Technical Connectivity Guide for OnDemand - OnPremise Hybrid Applications
Document History

Before you start working with this document, make sure you have the latest version. You can find the latest version on the Business Center for On-Demand Solutions from SAP.

For questions and feedback, please contact the author:
joerg.nalik@sap.com

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2012-01-30</td>
<td>Initial version</td>
</tr>
<tr>
<td>1.1</td>
<td>2012-02-29</td>
<td>Copy-edited version; no content changes – final draft version</td>
</tr>
<tr>
<td>1.2</td>
<td>2012-06-25</td>
<td>New template applied; released for CUSTOMER; no content changes</td>
</tr>
<tr>
<td>2.0</td>
<td>2014-08-19</td>
<td>New updated and extended version</td>
</tr>
</tbody>
</table>
## Table of Contents

1. **Introduction** .......................................................................................................................... 4

2. **Technical Connectivity Architecture** ...................................................................................... 7
   2.1 SAP's Network Technology Partners and Certifications ......................................................... 11

3. **Understanding Network Traffic Security** .................................................................................. 12
   3.1 Introduction to Common Encryption Schemes ........................................................................ 12
   3.2 Key Stores ..................................................................................................................................... 14
   3.3 Certificate Authorities, Management Tools and Cryptographic Software Sources .................... 15

4. **Other Network Technology Aspects** ....................................................................................... 16
   4.1 Network Addressing .................................................................................................................. 16
   4.2 Advanced Network Technologies Sometimes In Use .................................................................. 18

5. **Checklist for Project Managers** .................................................................................................. 21
   5.1 Implementation Team .................................................................................................................. 21
   5.2 Breakdown of Implementation Tasks ......................................................................................... 22
   5.3 Overview of Key Stores and Certificates ................................................................................... 23

6. **How to Implement** ...................................................................................................................... 25
   6.1 Security Implementations in a NetWeaver ABAP-Based Application Back-End System .......... 25
      6.1.1 Enable SAP back-end application systems for secure communication .................................. 25
      6.1.2 Set up your SAP back-end system as secure https server ......................................................... 26
      6.1.3 Import Client Certificates into your ABAP system for Client authentication and link them to a user for authorization purposes ................................................................................. 33
      6.1.4 Web Service Security Configuration ....................................................................................... 38
      6.1.5 Web Service Authentication Testing ....................................................................................... 42
      6.1.6 Configuring Authentication of a Trusted Reverse Proxy in an ABAP System ......................... 46
      6.1.7 Certificate Life Cycle Management ......................................................................................... 48
   6.2 Implementing a Secure Reverse Proxy ......................................................................................... 48
      6.2.1 Configuring Third Party Reverse Proxies ............................................................................... 49
      6.2.2 The Easy Way to Configure SAP WebDispatcher as a Reverse Proxy ................................. 50
      6.2.3 Advanced SAP Web Dispatcher Configuration Options ......................................................... 61
      6.2.4 Desirable Security and Other Configurations for the SAP Web Dispatcher ........................... 66
      6.2.5 SAP Web Dispatcher Monitoring and Administration Web Interface .................................. 69
1 Introduction

This document gives an overview and deep dive about technical connectivity for cloud solutions from SAP that are integrated with existing customer SAP on-premise (OP) products, such as SAP ERP. We refer to solutions that run partially in the cloud and in a customer's datacenter as hybrid applications.

⚠️ Caution

This document recommends how to set up the technical connectivity for cloud solutions from SAP that are integrated with existing customer SAP on-premise (OP) products, such as SAP ERP. SAP is not liable for any consequences or damages resulting from the use of this document.

At the time of writing, this document shows examples of these cloud solutions from SAP:

- SAP Business ByDesign,
- SAP Cloud for Customer,
- SAP Cloud for Travel and Expense,
- SAP Cloud for Financials,
- SAP Cloud for Sales,
- SAP Cloud for Service,
- SAP Cloud for Social Engagement,
- SAP Cloud Applications Studio,
- SAP Cloud for Marketing

SAP's cloud offerings are expanding rapidly and more SAP Cloud solutions might have been added by the time you read this document.

In contrast to application integration within one data center, the integration of application components in hybrid landscapes requires the penetration of network security perimeters of the customer data center and the cloud environment, connectivity via wide area networks (WAN) of the Internet (as opposed to just LAN - local area network - connectivity) and network traffic encryption for security. In addition, business applications always require the authentication of a client with a subsequent authorization step to determine whether it is permitted to process requests and to deliver the resulting business data back to a client.

For security reasons, we do not recommend that you connect on-premise back-end systems directly to the Internet. Therefore, most of this document focuses on the use of a reverse proxy in a customer's datacenter network when cloud applications need to request data from a customer's back-end systems.

Here's a list of the aspects we cover:

- Introduction to requirements and technical concepts for Cloud-OP connectivity.
- Checklist for Cloud-OP implementation project planning: save time by knowing up front all the steps you need to perform.
- Detailed how-to description of the steps needed to implement Cloud-OP connectivity.

Target Groups

This document contains an overview as well as deep dive because a number of topic areas need to be addressed for technical hybrid application integration, for which a number of people from different areas need to collaborate:

- Architects and program leaders, who oversee end-to-end hybrid application integrations (chapters 1+2)
- Collaborating technical experts for SAP platform, network infrastructure and end-to-end security (chapters 3+4)
We recommend that SAP application project teams use this document to clearly communicate with customer network and security IT groups in order to coordinate the necessary work and to safeguard that the achieved security levels match the requirements and security policies customers have set for themselves.

Scope

Technical connectivity is part of a 3-layer integration architecture. The top layer is the integration of business scenarios across business software solutions as is the case for hybrid scenarios or so called business network hubs such as Ariba for Supplier/Buyer integration or Financial Service Networks (FSN) for customer/financial institution integration.

Next, customers typically implement middleware components or services for application connectivity. Application connectivity typically addresses:

- Application level metadata and protocol transformations
- Persisting of messages for the purpose of guaranteed message delivery, often enhanced with a guarantee of in order delivery too.
- Application Program Interface (API) governance and management, such as the application of application-specific message filtering policies.

Business scenario integrations are very specific for individual use cases. Application connectivity is very specific to individual application and middleware components. Therefore, they are typically described in detail in specific individual product documentation.

In contrast, the third technical connectivity layer, which deals with network technology for technical network and location integration, is mostly standard-based and implemented according to common best practices. With this in mind, we offer this generic technical connectivity guide for all Cloud to customer on-premise datacenter spanning integration scenarios. We assume that customers have already established network operations for their company, meaning they have a company network infrastructure and Internet access capacity. This guide focuses on Layer 4-7 network technologies defined in the OSI model (see for instance [http://en.wikipedia.org/wiki/OSI_model](http://en.wikipedia.org/wiki/OSI_model)).

The central component to discuss is the implementation of a reverse proxy as a central network component or service that acts as a gateway between public Internet networks and a company's private network infrastructure for applications.

This document does not cover the following issues:

- End-user - meaning browser and mobile - technical connectivity with SAP Cloud or customers' on-premise systems
- Single sign-on and identity management.
- Implications of data privacy laws in the various countries where you operate.

