AGENT_QCF that will service queues on the queue manager via the channel SYSTEM.DEF.SVRCONN via port 1414. It is important to understand that the port 1414 must have a listener running on it for this to achieve the desired goal. See the instructions to create a listener through runmqlsr under *Configure Authentication to the Queue Manager and to Queues*.

The remaining four lines show how to create four JNDI entries for the four queues. Notice that the qmgr is called out on the queue JNDI definition.

Setting Up Security in IBM® WebSphere® MQ:
JNDI Security

Now that you have configured and bound the queues to the JNDI, you can now start to secure the JMS by configuring SSL on the JMS channels and enforcing the clients to authenticate to the message queue.

Important: This approach uses file binding. To secure access to the file binding, you need to ensure the bindings file is kept on a file system that only your Java clients can read from.

Queue Authentication

The queue users and their privileges are defined using the setmqaut command. The IBM agents need to pass in the sci.queuebasedsecurity.userid and sci.queuebasedsecurity.password parameters in the agent criteria in order to authenticate.

SSL with WebSphere MQ

IBM WebSphere® MQ supports one-way and two-way SSL. This procedure presents configuration of one-way SSL where the client verifies that the identity of the Queue Manager it is connecting to can be verified through a certificate. This concept can be extended to enable two-way SSL.

1. Use the runmqsc command to verify the location of the keystore and to create a channel that will be secured with SSL:
   ```
   mqm@devdell04:/opt/mqm/bin> ./runmqsc sciqm
   dis qmgr sciqm
   AMQ8404: Display Queue Manager details.
      QMNAME (sciqm)
      SSLKEYR (/var/mqm/qmgrs/sciqm/ssl/key)
   ```
2. Create a channel to be secured via SSL:
   ```
   define channel(SSL_SVRCONN) CHLTYPE(SVRCONN) SSLCIPH(RC4_MD5_US)
      SSLCAUTH(OPTIONAL) TRPTYPE(TCP)
           4 : define channel(SSL_SVRCONN) CHLTYPE(SVRCONN) SSLCIPH(RC4_MD5_US)
      SSLCAUTH(OPTIONAL) TRPTYPE(TCP)
   AMQ8014: WebSphere MQ channel created.
   ```
3. Set up the environment and start the gsk7ikm command.

   The gsk7ikm tool is used to create the required keystores and certificates.

   Note: These steps assume that you have xwindowing support. If you are using PuTTY, Windows, or the WebSphere MQ GUI-based configuration utilities, the steps may differ.
   ```
   export JAVA_HOME=/opt/mqm/ssl/jre
   export DISPLAY=10.10.20.81:8
   gsk7ikm
   ```
4. In the location discovered by the dis qmgr SSLKEYR command, create the server keystore file:
   - From IBM Key Management menu bar, select **Key Database File > New** to display the **New** dialog box.
   - From the **Key database type** drop down, select CMS.
   - Enter the key file name in the **File Name:** text box. Example: key.kdb

- In the **Location:** text box, enter the path returned by dis qmgr SSLKEYR and press `OK`.

  The **Password Prompt** dialog box displays.

5. Set the server keystore password:
   - In the **Password Prompt** dialog box, type a password in the **Password:** text box.
   - Retype the password in the **Confirm Password:** text box.
   - For **Set expiration time?**, accept the default.
   - Select **Stash the password to a file?** and press **OK**.

     A dialog displays, confirming that the password has been encrypted and saved.
6. Create a self-signed certificate in the server keystore:
   - From the **IBM Key Management** menu bar, select **Create > New Self-Signed Certificate...**

     The **Create New Self-Signed Certificate** dialog box displays.
   - In the **Key Label** text box, type in lower case: `ibmwebspheremq<`*`queue-manager-name`*`>`.

     Example: If your queue manager name is SSL_QMGR, set the **Key Label** to `ibmwebspheremqssl_qmgr`.

     Important: Set the Key Label correctly on the self-signed certificate. Not doing so will cause the SSL connections to fail.
   - From the **Version** drop down box, select the certificate version. Example: X509 V3
   - From the **Key Size** drop down, select the key size for the key pair to be generated. Example: 1024
   - In the **Common Name** text box, type the fully qualified name of the host machine and press **OK**. Example: SSL_QMGR

     The certificate is created.
7. Extract the certificate to an .arm file:
   - Select the newly created certificate and press **Extract Certificate...**.

     Note: The **Extract Certificate...** button is located in the bottom-right corner of the Key database content section.

     The **Extract Certificate to a File** dialog box displays.
   - From the **Data Type** drop down box, select the data type for the target file. Example: Base64-encoded ASCII data
   - In the **Certificate File** name: text box, type a file name for the data to be created or obtained. Example: cert.arm
   - In the **Location:** text box, type the location in which the specified file is located and press **OK**. Example: `<`*`dir`*`>/resources/ssl/mq/ssl_qmgr`

     The certificate is extracted to the .arm file.
8. Create the new client keystore file:
   - From the IBM Key Management menu bar, select **Key Database File > New**.
   - From the **Key database type** drop down, select **JKS**.
   - Type the key file name in the **File Name:** text box. Example: key.jks
   - In the **Location:** text box, type the path where the file will be saved. Example: `<`*`dir`*`>/resources/ssl/mq/ssl_qmgr`

- Press **OK** to bring up the `Password Prompt` dialog box.
- Type a password and press **OK**.

  The client keystore is created.

9. Add the server's signer certificate to the client keystore:
   - From the **Key database content** section, press the **Add...** button.

     The **Add CA's Certificate from a File** dialog box displays.
   - From the **Data Type** drop down, select the data type for the target file. Example: Base64-encoded ASCII data
   - In the **Certificate file name:** text box, type a file name for the data to be created or obtained. Example: cert.arm
   - In the **Location:** text box, type the file location and press **OK**.

     The **Enter a Label** dialog box displays.
10. In the **Enter a label for the certificate:** text box, type (e.g. ibmwebspheremqssl_qmgr) and press **OK**.
11. Type a label name for the certificate, using the name you created for your server keystore. Example: `ibmwebspheremqssl_qmgr`

    Important: Take care to provide the correct label. Not doing so will cause the SSL connections to fail.

Implementation Notes®

When getting a connection the queue manager via fscontext, you may see a MQJMS2013 error. Often, it can be solved by passing blank (single space) for username and password. These correlate to sci.queuebasedsecurity.userid and sci.queuebasedsecurity.password in agents and services.

It is possible to use nearly any JNDI repository to store the queue connection factory and queue lookups. IBM provides the following page for troubleshooting issues with WebSphere MQ when registered on WebSphere Application Server's JNDI:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/rmj_prob0.html

Specifically, this page explains issues found with BINDINGS transport and security.

Network Security

It is unfortunate, but computers exposed to untrusted or unrestricted networks such as the Internet are vulnerable to probes and attacks. The goal of network hardening is to reduce the risk of attacks by placing servers into network security zones that allow authorized network communication to support your business needs, and to restrict all other unauthorized network access.

Network hardening is a complex subject that is beyond the scope of this document. We recommend you work with your corporate network and security teams.

If you follow the planning steps outlined in "Security Engineering Steps" on page 3, you will have a very good understanding of the following:

- System architecture, including the list of software components

- Use case scenarios, including who your users are, what they will likely do on the system, non-functional requirements such as availability targets, and transactional volumes
- Integration architecture, including all data flows between components
- Your security threat model and security requirements

There are many approaches to deploying theSterling Selling and Fulfillment Suite applications. We present some of the more common approaches in the *Deployment Scenarios* section of this document.