For SAP Business ByDesign based cloud solutions, this security guide complements the technical connectivity guide and covers end-user connectivity security and more.
Assumptions and Prerequisites

- You plan to integrate an SAP Business Suite system, such as SAP ERP, with an SAP Cloud Solution.
- You have access to SAP Service Marketplace; that is, you have S-user access to SAP Service Marketplace.
- You have access - that is, you have licenses - for the download of SAP NetWeaver technology components from the SAP Software Download Center.
- You know experts who can configure SAP systems on operating system and SAP Basis level.
- Likewise you have cooperation contacts to IT security and network departments for the on-premise side.
- Your company security policy allows communication between your on-premise SAP systems and SAP Cloud environment over the Internet.
- You have the means of obtaining public IP addresses and Domain Name Service entries visible on the public Internet.
- You have the means of obtaining network security certificates from a company internal or public Certificate Authority according to the security policies of your company.
2 Technical Connectivity Architecture

As cloud adoption is rapidly growing, SAP’s customers want to take full advantage of their existing investments in SAP software running in their own on-premise (OP) datacenters by integrating it with new SAP Cloud based applications.

Your network infrastructure is extremely important for reliable integration of your OP/Cloud application components and services, and to ensure that they are fully secured and perform well.

- You need a **reliable network** so that your application is always available.
- You need **security** so that your confidential business data is protected.
- You need good **performance** to save costs and satisfy end users with a good user experience.

Security can be further broken down into the areas of data access, data transport, and data storage. While security of data storage is taken care of through backup and similar measures on the application side, data access and transport security is a service of your network infrastructure.

A well-defined network topology can eliminate many security threats based on software flaws (at both the operating system and application level) or network attacks such as eavesdropping. If users cannot log on to your application or database servers at the operating system or database layer, then there is no way for intruders to compromise machines and gain access to the SAP System database or files. Additionally, if users cannot connect to the server LAN, they cannot exploit well-known bugs and security holes in operating systems on the server machines.

Again, your strategy and your priorities are the most important factor in deciding what level of security is necessary for your network infrastructure. We offer general recommendations when establishing your network topology.

**Note**
Depending on your current situation, you may want to modify the secure network setup described here to fit your needs. We offer such suggestions and recommendations at various security levels. If the plan described here does not fit your needs, contact our consultants, who are also available to assist you in setting up your network securely.

On-premise-Cloud application integration scenarios require communication over the Internet that is both secure and authenticated to prevent unauthorized access to business critical SAP on-premise back-end systems of our customers. A “trust relationship” needs to be built up between the OP and OD site and the data transport has to be encrypted.

There are 2 cases to be distinguished:
1. An SAP Cloud-based application is sending a request to the On-Premise customer site
2. An On-Premise application is sending a request to the SAP Cloud

For case 1, we will use this paper to look out this can be realized technically, using reverse proxies at the OP site. A reverse proxy acts as a gatekeeper above the level of typical firewalls by intercepting, inspecting, and modifying traffic in many ways. If a connection from the Internet into your OP datacenter should be under attack, a reverse proxy will simply take the heat and thus protect SAP back-end applications in your innermost datacenter network security zones from direct attacks.

Case 2 just requires a call out of an SAP application to the Internet. Security concerns for such connections are typically much less. You might just want to use a forward proxy for network address translation so that your inner network server addresses are not transmitted to the Internet. In this paper, we focus only on the case 1 above.

We define a reverse proxy as a network service, which receives https requests from the Internet, applies certain network security policies, and then forwards requests to the SAP back-end application systems. The most basic security policies required from a reverse proxy are depicted in the picture above:

1) Route Through

2) Terminate SSL

3) Re-Encrypt

We recommend implementing case 3. Case 2 is not secure against company internal attackers, because attackers could monitor internal unencrypted network traffic, retrieve the legitimate clients certificate and then use that certificate to impersonate them towards the back-end system. You need also think about how trust relationships
from the SAP Cloud clients are extended all the way to the SAP back-end applications. As we will show in a later chapter of this guide, the security implementation for web service calls in an ABAP system is such that client authentication and encryption are very tight together.

**Note:** We recommend that you include ABAP-based back-end systems in the overall trust relationships between components of technical connectivity. If you follow this recommendation, case #3, encrypted traffic anywhere, is your only option.

There are many more network security policies and services known for edge networks, the boundary between inner datacenter application network zones and the Internet. Here are just a few of them:

- Distributed Denial of Service defense
- Intrusion Detection and Prevention Systems (IDS/IPS)
- Cross-site scripting prevention
- Web Service call inspection and filtering systems for SOA Governance
- Load Balancing to back-end application servers
- Traffic flow enhancements – for example, through filtering and fixing Internet quality issues

Customers should bring together business representatives, application, network, and security groups in their organization to decide on security investments for their business needs.

Often such network technologies are bundled together into “Application Delivery Controller” (ADC) devices (see Gartner: [https://www.gartner.com/doc/2615518](https://www.gartner.com/doc/2615518)). Any ADC always acts as reverse proxy at a minimum and then might add different security capability. Therefore, we focus in this paper on the reverse proxy and SAP application server interactions and the security configurations, which need to be done in conjunction with each other in ADCs/ reverse proxies and SAP application servers.

As shown in the first figure, the basic network traffic encryption over the Internet has to be complemented by an authentication mechanism to establish “trust”. It is not enough to protect traffic from eavesdropping through encryption; you also need to ensure that a legitimate external client is authorized to access your application back-end system and to receive confidential business data in responses. Authorizations in applications are typically pinned to user identities, so client authentication is needed.

In this paper, we will describe how x.509 certificate-based authentication can be used for application-to-application web-service based integrations between the SAP Cloud and customers on-premise systems. The x.509 certificates are specific to a particular client. That way, they can be used to identify a client. SAP back-end application systems have their built-in (typically ABAP) identity management system. For application to application integrations, a technical user can be set up which is then linked to a specific x.509 certificate (see the picture above). That way, a web-service request from an SAP Cloud client can be given the authorization of a specific technical user, which is set up in such a way as to allow execution of the business application logic needed for serving the cloud request.
The X.509 authentication mechanisms (see also X.509 Client Certificates on help.sap.com), Internet traffic encryption and the security shielding functions of a reverse proxy can be combined as shown in figure 4. Since a reverse proxy is terminating an https connection from a SAP Cloud client, it needs to forward the Cloud x.509 certificate to the back-end system even in cases where the reverse proxy itself uses only an http connection to the back-end. For this purpose, the SAP back-end systems can receive x.509 certificates also embedded as http request header field. The reverse proxy used for SAP Cloud to on-premise integration scenarios would need to support copying SSL x.509 client certificates into http header fields.

The important http header fields a reverse proxy has to set for x.509 forwarding are:

<table>
<thead>
<tr>
<th>Name of Header Field</th>
<th>Value</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_CLIENT_CERT</td>
<td>Client certificate</td>
<td>Base64-coded data, one line</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT_CHAIN_1</td>
<td>First non-root CA certificate issued by the client.</td>
<td>Same as SSL_CLIENT_CERT</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT_CHAIN_n</td>
<td>Last non-root CA certificate</td>
<td>Same as SSL_CLIENT_CERT</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT_CHAIN_n+1</td>
<td>This element is the corresponding root certificate. It must not be transferred to the server.</td>
<td>Same as SSL_CLIENT_CERT</td>
</tr>
<tr>
<td>SSL_CIPHER_USEKEYSIZE</td>
<td>Number of encryption bits actually used</td>
<td>Numeric decimal value transferred as a string (for example 128 or 256)</td>
</tr>
<tr>
<td>SSL_CIPHER_SUITE</td>
<td>Cipher suite in accordance with the definition in the SSL and TLS specification (draft free ssl version 3-02 or RFC 2246)</td>
<td>The cipher suite consists of two (SSL 3.0 / TLS) or three (SSL 2.0) numeric 8 bit values.</td>
</tr>
</tbody>
</table>

In order to protect from “insider threats”, encrypted https traffic between reverse proxies and back-end application systems might be required too. Insider threats can come, for example, from rogue employees with access to your company intranet. Nobody with access to your company internal network should be given a chance to eavesdrop or manipulate traffic.

As the reverse proxy to back-end traffic is encrypted, a trust relationship can be established here as well. The reverse proxy is a client to the back-end system and might become a “trusted intermediary” for SAP Cloud traffic by using its own client x.509 certificate to be authenticated by the back-end system as allowed proxy in the communication chain.

For more information, see also the SAP NetWeaver Network Security Guide:

- On SAP Service Marketplace at: https://service.sap.com/security
On the SAP Help Portal, for example, for SAP NetWeaver 7.3 EHP 1 at:
http://help.sap.com/saphelp_nw73ehp1/helpdata/ en/0a/0a2e00ef6211d3a6510000e835363f/content.htm

2.1 SAP's Network Technology Partners and Certifications

SAP collaborates with the leading network vendors and offers a network technology certification program for their products. Network certifications are granted in 3 categories:

- Reliability Enhancements, for example, thorough reverse Proxy and load balancing capabilities
- Security Enhancements: for supporting the SAP Cloud to on-premise application authentication and encryption methods described in the following in this paper or for other added network security capabilities.
- Performance Enhancements: Network performance optimization through means like content caching, TCP/IP traffic flow optimizations and symmetric WAN acceleration and Content Delivery Networks (CDN).

Reliability and Security enhancements are typical benefits of ADC products. We recommend that customers check the SAP Integration and Certification Center (SAP ICC) website for details about the network product certification program. For an up-to-date listing of SAP certified network products, go to the SAP Application Development partner directory site. Older certifications can be found there with the keyword search "esoa-aw-". Since Q3/2013, a new version of network product certification is offered and companies with certified products can be found with the keyword search "netwk-".
3 Understanding Network Traffic Security

3.1 Introduction to Common Encryption Schemes

Certificates need to be deployed on both sides before you can enable encrypted communication between a server and a client. To understand what exactly needs to be done, it is helpful to briefly review the fundamentals of encryption technology in order to also define common terms used in many deployment guides.

The basis of https is public key cryptography (http://en.wikipedia.org/wiki/Public_key), which is based on the idea that a message can be encrypted and decrypted with help of a private-plus-public key combination in such a way that public keys can be distributed to communication partners freely, while private keys are kept secret by the owner. That way, only the owner of the private key can decrypt a message.

Keys are just strings generated by key generation software in pairs consisting of a public and a private key. In a second step, the public key, which is shared between communication partners, needs to be associated or bound to the key owner's identity. If you want to send somebody an encrypted message, you need to ensure that the public key you got from your dedicated communication partner really belongs to that partner and nobody else.

Keys bound together with identities are called certificates (http://en.wikipedia.org/wiki/Public_key_certificate). Most browsers allow viewing certificates. One example is shown in the picture above. The "subject" is the field that specifies the key owner. The "Public key" field contains the key, which is a long binary string as shown in the details tab.

You should note that certificates have expiration dates, which are specified in the "Valid to" field. With this in mind, you should establish a certificate management practice in your organization, which ensures that certificates are renewed in good time, before they expire. An expired certificate would be refused by clients and servers for building up secure connections and could effectively cause application downtimes.
Certificates are generated and their distribution is secured through a something called Public Key Infrastructure (http://en.wikipedia.org/wiki/Public-key_infrastructure). This uses a trusted third notary party as a main element to ensure correct, trustworthy ownership of certificates. The trusted third party is called a Certificate Authority (CA). From http://en.wikipedia.org/wiki/Certificate_authority:

"A CA issues certificates that contain a public key and the identity of the owner. The matching private key is not made available publicly, but kept secret by the end user who generated the key pair. The certificate is also a confirmation or validation by the CA that the public key contained in the certificate belongs to the person, organization, server or other entity noted in the certificate. In essence, the certificate authority is responsible for saying "yes, this person is who they say they are, and we, the CA, certify that".

Fig: Alice and Bob (http://en.wikipedia.org/wiki/Alice_and_Bob) and a trusted CA

The validation of ownership of public certificates by CAs itself needs to be secured. For that purpose, CAs themselves distribute their own public key containing certificates, which are referred to as root certificates (http://en.wikipedia.org/wiki/Root_certificate). In a typical client-server https communication the client software needs the root certificate of the CA, which signed a communication partner's public key certificate in order to validate its authenticity. All common browsers, the most common clients to application servers, are delivered with the root certificates of well-known commercial CAs. Herein lies the value of buying certificates from such CAs: All servers using public key certificates from a known CA can be used by common browsers and other clients, like the SAP Cloud application as a client, right away.

You also have the option of, basically, being your own CA and creating self-signed certificates (http://en.wikipedia.org/wiki/Self-signed_certificate) or of using free certificate signing CAs. This alternative is great for test and training purposes, since it helps you to save costs on certificates, but it is not recommended for on-premise to cloud solutions from SAP integrations. An unknown CA would cause problems with clients not automatically having the root certificate available to them. This problem can be only avoided if you control both the server and the client so that you can import the unknown CA root certificate into the client. This situation exists between the reverse proxy to SAP back-end communication. It might be OK to use self-signed certificates here.

Next, we need to become a bit more specific on whether Alice is calling Bob or vice versa. In IT terms, the caller is called client and the receiver is the server. The practical implementation for secure Internet connectivity is that the server presents his public key wrapped into its "server certificate". This way, a client can trust a server.

However, business applications need to authenticate their clients as well. A server can only check authorizations for authenticated clients and requests can only be processed if the authorizations are sufficient. The client authentication can be done through a user/password submission. Another alternative is the use of client certificates. Since a certificate contains a public key bound to the identity of the client, a client certificate can be
used for authentication. For this "mutual" two-way server and client authentication to work, the server also needs
the CA root certificate of the CA signing the client certificate in its key store.
In essence, with the use of client and server certificates the Bob/Alice conversation and certificate exchange
becomes symmetric. A typical use case for client certificates is end user Single Sign On (SSO) and the on-premise
to SAP Cloud integration described in this paper.

An important secure implementation step is the creating and signing of certificates with a Certificate Authority as
depicted in steps 1 and 2 of the figure before. The detailed process is, for example, described in
http://en.wikipedia.org/wiki/Certificate_signing_request . The steps to perform are:

1. The applicant (Bob, or rather Bob's application server in the figure above) creates a public/ private key
   pair.
2. The applicant computes a certificate signing request (CSR) out of his personal identifiers and the public
   key and signs it with its private key.
3. The CSR is sent to a Certificate Authority, which returns the signed certificate. Bob can then use it for
   encrypted message exchanges.

" PKCS #10" is the standard CSR format used for this procedure. The certificate format standard is called x.509.

3.2 Key Stores

In a productively used SAP application landscape, a number of certificates might be needed, persisted in "key
stores". Each server, proxy and communication client has its own key stores for multiple purposes.

As shown in the figure above, each component might act as a server or as a client depending on who is calling
whom and in which direction. It might also be the case that multiple communication channels like the one shown
above exist in parallel if multiple integration scenarios are used in parallel. Further, you have to consider that you
also need to store public root certificates of trusted Certificate Authorities as explained in the previous section.