Sterling Sensitive Data Capture Server

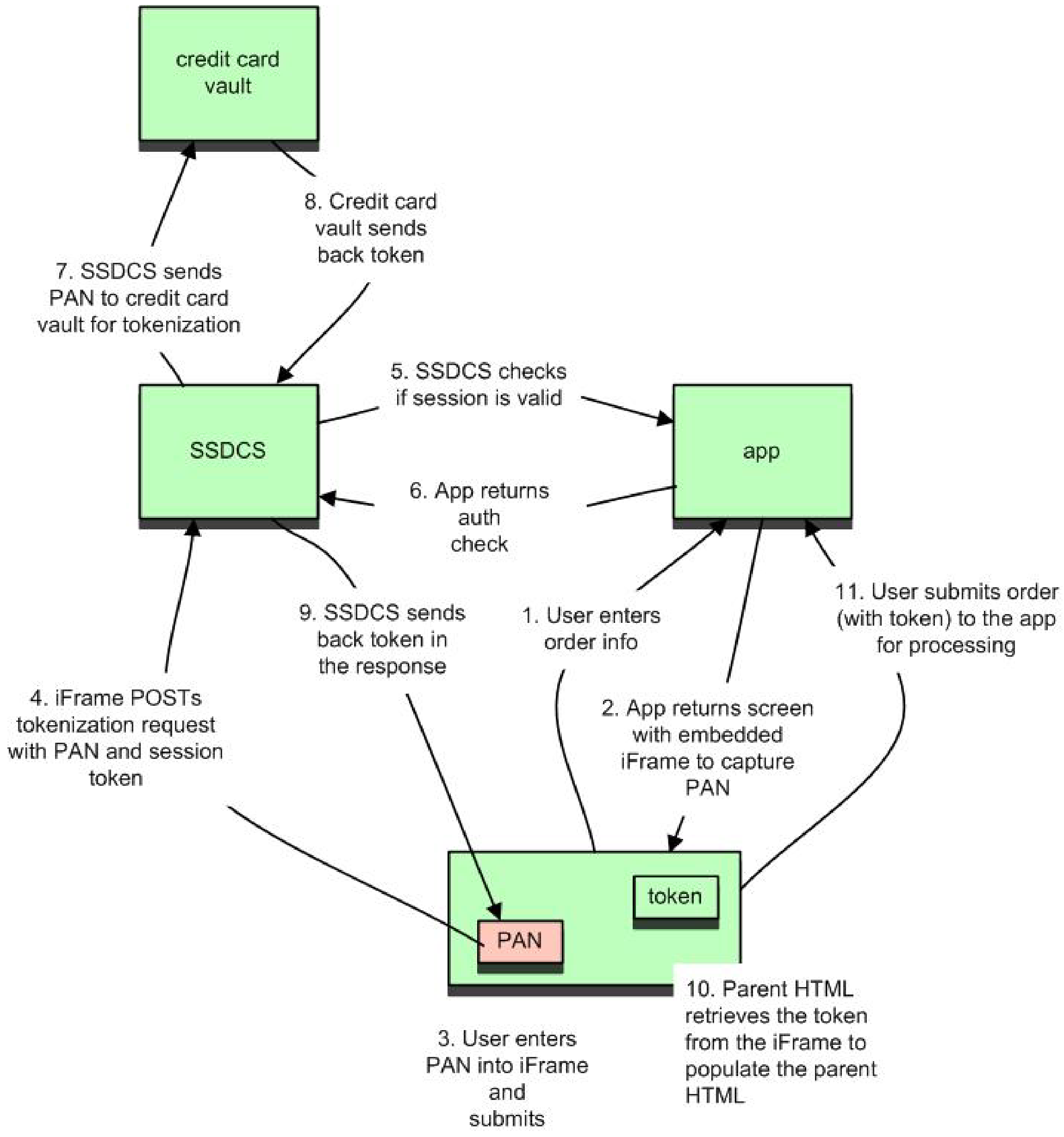
This information applies only if you want to configure the Sterling Sensitive Data Capture Server to capture credit card numbers on behalf of theSterling Selling and Fulfillment Suite applications.

The Sterling Sensitive Data Capture Server serves a simple but very critical function, which is central to our PCI PA-DSS strategy (see "PCI PA-DSS Strategy Introduction" on page 7). Specifically, the Sterling Sensitive Data Capture Server is designed to capture and convert sensitive PAN information to tokens so that only the tokens, never the sensitive PAN data, are sent to Sterling Selling and Fulfillment Suite applications for processing and storage. With this approach, customers should be able to keep the focus of the PCI DSS auditing effort on the Sterling Sensitive Data Capture Server and, more importantly, away from the rest of the Sterling Selling and Fulfillment Suite applications.

This section discusses:
- How the Sterling Selling and Fulfillment Suite applications are integrated to the Sterling Sensitive Data Capture Server
- How to configure load balancers to automatically route transactions between the Sterling Sensitive Data Capture Server and the Sterling Selling and Fulfillment Suite applications

Sterling Sensitive Data Capture Server Integration

The integration between theSterling Selling and Fulfillment Suite applications and Sterling Sensitive Data Capture Server is seamlessly transparent to your end users. The diagram shows the integration for browser-based applications.


credit card vault
8. Credit card vault sends back token
7. SSDCS sends PAN to credit card vault for tokenization
5. SSDCS checks if session is valid
SSDCS
app
6. App returns auth check
11. User submits order (with token) to the app for processing
9. SSDCS sends back token in the response
1. User enters order info
4. iFrame POSTs tokenization request with PAN and session token
2. App returns screen with embedded iFrame to capture PAN
token
PAN
10. Parent HTML retrieves the token from the iFrame to populate the parent HTML
3. User enters PAN into iFrame and submits

In the diagram, the following sequence of events occurs:

1. As a first step, the user at a browser interacts with one of the Sterling Selling and Fulfillment Suite applications.
2. At some point in the interaction, the application sends down a screen to solicit a Primary Account Number (PAN) from the user for payment. In that screen or parent HTML screen is a text field that is implemented as an embedded iFrame or inline frame (see shaded box).

   The iFrame will also contain a security token, which as you will see later, is used to authenticate tokenization requests.

3. The iFrame displays a text box for the user to enter the PAN. From the end user's perspective, that text field will look identical to the other text fields in the main HTML screen. However, it is a different window or conduit to the Sterling Sensitive Data Capture Server.
4. When the end user enters a PAN and presses the Save button, the iFrame sends the PAN and security token in a tokenization request to the Sterling Sensitive Data Capture Server.
5. On receiving a tokenization request, the Sterling Sensitive Data Capture Server will send the security token to a Sterling Selling and Fulfillment Suite application to ensure the tokenization request came from a valid session.
6. The application returns either a valid session confirmation or a rejection.
7. Upon receiving a successful confirmation, the Sterling Sensitive Data Capture Server then sends the PAN to your corporate credit card vault.
8. Upon tokenization (and storage of the PAN), the credit card vault service returns a token.
9. The returns the token to the calling iFrame. The iFrame moves the token into the payment method text field. The iFrame then submits the order to the Sterling Selling and Fulfillment Suite application.

To the user, the integration is seamless. All the user has to do is enter the PAN into a text box in the embedded iFrame inside the order entry screen (the parent screen). The iFrame sends the PAN to theSterling Sensitive Data Capture Server for tokenization, moves the returning token into the appropriate payment field and then sends the order capture transaction to the Sterling Selling and Fulfillment Suite.