The content of the server key store is at a minimum:

- The root certificates of the Certificate Authorities used for signing the server public key certificate and client
  public key certificate respectively
- The server's public key certificate
- The server's private key

Similarly, the content of the client key store is:

- The root certificates of the Certificate Authorities used for signing the server public key certificate and client
  public key certificate respectively
- The client's public key certificate
- The client's private key

Since the application server and reverse proxy are both owned and controlled by the customer, it is an option to
use the same content for the server key store of the application server and the reverse proxy as opposed to
having 2 sets of server key stores, one for each component. Through sharing server key stores the effort involved
in creating, deploying, and maintaining certificates can be lowered. However, sharing server key stores implies
sharing the secret private key inside them as well. Each customer should define his or her own security policy
about sharing server key stores among components in their SAP landscape. If server key store sharing is used, a
means of exporting and importing private keys alongside signed public key certificates is needed. Another
standard format, referred to as PKCS 12, exists for this; see, for example, http://en.wikipedia.org/wiki/PKCS_12.

SAP NetWeaver based components have their own key stores referred to as “Personal Security Environments” or
Similar reverse proxy solutions from various vendors and SAP’s WebDispatcher have their own key stores. The
SAP WebDispatcher can be used as reverse proxy and as a load balancer. As we will see later, entire PSEs can be
exported and imported from NetWeaver components, which is a convenient way to share the same server store
between an SAP back-end system and a reverse proxy in front of it. The SAP Web Dispatcher can directly use an
exported PSE server key store file from an SAP ABAP-based back-end system (but you may also create
WebDispatcher PSEs from scratch). For sharing ABAP server key stores with the reverse Proxy products of SAP’s
network technology partners PKCS 12 certificates can be exported from PSE files, see

3.3 Certificate Authorities, Management Tools and
Cryptographic Software Sources

For a hybrid on-premise to SAP Cloud deployment project, it is important that the same Certificate Authority a
customer is using is also accepted by the SAP cloud solution(s), which is(are) part of the integration scenario. For
a list of supported CAs by SAP Cloud solutions, please check the administration sections of the Cloud solution you
are using. In some instances, cloud solutions from SAP also allow the import of certificates including new CA root
certificates.

SAP itself offers a certificate signing service on its Service Marketplace websites (s- or p-user login required) in
the Trust Center Service (TCS) section at http://service.sap.com/tcs. You can purchase longer term valid
certificates as well as generate free of charge test certificates, which are valid for 8 weeks. In the TCS “download
area” you can also find the root certificates for SAP’s CA services.

While SAP’s CA service is, of course, supported by SAP Cloud integrations, it is not one of the pre-set CAs in
common browsers. Therefore, if you use SAP CA signed server certificates for web user interface encryptions
also, you need to import the SAP CA root certificate into the browser’s trusted authority key store in order to
avoid security warnings being shown in the browser.

Of course there are also Open Source and other free of charge CA offerings available, which might be useful for
security training and testing purposes. However, their support for productively used SAP Cloud integration
scenarios and by common browsers might be very limited.

The cryptographic software for SAP NetWeaver components is subject to international export controls and so has
to be downloaded as separate package from the SAP Service Market Place. You find the SAP Cryptographic
Library in the TCS download area as well. In newer ABAP kernel releases, this library is included as described in
SAP note 1848999.

SAP software components as well as third-party proxy solutions have their own in-built certificate and key store
management tools. You need to use those tools during hybrid scenario deployment projects and should ensure
that skilled personnel are available for this task.

For most integration scenarios with SAP back-end systems, the NetWeaver ABAP stack transaction STRUST (see
details at Trust Manager in the SAP Help Portal) should be used to manage key stores or PSEs in those
components. PSEs, which are stored in the database of an SAP ABAP-based system, can be exported into a file.
The command line tool sapgenpse can be used to extract and add certificates to this PSE file which is particularly
useful if your security expert has no access to ABAP-based administrative tools. Both tools will be described later
in this document.
4 Other Network Technology Aspects

4.1 Network Addressing

In this section, we'd like to briefly mention network addressing mechanisms, which need to be known in order to prepare technical connectivity between cloud and customer on-premise applications.

Networks connect servers on which application processes are running. Therefore, network addressing consists of an IP address, which identifies a server, and a port, which is an entry point to a particular process running on a server.

The TCP/IP network protocol also uses the “source IP address” of the communication client. Many customers have network security components installed which apply security filters based on source IP addresses of clients. So it is important for application and network teams to exchange such addressing information.

For opening, an IP address and port to the Internet gateways, which translate internal IP addresses into public Internet IP addresses, might be in use as well. Therefore, it is important to plan properly which reverse proxy outbound IP address has to undergo a “Network Address Translation” (“NAT”ing) to a public IP address. Depending on particular customer network topologies, even more IP-address relevant components might be in the communication link between an SAP back-end and Cloud application and would need to be planned for. A typical list of needed IP addresses and ports is given in the table below.

<table>
<thead>
<tr>
<th>Component</th>
<th>Address</th>
<th>Example Value in picture above</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP Back-end System</td>
<td>Target IP address and port</td>
<td>10.0.140:8443</td>
</tr>
<tr>
<td>Reverse Proxy</td>
<td>Server-side source IP address</td>
<td>10.0.150</td>
</tr>
<tr>
<td></td>
<td>Client-side target IP address</td>
<td>10.0.2.60:443</td>
</tr>
<tr>
<td></td>
<td>NATed public IP address and port</td>
<td>xxx.111:443</td>
</tr>
<tr>
<td>SAP cloud application</td>
<td>public source IP address</td>
<td>yyy.0.140</td>
</tr>
</tbody>
</table>

IP addresses are commonly mapped to server host name and public Internet domain names through Dynamic Name Services (DNS) or other network services. Through such mechanisms, you might send browser requests of the kind https://mycompany.com instead of typing https://xxx.111443. It is important to track such IP addresses to name mappings as well.

A good overview of IP addresses and their server, host, and Internet domain names is needed for application and network teams to configure the traffic routing and application communication configurations. For example, an application might respond with reference URLs to further services it is providing. In such cases, the application cannot just provide references built with its local server hostname and port. Rather, all proxy and NATing address and port transformations need to be considered so that an external client receives a reference that is accessible via the Internet.

Example:

An internal ABAP-based application server might have the IP address 10.0.140 and exposes a
"myservice" Web-service APIs on port 8000, which is set up as an http port. The IP address might be mapped via a DNS service to the hostname "myhost". So, you can call this resource from internal datacenter networks as http://myhost:8000/myservice. However, such a URL is not known on the Internet and it is not secured.

Therefore, you might set up a reverse proxy, which adds encryption and with that an http → https protocol switch and an 8000 → 443 port switch. The reverse proxy will expose your web service on the IP address:port 10.0.2.60:443. A NATing-capable Internet gateway component translates this address to an Internet known address your company owns, let’s say xxx.11.1.443. This Internet address might be mapped by your company to the Internet domain mycompany.com. This way, your web service becomes available to external clients on the Internet under the secure URL https://mycompany.com/myservice (443 is the https default port and therefore can be omitted in the URL).

This information is also important when you create certificates for securing and authenticating connections.

For example, server certificates need a Common Name (CN) parameter that defines the host name or Internet domain. When a service is exposed to the Internet through proxies and NATing Internet Gateways, you need to make sure that you use the Internet common name instead of the internal application server host name.
4.2 Advanced Network Technologies Sometimes In Use

In section 2, we discussed that the reverse proxy function is just one out of many network application delivery services provided by application delivery controllers or ADCs. Therefore, if any of the ADC services is used for protecting a customer's datacenter, this paper should be helpful because ADCs always act as a reverse proxy in such cases too.