How Sterling Sensitive Data Capture Server Integration is Achieved

At a high level, the integration is achieved through iFrame scripting and communication. The following three technical components are needed to achieve seamless integration:

- First, the Sterling Sensitive Data Capture Server and the rest of the Sterling Selling and Fulfillment Suite applications have to be deployed with the same domain.
- Second, content based routing or Layer 7 routing capable load balancer is needed to route transactions appropriately to the Sterling Sensitive Data Capture Server and the Sterling Selling and Fulfillment Suite applications.
- Third, a restrictive cookie path has to be set to ensure the session cookies are sent to their application.

Same Domain Requirement

For browser-based applications, the Sterling Selling and Fulfillment Suite applications rely on iFrame scripting to move information. iFrame scripting, however, is feasible only if the Sterling Sensitive Data Capture Server is in the same URL domain as the rest of the Sterling Selling and Fulfillment Suite applications in order for the iFrame to be allowed to move the token to the appropriate text fields in its parent HTML. This is a characteristic of browser security—specifically the browser's adherence to the same origin policy.

Same Origin Security Policy

The "same origin" policy is one of the most fundamental browser security mechanisms. It prevents HTML documents or scripts from a different domain from changing a document loaded from the current domain. This prevents malicious content from one domain or site from loading into the current HTML in order to steal data such as cookies or to issue unintended transactions.

What Makes a Domain the Same?

Two page requests are considered to have the same origin if the protocol, hostname, and port are the same. The following table from the Mozilla site (https://developer.mozilla.org/En/Same_origin_policy_for_JavaScript) provides examples of origin comparison when the target URL is validated against the URL http://store.company.com/dir/page.html.

Target URL	Outcome	Reason
http://store.company.com/dir2/other.html	Success	
http://store.company.com/dir/inner/another.html	Success	
https://store.company.com/secure.html	Failure	Different protocol
http://store.company.com:81/dir/etc.html	Failure	Different port
http://news.company.com/dir/other.html	Failure	Different host

In the first two target URLs, the same origin check is successful because the protocol (http), hostname (store.company.com) and port (default 80) are the same as the comparison URL. The third, fourth, and fifth target URLs fail because the protocol (https), port, and hostname, respectively, are different from the comparison URL.

The "How to Deploy with the Same Domain" section present in the 9.0 was revised for 9.1 and moved to a new topic by that name.

How to Deploy with the Same Domain

There are three different approaches to deploying theSterling Sensitive Data Capture Server so that same domain/same origin constraints are satisfied. These options are:

- Demo or Development mode
- Layer 7 Routing
- Document.Domain

Demo or Development Mode

The easiest way to meet the "same domain" requirement is to deploy theSterling Sensitive Data Capture Server into each application server instance that is running the Sterling Selling and Fulfillment Suite applications. In this simple deployment, eachSterling Selling and Fulfillment Suite application including the Sterling Sensitive Data Capture Server could be deployed to (for example) https://sw.acme.com/sw and https://sw.acme.com/ssdcs or as https://localhost:8000/sw and https://localhost:8000 /ssdcs respectively. Since the domain (protocol, host, and port) are the same, the iFrame from Sterling Sensitive

Data Capture Server will be able to interact with its parent HTML from theSterling Distributed Order Management application.

This simple deployment is suitable for demonstration or development purposes only. For production, you should segregate the Sterling Sensitive Data Capture Server in its own application server and its own network segment away from the Sterling Selling and Fulfillment Suite applications to limit PCI DSS auditing effort.

Restrictive Cookie Path

The last step in the Sterling Sensitive Data Capture Server integration strategy is to set a cookie path so that the browser correctly sends cookies back to their applications.

Oracle WebLogic

To set the cookie path in Oracle WebLogic, complete the following:

1. Copy the weblogic.xml from `<install_dir>/repository/eardata/smcfs/descriptors/weblogic /WAR/WEB-INF/weblogic.xml` to the *<install_dir>*/ extensions directory.
2. Add the following:
   ```
   <session-descriptor>
      <session-param>
         <param-name>CookiePath</param-name>
         <param-value>/<context-path></param-value>
      </session-param>
   </session-descriptor>
   ```
   where *<context-path>* is the context path for the deployed application. For example, this would typically be /ssdcs, /smcfs, /swc, /sfs for the Sterling Sensitive Data Capture Server Sterling Distributed Order Management Sterling Web , and Sterling Field Salesapplications, respectively.
3. Rebuild the EAR file.

IBM WebSphere

In the WebSphere Administration Console, navigate to the **Session Manager > Cookie** tab. In the **Cookie** tab, set **Cookie Path** to the context path of your application. For example, this would typically be /ssdcs, /smcfs, /swc, /sfs for the Sterling Sensitive Data Capture Server,,Sterling Distributed Order Management Sterling Web , and Sterling Field Sales applications, respectively.

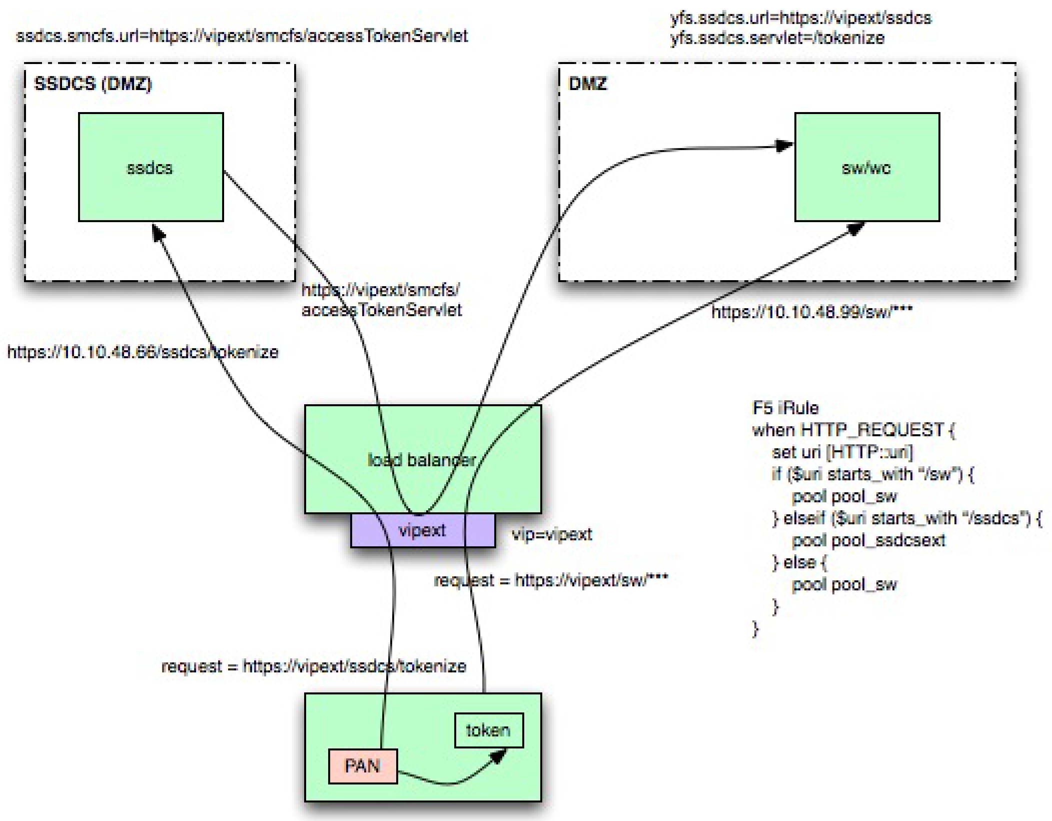
Deploying Application and Sterling Sensitive Data Capture Server on Different Nodes