However, there are other network services, typically not included in ADCs. In this section, we want to briefly mention them and their co-deployment with a reverse proxy in case your company is using any of these services.

Below the network security layer, of which an ADC and reverse proxy are part, is the pure network connectivity layer. It consists of physical network cabling across the world and network switching and routing hubs operated by network carrier and telecommunication companies. The Internet is built on top of such network infrastructure. The Internet is the easiest way to interconnect Cloud services and customers on-premise datacenter applications. An alternative to the use of the public Internet is to use private network lines, which can be leased from carriers. Such private lines are often used as backbones for a company's "intranet", a company's internal cross-location network. Sometimes, such private networks are referred to as MPLS networks (see [http://en.wikipedia.org/wiki/Multiprotocol_Label_Switching](http://en.wikipedia.org/wiki/Multiprotocol_Label_Switching)) according to the low level network protocol used to provide connectivity across locations.

Intranets typically offer guaranteed SLAs and privacy of network traffic, since no public access to such networks is given. Intranets, of course, add leasing costs of the private global network lines. Since an intranet locks out a lot of common security threats, omitting firewalls and reverse proxy as access control point in front of application servers might be permitted by a customer's security policies if a customer trusts the security of their intranet as a whole.

In the case of customers who extend that trust in their intranet integrity to their Cloud services provider, MPLS connectivity without the use of a reverse proxy between their datacenter applications and the Cloud datacenter might be an option, if that is an option offered by a Cloud Service too. However, it would be safer and more advisable to use MPLS connections in addition to a reverse proxy and not as an alternative to one. MPLS connections would complement the reverse proxy security and add network reliability and higher network performance. The reverse proxy would prevent bridging of a company's private intranet with the tenant network of the Cloud service provider.

When using the Internet for site-to-site connectivity, customers need to open a network port to the Internet through which a Cloud service can send request. While a reverse proxy can filter requests, for example through authenticating the client, it cannot prevent malicious requests being sent to it through the open Internet port. While such malicious requests can't usually penetrate network defenses of firewalls and reverse proxies, a massive attack, referred to as Denial of Service (DoS) attack, can overload them. For instance, coordinated
"botnet" attacks might be staged through Trojan infested PCs and can render open Internet ports unresponsive to legitimate requests.

A remedy against such attacks is the above mentioned deployment of private network lines or the use of a Virtual Private Network (VPN) solution. With a VPN connection, the customer is calling out of the on-premise datacenter to the Cloud service for establishing a network connection. That way, the customer does not need to open an entry port to the Internet and no DoS attacks can be staged. The VPN connection is kept open indefinitely so that the Cloud Service can send requests to on-premise systems via the "VPN tunnel". Like MPLS private lines, VPN tunnels bridge the Cloud tenant network and the on-premise datacenter network. Therefore, VPN solutions should be used in addition to a reverse proxy, which would isolate the customer's and Cloud provider's network and act as an access control point to a customer's applications.

Content Delivery Networks or CDNs are another way to enhance security and performance.

A CDN is a cloud network service that works through a large numbers of CDN entry hubs around the world. Your datacenter, as well as any application client, connects to their nearest CDN hub, which might be just a few miles away. Traffic between the CDN hubs is routed through the CDN provider's private network. CDN providers might...
offer you a service to filter malicious attacks at their many end point hubs so that the attack traffic never reaches your datacenter. CDN's also enhance network performance compared to using just the Internet.

The last services we should mention are WAN Optimization Controllers or short WOCs. WOCs are primarily designed to enhance the long distance network performance. They transform standard network protocols and common content formats like HTML and XML into more optimized and highly compressed protocols and formats through which network traffic can be accelerated greatly. In order to function with network standard based application and client end-points, WOCs need to be deployed at all communication end points. Through such symmetric deployments, they can optimize network traffic transparently for the end points.

The deployment of WOCs is very similar to that of VPNs. WOCs are like VPNs with the additional benefit of network performance enhancements. As such, WOCs bridge datacenter and Client site local networks. As mentioned before, a reverse proxy serves as network isolation and access control point. Originally, this network technology has been applied to application end-user traffic. They are also applicable to application integration traffic but maybe not as commonly used. In the case of Cloud to on-premise application integration traffic, the different ownership of the end points with the cloud provider and their customers respectively is an additional obstacle. The same proprietary WOC network services need to be available from both ends to be connected and there has to be agreement which side pays for this advanced network services.

**Note:** different types of network services such as reverse proxies, VPN, WOC and others are not commutable in a mathematical sense.

The order in which they are applied to network traffic cannot be exchanged arbitrarily. One example would be VPN and reverse proxy services. A VPN tunnel needs to be terminated before traffic hits a reverse proxy. This is because a reverse proxy would need to terminate SSL sessions and it can't do that on traffic that is still VPN encrypted. Consult with your network experts on the concrete network services topology in your network. You should also check that network services used in addition to one reverse proxies or the use of multiple reverse proxies in a sequence do not interfere with the x.509 certificate based trust relation handling of clients to back-end systems.

In summary, it can be said that reverse proxies are the foundation of any Cloud to on-premise datacenter application integration scenarios. They provide traffic security and authentication as described in this paper. Other complementary network services may be applied in addition to the use of a reverse proxy to further enhance security and performance of cross location networks.

Note that not all cloud solutions of SAP offer the same technical connectivity options since they are very different in nature and cost aspects are also different. For most SaaS cloud solutions, the use of reverse proxies is typical and this is the focus in this paper. Since SaaS end-user response times are sometimes critical, in particular when individual users might visit an application only very infrequently like for travel booking, CDN services might be added to them.

SAP's Hana Cloud Platform (HCP) offering is a platform as a service (PaaS) offering for which the use of reverse proxies on the customer datacenter side is possible too, but the use of the SAP Hana Cloud Connector (HCC) is the recommended way. The HCC is a combination of an SSL-VPN service plus a reverse proxy as described in the second case above.

The SAP Hana Enterprise Cloud (HEC) is a managed services plus infrastructure as a service (IaaS) offering by SAP. Since IaaS services are often used as an extension of a customer's on-premise datacenter, more connectivity options might be applicable, such as MPLS and iPsec-VPN.

In any case, please check the detailed documentation of your cloud solution from SAP for the best connectivity options.
5 Checklist for Project Managers

The previous sections gave you an overview of the various aspects for setting up technical connectivity between customer on-premise back-end applications and SAP Cloud services. This chapter summarizes the different tasks and project member roles for implementing such technical connectivity for productive use. The following information is intended only as a starting point and advice to customers. No warranty and liability nor completeness for the situation at your company is assumed by SAP. You should customize this checklist according to your company's needs.

The biggest risk for the execution of any implementation project is the unknown. If you discover what to do, who to involve, which approvals to get and so on as you go, your work is sequential. You will likely encounter delays and cost increases. On the other hand, knowing all details at the beginning allows different contributors to an implementation project to work in parallel and a lot of time and costs can be saved.