When you deploy applications on different nodes, the host (and therefore, the domain) will be different. For example, if we were to deploy the Sterling Sensitive Data Capture Server and Sterling Webapplications to ssdcs.acme.com and smcfs.acme.com respectively, the hostname portion of the domain would be different. The following are two approaches to meeting the same domain requirement:

- Layer 7 Routing
- Document.Domain

Layer 7 Routing

In this approach, you can use a Layer 7 or Content Based Routing capable load balancer to meet the same domain requirement.


ssdcs.smcfs.url=https://vipext/smcfs/accessTokenServlet
yfs.ssdcs.url=https://vipext/ssdcs
yfs.ssdcs.servlet=/tokenize
SSDCS (DMZ)
ssdcs
DMZ
sw/wc
https://vipext/smcfs/accessTokenServlet
https://10.10.48.99/sw/***
https://10.10.48.66/ssdcs/tokenize
load balancer
vipext
vip=vipext
request = https://vipext/sw/***
request = https://vipext/ssdcs/tokenize
token
PAN

```
F5 iRule
when HTTP_REQUEST {
   set uri [HTTP::uri]
   if ($uri starts_with "/sw") {
      pool pool_sw
   } elseif ($uri starts_with "/ssdcs") {
      pool pool_ssdcsext
   } else {
      pool pool_sw
   }
}
```

Diagram Key: Abbreviation	Product or Application
ssdcs	Sterling Sensitive Data Capture Server
sw/wc	Sterling Web

In the diagram above, requests from the browser are sent to a load balancer to the virtual IP (VIP) https://vip.acme.com. The load balancer is configured to route the traffic to either the Sterling Sensitive Data Capture Server or Sterling Selling and Fulfillment Suiteapplications based on the URI shown to the right of the host and port. For example, if the load balancer is an F5 Big-IP, you could set up the following iRule to route requests to the appropriate load balancing pool (of servers) based on the URI.

```
when HTTP_REQUEST {
       set uri [HTTP::uri]
       if (uri starts_with "/ssdcs") {
             pool pool_ssdcs
       } elseif (uri starts_with "/smcfs") {
             pool pool_smcfs
       } else {
             pool pool_smcfs
       }
}
```

In this example, the Sterling Distributed Order Management and Sterling Sensitive Data Capture Server applications are deployed with the context root /smcfs and

/ssdcs, respectively. During the normal course of interaction, the F5 iRule will route all HTTP requests with the URL https://vip.acme.com/smcfs to the pool of Sterling Distributed Order Management application servers. When the iFrame at the browser sends a tokenization request to https://vip.acme.com/ssdcs, the Layer 7 routing load balancer will send the request to the pool of Sterling Sensitive Data Capture Server servers.

You must configure the following parameters in the Sterling Selling and Fulfillment Suite applications and Sterling Sensitive Data Capture Server:

- yfs.ssdcs.url — This parameter provides the Sterling Selling and Fulfillment Suite applications with the location of the Sterling Sensitive Data Capture Server.
- ssdcs.smcfs.url — The corresponding parameter tells the Sterling Sensitive Data Capture Server where to find a Sterling Selling and Fulfillment Suite application to verify that the tokenization request came from a valid user.

The*Sterling Selling and Fulfillment Foundation: Sterling Sensitive Data Capture Server, Release 1.1: Configuration Guide* provides detailed instructions on how to set these parameters.

Document Domain

The third option for deployment in the same domain is to set the Web pages from the Sterling Sensitive Data Capture Server and Sterling Selling and Fulfillment Suite to a common domain. In the example below, the requests are sent to the respective host or cluster—that is, requests to Sterling Weband the Sterling Sensitive Data Capture Server will be sent to sw.acme.com and ssdcs.acme.com, respectively. The response sets the document's domain to a commmon acme.com. By setting the domain to the suffix "acme.com," the browser treats the pages as belonging to the same domain.

This approach works only if the Sterling Sensitive Data Capture Server and main applications are configured with the same domain suffix. In the example above, the two applications have the same domain suffix ("acme.com"). This technique does not work if the Sterling Sensitive Data Capture Server is, for example, deployed to a different domain suffix such as ssdcs.acmeinc.com.

To set the document.domain, set the ssdcs.document.domain in the Sterling Sensitive Data Capture Server property file and the yfs.document.domain in the Sterling Selling and Fulfillment Foundation property file.

Sterling Web Only Deployment Scenario

The Sterling Web application is a Web shopping environment, with users coming from the Internet. Because it is an Internet-facing application, one typically hosts Sterling Web servers in the Internet-facing network segment, typically called the Demilitarized Zone (DMZ).

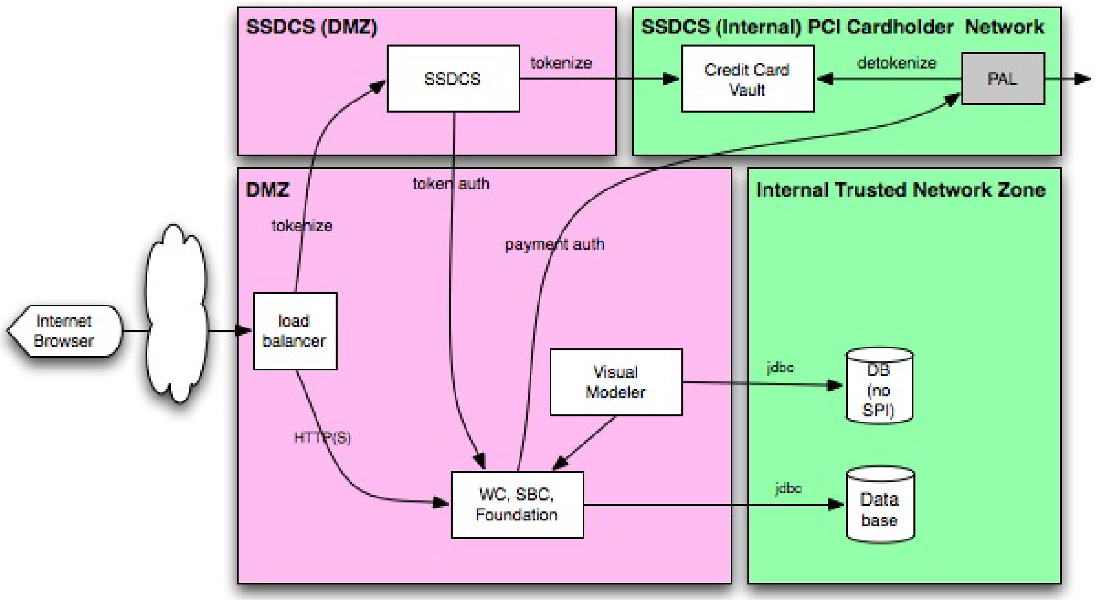


Diagram Key: Abbreviation	Product, Application or Type of Data
SSDCS	Sterling Sensitive Data Capture Server
PCI	Payment Card Industry
PAL	Payment Abstraction Layer
WC	Sterling Web
SBC	Sterling Business Center
SPI	Sensitive Personal Information

In the diagram above, Sterling Web users from the untrusted networks or from internal networks send requests to a reverse proxy or load balancer. The load balancer sends browsing or shopping cart transactions to the Sterling Web application. In this scenario, customers typically access the Sterling Web application over two network zones—the DMZ and the Internal Trusted Network Zone. Depending on your operational requirements or your corporate security standards, your deployment scenario could differ.