5.1 Implementation Team

A technical connectivity implementation project needs a cross organizational virtual team consisting out of application, network and security experts. The roles in detail are:

<table>
<thead>
<tr>
<th>Organizational Group</th>
<th>Role</th>
<th>Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application team</td>
<td>Application Expert</td>
<td>Application integration, which APIs to use for a hybrid scenario</td>
</tr>
<tr>
<td></td>
<td>NetWeaver Platform Expert</td>
<td>Security configurations, certificate set-ups in the NetWeaver platform. Optional, if SAP WebDispatcher is used as reverse proxy, WebDispatcher configuration skills are needed too</td>
</tr>
<tr>
<td>Network team</td>
<td>Reverse Proxy Expert</td>
<td>If a commercial reverse proxy/ application delivery controller product is to be used, a configuration expert for your RP/ ADC solution is needed</td>
</tr>
<tr>
<td></td>
<td>Internet Connectivity Expert</td>
<td>Provide Internet address/ domain name, configure Firewalls and Internet Gateways</td>
</tr>
<tr>
<td>Security team</td>
<td>Certificate Expert</td>
<td>Provide security certificates for application and network components</td>
</tr>
<tr>
<td></td>
<td>Governance</td>
<td>Ensure that all corporate security guidelines are followed for the technical connectivity implementation</td>
</tr>
<tr>
<td>Project</td>
<td>Project Manager</td>
<td>Overall Project Management</td>
</tr>
<tr>
<td></td>
<td>SAP Cloud Expert</td>
<td>Provide all application and security connection information for the SAP Cloud as client of customers back-end applications</td>
</tr>
</tbody>
</table>
### 5.2 Breakdown of Implementation Tasks

You need to plan the following task items for a hybrid application deployment scenario. You should review this list and customize it for your situation. Depending on other integration scenarios that might have been implemented against your SAP back-end system in the past, some of the steps listed might have been implemented before and don't need to be done again. Therefore, review the list below with your experts carefully.

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Task</th>
<th>Detail</th>
<th>Depends on Nr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get the SAP Cloud Client x.509 certificate, and corresponding root certificate</td>
<td>These certificates have to be delivered by the SAP Cloud Solution and imported into the reverse proxy and SAP back-end system</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Generate Server certificate(s) for your SAP back-end System and reverse proxies with the Certificate Authority your company prefers</td>
<td>This is typically handled by your security group. You might use the same or different server certificates in the back-end system or reverse proxy. Ensure that the Certificate Authority used is supported by the SAP Cloud Service.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Generate a client certificate for your reverse proxy so that it can be authenticated by the SAP back-end system</td>
<td>Typically, this is handled by your security and network group. This client certificate needs to be imported into your reverse proxy and the SAP back-end system.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ensure that any certificates you use are tracked by your company's certificate life cycle management procedures</td>
<td>Typically, certificates have an expiration date. Once they have expired, clients, reverse proxies and SAP back-end systems can no longer use them for secured communications. Thus, in the worst cases, a forgotten certificate expiration date can cause unexpected application downtimes.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Enable SAP back-end application systems for secure communication.</td>
<td>A basis consultant would need to download SAP Crypto libraries from the SAP Service Market Place and implement them in your back-end systems.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Setup your SAP back-end system as secure https server</td>
<td>Use the NW-ABAPABAP transaction STRUST to create your server certificates</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Implement your reverse proxy as &quot;trusted intermediary&quot; in the SAP back-end system</td>
<td>You need to import your reverse proxy client certificates into the reverse proxy as well as into the SAP back-end system. Further, you need to configure the SAP back-end system so that it recognizes your reverse proxy as trusted intermediary of SAP Cloud clients</td>
<td>3,5,6</td>
</tr>
<tr>
<td>8</td>
<td>Identify the web services in the SAP back-end system, which are needed for the hybrid application integration</td>
<td>For detailed information, take a look at your SAP hybrid application documentation will provide you detailed information</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Set-up a technical user in the SAP back-end system to link the SAP Cloud client authentication to the necessary authorization for executing web services in the SAP back-end system</td>
<td>Create the technical user. Maintain table VUSREXTID to link the user to the SAP Cloud client certificate authentication. Maintain user profile and roles for sufficient permission to execute the integration web services.</td>
<td>1through 9</td>
</tr>
<tr>
<td>10</td>
<td>Perform technical configuration of the integration web Services for security and client authentication</td>
<td>Use transactions SCIF and SOAMANAGER. Some cloud solutions from SAP might also provide wizard-reports to accelerate these configurations, check your cloud solution implementation guides for details.</td>
<td>1through 10</td>
</tr>
<tr>
<td>11</td>
<td>Perform proper end-to-end testing</td>
<td>anchise</td>
<td>1through 11</td>
</tr>
</tbody>
</table>
5.3 Overview of Key Stores and Certificates

As you can see from the task list and throughout this document, deploying and maintaining certificates throughout the various components involved in end-to-end connectivity and security is absolutely essential. So, here’s a list of key stores, keys, and certificates. It might be helpful to keep this list handy as a cheat sheet while you navigate through this guide and implementation work.
<table>
<thead>
<tr>
<th>Component</th>
<th>Key Store</th>
<th>Key or Certificate</th>
<th>#</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>On premise ABAP backend System</td>
<td>Server key store</td>
<td>private/ public key pair</td>
<td>1</td>
<td>automatically created when you create a server PSE and initially self-signed server certificate in transaction STRUST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>server certificate</td>
<td>2</td>
<td>You need to &quot;sign&quot; your initial server certificate with your preferred Certificate Authority</td>
</tr>
<tr>
<td></td>
<td></td>
<td>root certificate of CA used for signing the server certificate</td>
<td>3</td>
<td>best imported together with the signing response from your CA in a &quot;chained&quot; PEM format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>root certificate of CA used for signing the r.p. client certificate</td>
<td>4</td>
<td>Needed to enable the backend system to authenticate the reverse proxy as a trusted intermediary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>root certificate of CA used for signing the SAP Cloud client certificate</td>
<td>5</td>
<td>Needed to enable the backend system to authenticate the SAP Cloud as a trusted client.</td>
</tr>
<tr>
<td>Optional Reverse Proxy client key store</td>
<td>private/ public key pair</td>
<td>6</td>
<td></td>
<td>If you prefer to use the ABAP-STRUST transaction to create and manage your key stores, you could create the client side key store for your reverse proxy in your ABAP backend system and then export the entire key store, including the private key, to your reverse proxy. This option is particular useful if you choose SAP Web Dispatcher as reverse proxy.</td>
</tr>
<tr>
<td></td>
<td>reverse proxy client certificate</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>root certificate of CA used for signing the r.p. client certificate</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>root certificate of CA used for signing the backend system server certificate</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reverse Proxy</td>
<td>client key store</td>
<td>private/ public key pair</td>
<td>6</td>
<td>The reverse proxy is a &quot;client&quot; to the SAP backend system. Therefore a client side key store is needed.</td>
</tr>
<tr>
<td></td>
<td>reverse proxy client certificate</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>root certificate of CA used for signing the r.p. client certificate</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>root certificate of CA used for signing the backend system server certificate</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server key store</td>
<td></td>
<td>private/ public key pair</td>
<td>9(1)</td>
<td>You can create this key store with tools provided by your reverse proxy or you can re-use the same key store as you use for the backend system. Re-using the backend key store is particular time saving if you choose SAP Web Dispatcher as your reverse proxy. Numbers in () in the left column indicate the re-use option.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>server certificate</td>
<td>10(2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>root certificate of CA used for signing the r.p. server certificate</td>
<td>11(3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>root certificate of CA used for signing the SAP Cloud client certificate</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP Cloud Service</td>
<td>client key store</td>
<td>private/ public key pair</td>
<td>12</td>
<td>Don't forget that the SAP Cloud client has to verify the server certificate it receives from the on-premise reverse proxy. For that it needs the root certificate of the CA used for signing the on-premise server certificate.</td>
</tr>
<tr>
<td></td>
<td>SAP Cloud client certificate</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>root certificate of CA used for signing the SAP Cloud client certificate</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>root certificate of CA used for signing the reverse proxy's server certificate</td>
<td>11(3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The enumeration in the # column indicates which certificates need to be placed into multiple key stores.
6 How to Implement

When you integrate your on-premise SAP back-end application with an SAP Cloud application you need to configure security details and network connectivity across your back-end system, your network reverse proxy, and the SAP Cloud solution itself. Since most SAP back-end systems are integrated already with many other systems, you might not need to perform all configurations in that system from scratch. The same holds for the reverse proxy. Your network group probably already has a reverse proxy appliance installed, which can be easily configured to carry another reverse proxy address forwarding from the Internet to your back-end system. However, even if many configurations are in place, you need to review them before using them for any new integration. Therefore, the following subsections describe what needs to be configured so that you either can perform these configurations or verify them.