The Sterling Sensitive Data Capture Server , Payment Abstraction Layer (PAL), and credit card vault applications are required only if your Sterling Web application has to capture credit card number as payment information. As described in *IBM PCI PA-DSS Strategy* and Sterling Sensitive Data Capture Server guides, the primary purpose of the Sterling Sensitive Data Capture Server is to offload the capturing of Primary Account Number (PAN) information from the Sterling Web application. The Sterling Sensitive Data Capture Server sends the PAN to your corporate credit card vault for tokenization. Once tokenized, the Sterling Sensitive Data Capture Server transfers the token to the Sterling Web application so that Sterling Web can complete the order capture. This PAN tokenization strategy should allow you to removeSterling Web from the PCI DSS auditing scope.

If your solution requires the Sterling Sensitive Data Capture Server, you could deploy it into a separate network zone or in the DMZ with the Sterling Web application. Your credit card vault will likely be implemented in your corporate cardholder data network.

The Sterling Web application requires a Visual Modeler to define its item models. Since the Visual Modeler is typically not Internet-facing, you could implement it in the Trusted network. Item maps created or modified in the Visual Modeler can be transferred to the Sterling Web application through a variety of means, including SFTP or SCP. If the Visual Modeler is located in the Trusted Zone, you will have to open your inner firewall to allow outbound file copies or transfers. Alternatively, as depicted in the diagram above, you can place the Visual Modeler in the DMZ and its database in the trusted zone.

Item maps created or modified in the Visual Modeler can be transferred to the application through a variety of means, including SFTP or scp. If the Visual Modeler is located in the Trusted Zone, you will have to open your inner firewall to allow outbound file copies or transfers.

Sterling Distributed Order Management with Optional Applications Deployment Scenario

The Sterling Distributed Order Management application provides order fulfillment functionality for orders captured from your order capture channels. Application users, such as customer service representatives, typically come from the corporate internal network. As a result, the application is on the internal network, along with other applications that are typically not Internet facing, such as the Sterling Warehouse Management System, Sterling Business Intelligence, and Sterling Call Center and Sterling Store.

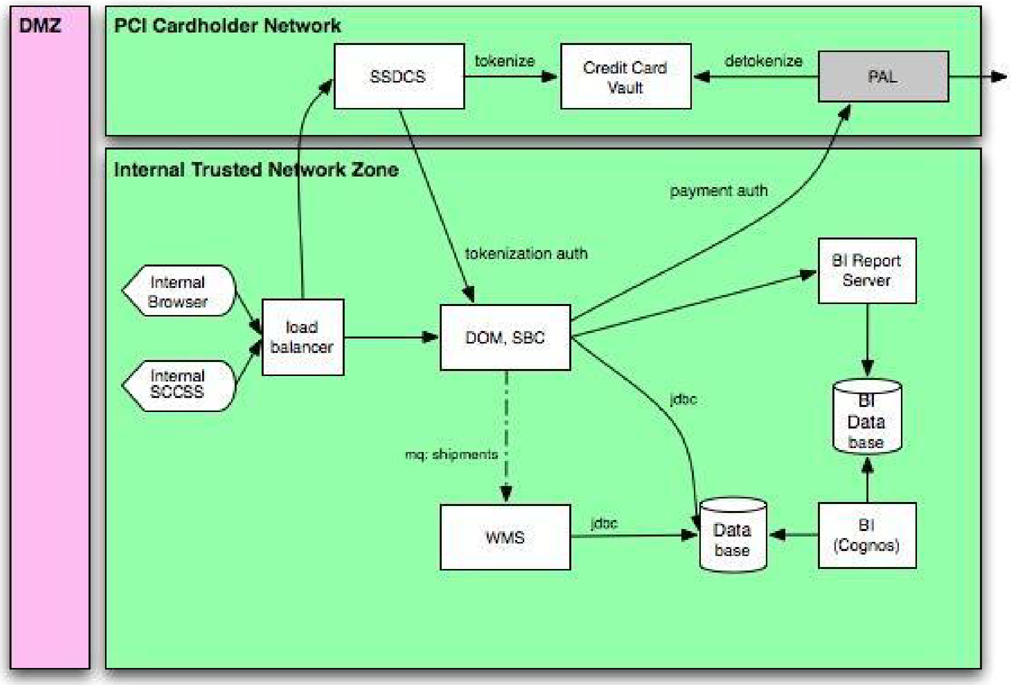


Diagram Key: Abbreviation	Product or Application
SSDCS	Sterling Sensitive Data Capture Server
PAL	Payment Abstraction Layer
SCCSS	Sterling Call Center and Sterling Store
DOM	Sterling Distributed Order Management

Diagram Key: Abbreviation	Product or Application
SBC	Sterling Business Center
WMS	Sterling Warehouse Management System
BI	Sterling Business Intelligence

The ,Sterling Sensitive Data Capture Server PAL, credit card vault, and the cardholder data network are needed if the Sterling Call Center and Sterling Store or Sterling Distributed Order Management applications capture PAN information. You may want to place them into your PCI cardholder network if they are needed.

The Sterling Warehouse Management System can be placed in the same network zone as theSterling Distributed Order Management application. Alternatively, they could be placed into their own remote network zones, especially if they are implemented in remote global locations.

Sterling Distributed Order Management and Sterling Web Deployment Scenario

In addition to the standalone deployment options for theSterling Distributed Order Management and Sterling Web applications, you can also deploy theSterling Selling and Fulfillment Suite applications together in an integrated approach where they all use the same database. This tightly coupled level of integration allows each application immediate access to information that has been changed or created in the other application. For example, orders captured in Sterling Web, Sterling Field Sales, Sterling Distributed Order Management, orSterling Call Center andSterling Store become immediately available in Sterling Distributed Order Management for fulfillment.

Consider the following when you run the Sterling Selling and Fulfillment Suite applications in the integrated mode:

- Sterling Selling and Fulfillment Suite applications are designed to cache database records for performance and scalability. When one of the applications modifies cached data, it will automatically notify the other application.

  You can find more data on the database caching feature in *Sterling Selling and Fulfillment Foundation: Performance Management Guide*.
- Application administrators can use the System Management Console to manage these applications, including managing thread levels, starting application traces, and requesting application shutdown.

When designing your deployment strategy and your network security zones, you must allow the flow of these critical intra-application communications. Two possible approaches are:

- Deploy the Sterling Selling and Fulfillment Suite applications in the same network security zone. Doing so means the applications are not separated by firewalls and, as a result, have unrestricted intra-application data flows.
- Deploy Internet-facing applications, such as Sterling Web and Sterling Field Sales , into the DMZ and deploy the non Internet-facing applications, such as Sterling Distributed Order Management and Sterling Call Center and Sterling Store applications, into the internal trusted network zone. If you choose this deployment option, you need to configure the applications to work with your firewalls.

Same Network Security Zone Deployment

In this deployment approach, theSterling Web, Sterling Distributed Order Management, and Sterling Field Salesapplications are deployed in the same network security zone, ensuring unrestricted intra-application communication flow.