Note: The following configuration description is given only on a best effort basis. Details might depend on your SAP application release level and other details. Please also check SAP's http://help.sap.com websites for more details. We used an Amazon VPC-based CRM system to test the following steps. Details you'll see in screenshots, for instance host names, IP-addresses and URLs, would need to be adjusted to your landscape's specific parameters.

6.1 Security Implementations in a NetWeaver ABAP-Based Application Back-End System

According to the task list provide in section 5.2, you need to ensure that the following security implementation tasks are performed:

1. Enable SAP back-end application systems for secure communication.
2. Setup your SAP back-end system as secure https server.
3. Implement your reverse proxy as “trusted intermediary” in the SAP back-end system.
4. Set-up a technical user in the SAP back-end system to link the SAP Cloud client authentication to the necessary authorization for executing web services in the SAP back-end system.
5. Perform technical configuration of the integration web Services for security and client authentication.

We will describe these steps in the following subsections

6.11 Enable SAP back-end application systems for secure communication

A good step-by-step description is given at http://help.sap.com/saphelp_nw70/helpdata/en/3a/7cddde33ff05cae10000000a128c20/frameset.htm. Newer SAP NetWeaver kernel versions are delivered with cryptographic software packages (see OSS note 1848999 for release details). Older versions came without it. Therefore, the first step you need to perform is to check in your back-end system whether SAP's cryptographic software packages are included or have already been installed in your back-end system.
If those packages are not installed already, you can download the Cryptolib software from the SAP Service Marketplace "Trust Center Services (TCS)" section http://service.sap.com/tcs. This section is worth bookmarking since you can get common SAP root certificates and a free Certificate Authority test service from this site as well. On the TCS site, go to the "Download Area" and then click the "SAP Cryptographic Library" link. Download the Library version matching your back-end system operating system.

Then follow the instructions in http://help.sap.com/saphelp_nw70/helpdata/en/3a/7cddde33ff05cae10000000a128c20/frameset.htm to deploy the cryptographic library into the right file system directories of your back-end system. This information is also useful if you need to check if the cryptographic software is already installed.

Finally, you have to set a number of profile parameters to make the Cryptolib location and file systems locations known to your ABAP servers. Set the Trust Manager parameters as described in http://help.sap.com/saphelp_nw70/helpdata/en/3a/7cddde33ff05cae10000000a128c20/frameset.htm.

6.12 Set up your SAP back-end system as secure https server

NetWeaver systems have an in-built web-server component called Internet Connection Manager or short ICM (http://help.sap.com/saphelp_nw70/helpdata/en/56/2e453cabf4ef6fe10000000a114084/frameset.htm). As a first step, we'd recommend that you configure an http as well as an https port for your back-end system for the purpose of testing during the implementation phase. Depending on your company’s security policies, you might use the https port later in a productive setting and then disable the other http port if necessary.

A sample ABAP profile parameter setting would be:

```
icm/server_port_0 = PROT=HTTP,PORT=8000
icm/server_port_1 = PROT=HTTPS,PORT=44300
```

Note that the example does not use the standard ports 80 for http and 443 for https, which would also be possible. However, for better security, we'd recommend you not to use Internet standard ports in internal networks and for back-end systems.

With a browser connected to your internal application network you might want to perform a simple "ping" test by calling

```
http://<your host.domain:port>/sap/bc/ping
```

This could be as an example:

```
http://crm371.dummy.nodomain:8000/sap/bc/ping
```
For the "sap/bc/ ping" service to work, you'd need to ensure that it is activated through the ABAP transaction SICF.

The next step is to enable your ABAP system to support secure https communication. The central transaction for such configuration is STRUST, see http://help.sap.com/saphelp_nw70ehp2/helpdata/en/0e/fb993af7700577e10000000a11402f/content.htm
In STRUST, you maintain key stores, called Personal Security Environments or PSEs. In order to enable the ABAP system to act as a server to https requests, you need to create or maintain the "SSL server Standard" PSE.
You can start by right clicking the SSL server Standard line to start the PSE creation process. When you perform the PSE creation process, STRUST creates a public/private key pair and a self-signed server certificate for you.

Here you need to coordinate 2 important details with the network and security group of your company:

1. The domain name you will use later on the public Internet to connect to your back-end system via reverse proxies. If you don’t use the right domain name in this certificate, end users might encounter security warnings later when they are sending requests to your system.

2. The Certificate Authority you will use for signing your certificates, including your ABAP system’s root certificate.

We tested procedures described in this paper in a test landscape on the Amazon Web Services (AWS) cloud. Therefore, our Internet domain is ec2-XXX-XXX-XXX-XXX.compute-1.amazonaws.com. We also used the OpenSSL based tool XCA (http://xca.sourceforge.net/) for being our own Certificate Authority. We created a CA root certificate for our own test CA called “SAP ICC CA”.

You should use Org. and Comp. filed entries that are meaningful to your organization. We recommend using the largest available key length in the last field, which is RSA 2048.
Once this is confirmed, a new popup occurs for creating instance specific PSEs. You might have multiple instances per ABAP system deployed. In our example we have only one instance. Ensure that the "CN" and all following values of the Standard PSE and the Instance Specific PSE's are equal. You might have to edit the instance PSEs, which might have a different default CN value. Once ready, confirm your entries.

After confirming these entries, you will find a new entry in the "own certificate" box. Double click the "owner" line to see the details of this certificate in the lower "Certificate" box.
At this point, the certificate has been self-signed in the ABAP system itself. Therefore, Owner and Issuer values are equal. If your system is already connected to the Internet, you can perform an https ping test with your public URL. For our example, system the URL would be:

https://ec2-54-208-16-57.compute-1.amazonaws.com:44300/sap/bc/ping

As response, you see a security warning.
This warning occurs because your browser correctly does not recognize the “issuer” Certificate Authority of your new self-signed certificate. This way the SAP ABAP system is not yet trusted by the browser client. However, if you click the continue option on the warning page, the original ping request is continued and you receive the proper response over an encrypted https connection.

To establish trust to clients of your ABAP server, you need to "sign" your server certificate with a Certificate Authority that is trusted by your client.

**Note:** You need to coordinate with your network and security group which Certificate Authority your company prefers and then cross check with the SAP Cloud solution documentation which Certificate Authorities are supported by them. The CA used for signing your server certificates needs to be on the trusted CA list of the SAP Cloud solution or you need to be able to import your CA's root certificate into the key store of your cloud solution from SAP.