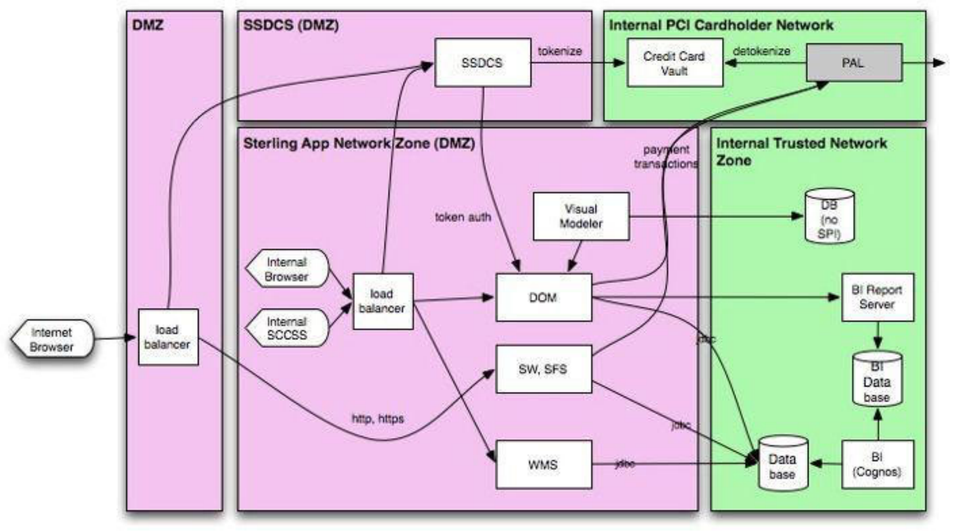


Diagram Key: Abbreviation	Product, Application, or Type of Data
SSDCS	Sterling Sensitive Data Capture Server
PCI	Payment Card Industry
PAL	Payment Abstraction Layer
SCCSS	Sterling Call Center and Sterling Store
DOM	Sterling Distributed Order Management
SW	Sterling Web
SFS	Sterling Field Sales
WMS	Sterling Warehouse Management System
SPI	Sensitive Personal Information
BI	Sterling Business Intelligence

The Sterling Distributed Order Management, Sterling Web, and Sterling Field Sales applications are placed into a security zone: the Sterling Application Network Zone (SANZ). Optionally, you can then add the Sterling Warehouse Management System, which has the same inter-application communication requirements as the other applications, into the SANZ.

The database and, optionally, the Sterling Business Intelligence application, are placed into the Trusted Zone.

The Sterling Sensitive Data Capture Server is placed into its own network zone in the DMZ. The PAL and credit card vault are placed into a PCI cardholder data network that is part of the internal network.

Different Network Security Zone Deployment

In this deployment strategy, we placed theSterling Call Center in the DMZ and Sterling Distributed Order Management in the trusted network zone. This is similar to the Sterling Call Center only and Sterling Distributed Order Management only deployment scenarios.

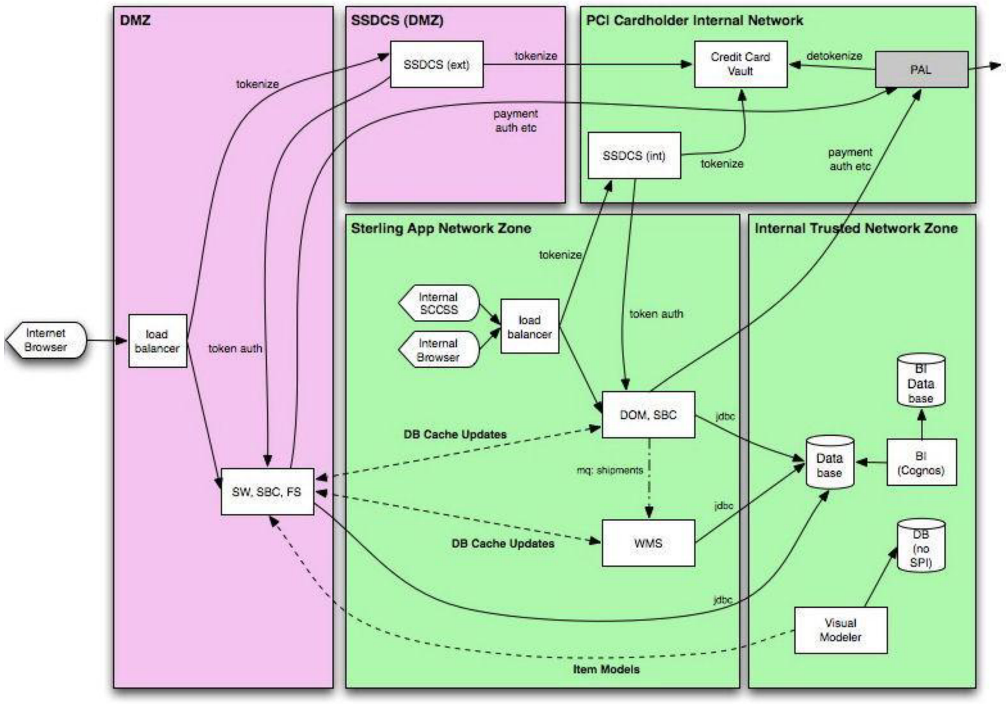


Diagram Key: Abbreviation	Product, Application, or Type of Data
SSDCS	Sterling Sensitive Data Capture Server
PCI	Payment Card Industry
PAL	Payment Abstraction Layer
SCCSS	Sterling Call Center and Sterling Store
DOM	Sterling Distributed Order Management
SW	Sterling Web
SBC	Sterling Business Center
SFS	Sterling Field Sales
WMS	Sterling Warehouse Management System
SPI	Sensitive Personal Information
BI	Sterling Business Intelligence

In this deployment strategy, you must open up firewall ports to allow inter-application communication between the Sterling Call Center and the Sterling Distributed Order Management applications.

Opening Firewall Ports for Intra-app Communication

This topic applies to you if you implement two or more Sterling Selling and Fulfillment Suite applications in an integrated manner such that:
- they share the same database, and
- some of the applications have to communicate through firewalls.

If this scenario applies to you, you have to open firewall ports to allow intra-application communication.

Intra-app Communication

"Intra-app communication" refers to management messages and notifications that flow between the Sterling Selling and Fulfillment Suite applications. Some of these intra-app communications include:
- Database cache notifications
- System management console commands

The intra-app management messages and notifications are implemented using Remote Method Invocation (RMI). When an IBM agent or application server starts up, it first creates an RMI method that listens on a port. Next, it registers that RMI service in the YFS_HEARTBEAT table. Agents or application servers that need to send management messages to its peers iterate through the YFS_HEARTBEAT table, picks up the RMI information, and then calls the RMI service of its peers.

By default, the port the RMI method listens-on is random. To enable intra-app communication, one would have to open up a wide range of ports.

Deterministic RMI Port Feature

The Deterministic RMI Port feature allows you to specify a range of ports that your JVMs can bind to so that you can control the number of firewall ports to open. To specify the range, set the rmi.portrange parameter in the YFS_PROPERTIES file. You can find instructions on how to set this parameter in the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

For example, if you set rmi.portrange=27001-27050, each JVM will randomly pick a number in that range to bind to. If that port is busy or in use, the JVM will then try each number in the range in sequence until it finds a free port. If it reaches the end of the range, it will wrap around to the beginning of the range and continue trying. If it reaches its original starting point, the JVM gives up and will not start up.

For backward compatibility, RMI will be allowed to choose any valid port number. To do this, either do not set the rmi.portrange parameter or set rmi.portrange=0.

Configuration Recommendations

On one hand, you should specify a small port range to reduce the number of ports you open. However, you also need to specify a big enough port range to accommodate the highest concurrent ports used from any single computer node governed by that port range. When estimating your port ranges, consider the following:

- You need to estimate the largest number of JVMs you expect to start in a single computer node or more precisely from one IP address. Set rmi.portrange to a number that is larger than that estimate.
- You need to configure the firewall that sits between the Sterling Selling and Fulfillment Suite applications to allow that intra-app communications across the port range above.

You can use the customer_overrides.properties file to selectively set your port range to suit your operational needs. For example, you may run a few (for example, up to 5) JVMs on computer nodes that are dedicated to running application servers. Conversely, you may run a large number (for example, 20) of JVMs on computer nodes that are dedicated to running the Sterling Integration Servers. Using customer_overrides.properties, you could set a low value for the application servers and a higher number for the integration servers.

How Port Selection Works

The application selects a starting point at random in the list of ports, and tries to create an RMI stub at that port number. If the operation fails because the port is in use, the application moves on to the next port in sequence, wrapping at the beginning. If the application reaches the starting point without a successful export, it will throw an exception and exit.

Note: A random starting point is used to reduce the chance of conflicts when multiple JVMs are running on the same server.

For tracking purposes, an INFO-level message with the port number is printed after binding to a specific port when a range is requested. This does not apply when the port number 0 (default) is used, because that is internal to the RMI implementation.

Troubleshooting

If your JVMs cannot start because they cannot find a free port, you can use network tools like netstat to see which port numbers are busy. You can also use tools like lsof to find out which process is listening on the port number and the peer it is talking to.

Limitations

The following limitations apply to use of an RMI port range:

- The internal and external IP addresses of the hosts (from the point of view of the network zones) must be identical. Effectively, this means that Network Address Translation (NAT) or Secure NAT (SNAT) cannot be used. The reason is that JVMs register their RMI stub with their network address. When a JVM picks up an RMI stub to call, the network address must be reachable from that JVM.
- All existing RMI configuration requirements apply. For example, a server's hostname must resolve to the external IP address.

Chapter 5. Securing Sterling Selling and Fulfillment Suite Applications

Application Security Introduction

This section presents steps that you need to implement or configure to secure theSterling Selling and Fulfillment Suite applications.

The Sterling Selling and Fulfillment Suite applications rely on the following security controls:

- Authentication
- Authorization
- Configuration
- Logging

Authentication and Authorization

Authentication

Authentication controls whether a user is given access to portions of the application. It also provides a security control to deny attackers access to the application. The Sterling Selling and Fulfillment Suite applications use one or both of the following:

- Application-based authentication — In the Sterling Selling and Fulfillment Suite applications, you define users in the Applications Manager. A user is a single person assigned with a certain task, such as Hub Administrator or Customer Service Representative, depending on what role they play in the organization. Users can be assigned to one or more user groups and teams.
- Application server security — Application servers allow you to create users. These users could be used to control access to application server resources such as the Java Messaging Service (JMS) queues.

Authorization

Once a user's identify has been authenticated, authorization ensures that users with sufficient privileges can run certain code or get to certain data. In the Sterling Selling and Fulfillment Suite applications, user groups control access to code (or APIs) and teams control access to data.

User groups are a collection of users who perform a similar task or are granted similar security privileges. For example, a group of customer service representatives might be put in a Customer Service Representative user group. Users can belong to multiple user groups to which permissions are assigned. A user who belongs to multiple user groups retains the least restrictive set of permissions defined by the groups they belong to. For example, if a user belongs to a user group that permits the user to use the Application Console, and this user also belongs to a user group that permits the user to access only the Application Console and Applications Manager, the user has access to both applications.

User groups can be used to control access to application API's (such as the getOrderList API), HTML inner panels and the Interoperability Servlet.

A Team is a collection of users who have common data access requirements. Teams, for example, can have access to specific document types, Enterprises, ship nodes, and customers. Teams can be assigned to specific customers.

Creating a team is an optional process. If a user is not associated with a team, that user is considered to have the least restrictive access, or default access to customer orders and information. By defining a team, you can further restrict access to any Enterprises, document types, or participating ship nodes that are a subset of the default access list.

See the following guides for more information:
- *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide* for instructions on how to define and configure users, user groups, and teams
- *Sterling Selling and Fulfillment Foundation: Product Concepts Guide* for information on how you can use users, user groups, and teams in the Sterling Selling and Fulfillment Suite applications

Recommendations

Both authentication and authorization form the cornerstone of theSterling Selling and Fulfillment Suite applications security offering. You should follow industry best practices around authentication and authorization.

Removal of Default Sterling User Accounts

When installing the Sterling Selling and Fulfillment Suite applications, you must (at a minimum) load the factory default, which is a set of mandatory out-of-the-box configuration data. You can optionally load reference implementation data, which is a set of sample data containing sample organizations, sample items, and so on.

Both the factory defaults and reference implementation set up user accounts. You should not implement the optional reference implementation into your production system.

Securing JMS Integration

As stated previously, Sterling Selling and Fulfillment Suite . The Suite includes the following applications (the “ applications use Java Messaging Services (JMS) for many reasons, including:
- Intermediate message queue for interoperability or integration reasons
- Work-in-progress messages by theIBM agents

The queues are unfortunately implemented without security, which potentially would expose them to unauthenticated access. One way to prevent this is to secure access to the JMS resources. Once you have secured access to the JMS resources, you will have to configure the agents so that they authenticate to the queues.

Related Examples

Examples of how to secure JMS resources can be found in these topics:
- "Securing JBoss Messaging"
- "Securing Oracle WebLogic Messaging"
- "Securing TIBCO Messaging"

- "Securing IBM WebSphere MQ Messaging"

JMS Security

The JMS security for an IBM agent or service to authenticate to the JMS queues can be enabled in one of the following ways:

- By adding the JMS security parameters in the agent criteria or agent Java command line.
- By setting the JMS security parameters in the customer_overrides properties file.

Setting up JMS Security Using Agent Criteria Or Java Command Line

The steps for an IBM agent or service to authenticate to the JMS queues are enabled by adding the following two parameters into the JMS Security properties dialogue in the agent criteria:

- sci.queuebasedsecurity.userid
- sci.queuebasedsecurity.password

JNDI Security

Enable Java clients to authenticate to the JNDI by adding the following two parameters when creating the initial context:

- java.naming.security.principal
- java.naming.security.credentials

For IBMagents and services, you should add the two parameters above into the JMS Security properties dialogue in the agent criteria. Instructions on how to set up JMS security in the agent criteria are documented in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

SSL Channel Encryption

Finally, in order to protect messages transiting through your network, you should ensure that your client programs use SSL to encrypt communications with message queues. Note that SSL can be used to protect messages that are in-flight or in-motion across the network; however, it does not protect messages in queues when they are at rest.

To initiate a one-way SSL session from an IBM agent or service to a JMS server, set the following two -D options on the Java command line of the client:

- javax.net.ssl.trustStore
- javax.net.ssl.trustStorePassword

When using the IBM JDK, you can add the following -D property to help you diagnose your SSL setup:

```
ssl.debug=true
```

Securing JMS Queues

The implementation of JMS security is left to the discretion of the application server developers. As a result, the requirements differ from one vendor to the next. If using these vendors, note the following:

- TIBCO — If queue security is enabled at the TIBCO server, the client must pass both queue security parameters and jndi security parameters. Also, TIBCO client configuration for SSL is specific to that server.
- Oracle WebLogic —

  Add the wlcipher.jar and wlfullclient.jar WebLogic JARs to your client side CLASSPATH.

  With WebLogic, queue security parameters have no bearing. If a queue is secured in the WebLogic console, the JNDI security parameters must be set up at the client. Queue security parameter can be set (or not set) on the client without changing the outcome.

Setting up JMS Security Using customer_overrides.properties File

You can set the JMS security parameters for an IBM agent or service to authenticate to the JMS queues in the customer_overrides.properties file.

To enable Java clients to authenticate to the JNDI set the following two parameters:
- java.naming.security.principal
- java.naming.security.credentials

To initiate a one-way SSL session from an IBM agent or service to a JMS server, set the following two parameters:
- javax.net.ssl.trustStore
- javax.net.ssl.trustStorePassword

When using the IBM JDK, you can also set the following parameter to help you diagnose your SSL setup:

ssl.debug=true

To enable an IBM agent or service to authenticate to the JMS queues set the following two parameters:
- sci.queuebasedsecurity.userid
- sci.queuebasedsecurity.password

For additional information about overriding properties using the customer_overrides.properties file, see the Applications: *Properties Guide*.

Securing the Interoperability Servlet

The Interoperability Servlet is one of the integration technologies that allow client programs to interact with Sterling Selling and Fulfillment Suite applications. In this topic, we will discuss how you can secure access to the Interoperability Servlet through authentication, authorization, and confidentiality.

Parameters

The following parameters, at their default values, enforce authentication and authorization:
- interopservlet.security.enabled=true
- interopservlet.auth.container.enabled=false
- interopservlet.auth.token.enabled=true

- interopservlet.auth.userPassword.enabled=true

See the*Sterling Selling and Fulfillment Foundation: Properties Guide* for more information.

Authentication

By default, with both interopservlet.security.enabled and interopservlet.auth.token.enabled set to true, client programs must supply a userid and password for authentication. The user is defined in the Applications Manager. The process of creating a user is described in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Authorization

You can also configure user groups to control which users have access to APIs (for example, to create orders or to view orders) as well as to the interoperability servlet itself. See "Setting Up Permissions for APIs" and "Setting Up Permissions for Interoperability Servlet" in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*

Confidentiality

Finally, you should send requests to the Interoperability Servlet under SSL to protect the confidentiality of the request when it is transiting through the network.

Reduction of Attack Surface

We recommend you remove the interoperability servlet from your deployment if you do not intend to use it. Doing so reduces the attack surface of the application.

Securing Web Services

Web services is one of the integration technologies that allow client programs to interact with theSterling Selling and Fulfillment Suite applications. In this topic, we will discuss how you can secure access to Web services through authentication, authorization, and confidentiality.

Sterling Selling and Fulfillment Suite APIs and services can be invoked as Web services.

Steps to Enabling Web Services

By default, the Sterling Selling and Fulfillment Suite APIs and services are not exposed as Web services. To enable Web services, first define which APIs are exposed through the namedwebservices.xml file. In the XML, we strongly recommend you set the attribute ExposeAllAPIs to "N" to allow you to select which APIs are exposed. A setting of "Y" means that all APIs are exposed.

Note: At a minimum, you will need to expose the login API for authentication.

The *Sterling Selling and Fulfillment Foundation: Installation Guide*provides detailed instructions on how to enable Web services.

Authentication

By default, the APIs and services will not run without a valid user token. That token is obtained when you call the login API. Once the token is obtained, you set the token as the tokenId in YFSEnvironment and then call the APIs your user has access to. The user is defined in the Applications Manager. The process of creating a user is described in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*. For details about the YFSEnvironment interface, see the*Sterling Selling and Fulfillment Foundation: Javadocs*.

Authorization

You can also configure user groups to control which users have access to APIs (for example, to create orders or to view orders). Please see "Setting Up Permissions for APIs" in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Confidentiality

Finally, you should send requests to Web services under SSL to protect the confidentiality of the request when it is transiting through the network.

Reduction of Attack Surface

We recommend you do not configure Web services into your deployment if you do not intend to use it. This will reduce the attack surface of the application. To further assure that Web services are not configured, take the following actions:

- Pass the -Dnowebservice=true flag into the buildear.sh. This instructs the EAR building process not to build the Web services components.
- Ensure that the namedwebservices.xml file is not defined.

Securing EJB Interoperability

Enterprise JavaBeans (EJB) is a core J2EE technology. It is also another of the integration technologies that allow client programs to interact withSterling Selling and Fulfillment Suite applications. In this topic, we will discuss how you can secure access to EJB through authentication, authorization, and confidentiality.

At a high level, to invoke EJB, you first have to perform a Java Naming and Directory Interface (JNDI) lookup to find the appropriate home stub. Given the home stub, you then make a Remote Method Invocation (RMI) call to the stub. The RMI stub on the application server, in turn, calls the appropriate XAPI. By default, theIBM application XAPIs are exposed as EJBs when the EAR is built.

EJB Implementation

Currently, the IBM applications expose two different types of EJBs. First, there is a variant of EJB where parameters (such as the YFSEnvironment) are passed in as w3c Documents and results are also returned as w3c Documents. Technically speaking, this variant of the EJB is not compliant with the EJB specification, since the w3c Document is not serializable, as required by the EJB specification. The second variant of our EJB implementation accept string versions of the parameters and returns a string.

Note: With some appservers, warnings will be thrown if the Document-based EJBs are deployed. These warnings can be avoided by suppressing deployment of these EJBs.

Security and EJB

As discussed earlier, an EJB invocation involves a JNDI lookup and an RMI invocation. From a security perspective, you should first secure access to the JNDI so that only authorized users can lookup EJB. Secondly, you should ensure the JNDI transactions are performed under SSL to prevent attacks such as network sniffing. Once you have found the EJB stub, you then need to secure access to the RMI invocations. The RMI component of EJB can be secured through SSL.

Note: Some application servers might provide proprietary support for user authentication and authorization to the EJBs themselves. Those features are application server-specific and beyond the scope of this document.

Invoking EJBs

IBM provides a toolset called the YIFClient to invoke XAPIs from a Java client program. For more details on how to invoke EJB via the YIFClient toolset, refer to "Invoking APIs from the Client Environment" in the *Sterling Selling and Fulfillment Foundation: Customizing APIs*. In addition, see "Securing Java Protocols" in the*Sterling Selling and Fulfillment Foundation: Installation Guide* for more information on how to call the EJB securely through the YIFClient.

Reduction of Attack Surface

We recommend you do not configure EJBs into your deployment if you do not intend to use it. Doing so will reduce the attack surface of the application. In order to suppress EJB generation, pass the -Dnoejb flag sed during the EAR build.

Securing Property Files

Sterling Selling and Fulfillment Suite applications use property files to store parameters that control the way the application works or that are used to obtain credentials to access critical resources.

Threat

Malicious users who gain access to property files can obtain sensitive information such as user names, credentials, or IP addresses.

Mitigation

IBM recommends this approach to secure property files:

1. Restrict access to property files.

   Place property files on file systems that have restricted file privileges to limit which applications and users can access them.
2. Encrypt sensitive information in the property files.

   Sterling Selling and Fulfillment Suite applications allow you to encrypt parameter values.

Property Parameter Encryption

To encrypt properties, you need to indicate which property has encrypted values and provide an encrypter class. See the *Sterling Selling and Fulfillment Foundation: Properties Guide*and the*Sterling Selling and Fulfillment Foundation: Extending Transactions* for more information.

Securing Logs and Traces

Sterling Selling and Fulfillment Suite applications have a rich set of logging and tracing facilities. For example, from the System Management Console, you can dynamically activate detailed traces on any API or screens on the running system.

Threat

It is important that you secure your log and trace files. Logs may:

- Contain information (for example, stack traces) that can help malicious users understand how the application works
- Contain sensitive personal information that attackers would like to collect
- Be needed for forensics or for breach investigation. Malicious users may want to tamper with log files to hide their activities.

Mitigation

At a minimum, IBM recommends the following:

- Store your log files into directories that have restricted access. For example, the file system should allow the application to write logs, but not to read them. The file system should also restrict who can read and copy the logs.
- Ensure that access to the System Management Console is restricted to privileged individuals to prevent attackers from starting traces.
- During your system test, enable all traces at the highest level. At the end of the test, make sure you understand which API's or screens log out sensitive information.

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

Trademarks

Product Number: xxxx-xxx

Printed in USA