For test purposes, we imported our “SAP ICC CA” Certificate Authority root certificated into the "Trusted Certificate Authorities" key store of our test browser as shown above. That way, the browser as a client will trust server certificates signed by the SAP ICC CA.

As the next step, we need to move from a self-signed to a signed server certificate in the ABAP system. For that, you need to create a certificate signing request first, by clicking the create certificate request item as marked below.
You can cut-and-paste the request section or download into a file. Use this request for generating a certificate response with your Certificate Authority. Any CA will have a certificate signing procedure to which you submit your certificate request and get back a certificate response in return. For testing purposes, you could get a signed test certificate from the SAP Trust Center Site of the SAP Service Marketplace or you could use the XCA tool. The response should be best given in a "PEM format with certificate chain". That means that not only the signed version of your server certificate will be given, but the CA’s root certificate will be included as well (see also SAP note 1468249). The CA’s root certificate needs to be imported into your ABAP system as well in order to establish trust between your ABAP system and a CA. The import of the signed server certificate response then looks as below.
Note that your certificate response contains two certificate sections for your signed server and the CA’s root certificate.

After confirming the response import, do NOT forget to click on the ABAP save icon to save your changes. You can double click the “Owner” line again to see your new signed server certificate. The issuer line changed to your CA’s name, in our example it is “SAP ICC CA”.

If you now repeat the https ping test with a browser that knows your Certificate Authority root certificate, you should get a response without security warning.

This concludes the setup of the ABAP back-end system as secure https server for SAP Cloud integration scenarios.

6.1.3 Import Client Certificates into your ABAP system for Client authentication and link them to a user for authorization purposes

The earlier sections described how an ABAP system can be configured as server for encrypted (https) network requests. However, as pointed out before, trust relationships between server and clients and intermediaries like a reverse proxy need to be deployed as well. In short, all traffic needs to be both encrypted and authenticated. Both server and clients should be authenticated, so both client certificates and service certificates need to be used. The back-end system authenticates clients such that the client can be mapped to a user in the back-end system which carries the authorization via profiles and roles to access data and to serve web service requests.

Cloud to on-premise application integration is facilitated through web service requests. Therefore, one task for an integration project is to identify the back-end system web services needed for a particular integration scenario. Then the web service security configuration described below has to be performed for those web services.
In order to test the on-premise datacenter back-end, reverse proxy and DMZ firewall end-to-end configuration for web services, it is most convenient to create a test service in the back-end system and use a web-service client emulation tool to call the test web service. That way, you don't need to trigger test web service calls from the SAP Cloud and you avoid creating artificial business data in your back-end system. A good web-service client test tool with free trial versions is SoapUI (http://www.soapui.org/). We used version 4.01 for our testing.

To start the ABAP site configuration, you need to get an x.509 SAP Cloud client certificate assigned to your cloud application tenant of the cloud solution, which will be integrated with your on-premise SAP application. You need to get this client certificate and the root certificate of the Certificate Authority for it from your SAP cloud solution administration as PEM formatted files. Then import the client and root certificates via the STRUST transaction into your SSL server Standard PSE. If this client certificate is signed by a Certificate Authority not used yet in your PSE, you need to import your CA root certificate as well.

Before you import those certificates, you should maintain the certificate database in the STRUST transaction.
Clicking this menu point takes you to the VSTRUSTCERT table maintenance screen.

In our example above, you can see that we maintained 2 new entries, Z_ICC and Z_ICC_CL. Z_ICC is the entry for our sample server certificate's Certificate Authority and Z_ICC_CL stands for our sample Certificate Authority used for SAP Cloud Client certificates.

When you import certificates into your key stores using the STRUST transaction, you should specify the corresponding certificate database as part of the certificate import step.
For test and documentation purposes we used the XCA tool again to create our own "SAP ICC Cloud" CA root certificate and then a client certificate signed by that CA.

When a client request enters the back-end system the validity of the client certificate is checked in a first step. For this check, the back-end system needs the root certificate of the Certificate Authority that signed the client certificate. In our example, this would be the SAP ICC Cloud CA root certificate.

You can import this root certificate using the import button in the certificate box as shown above. To conclude the import, click "Add to Certificate List", then click the ABAP "save" icon to add the client certificate to your server PSE.

Note the Owner CN=SAP ICC Cloud CA is just used for demonstration purposes. You would get the root certificate from your SAP Cloud solution and it would have a different common name (CN).

Once the client certificate is validated against its CA root certificate, the ABAP system needs to authenticate your SAP Cloud Client requests in a second step. In order to link the authentication to authorization, you have to create a technical ABAP user with proper roles and permissions to execute the web-services needed for your integration scenario. Coordinate the technical user setup with your ABAP-basis and -application experts.

You need to link the client certificate authentication to your technical user authorization through maintaining an entry in the view VUSREXTID using transaction SM30. Choose work area "DN" as shown in the next screenshot.
If the view is not empty, click "new entry" to get to the following screen.

Use the Import icon on the external ID to import your client certificate, similar to the import you did in the STRUST transaction before. The certificate import file needs to have the certificate "PEM" format with file extension .cer (which is provided by the XCA tool as well). If you got the certificate PEM file with a different extension, just rename the file to have a .cer extension for this import step.
After the import, add values to the Seq.No, user and Min. Date filed. The sequence number is typically 000 unless you have multiple entries for the same user. The user has to be your technical user with proper authorization for executing the integration web services (sapuser in our example). It’s probably best to take today’s date as the minimum (starting) date. Flag the “Activated” field too. Then save those entries.

6.14 Web Service Security Configuration

The next step is configuring security settings of individual web services needed for an integration scenario. We do this best in multiple steps again so that we can perform and simultaneously test and verify individual configuration steps one after another.

The configuration transaction in your ABAP back-end system to use to use is SOAMANAGER. Your application experts should have set up web service HTTP endpoints here. For the purposes of this document, we created a test service called ZHCI_WEBSERVICE_V2. You have to perform steps described in the following for your integration web-services instead.

If you call the SOAMANAGER transaction out of the SAPGUI frontend tool, a web browser window is opened. You might have to work on the same local network as the application server is running in for this to work. This depends on how your ABAP system is configured to generate URLs. By default URLs are generated with the configured ABAP host name, for which you might have DNS resolution only available in your local network.

SOAMANAGER prompts you to log in again and after login you navigate to “Single Service Configuration”. Enter a suitable “Search Pattern” (Z* in our example) and click “Go” to get your web services listed. Mark the line with the service you’d like to configure and click “apply section”. Click the line with the endpoint you might want to configure (there might be multiple) and continue with Display for looking at the configuration or Edit for changing it. Click the security tab. You should arrive at a screen like the one shown below.
At first the details of the security configuration can be set to use http and simple user/password checks.
With these settings you can do a first test calling this web service. Assuming that we already linked our ABAP system to an Internet URL, as was described in the previous sections, our first test URL is:

```
http://ec2-54-208-16-57.compute-1.amazonaws.com:8000/sap/bc/srt/rfc/sap/zhci_webservice_v2/167/166baad04c741ee391d9c5b4b30012ce/binding
```

The URL consists of the Internet domain:port followed by the web service path (defined in transaction SICF by SAP or your custom application development), the SAP Client (Mandt) number where your application data resides (here 167) and the http endpoint path. Your integration web services would have a different URL with a similar structure.

As a web service Client, we use the tool SoapUI for demonstration purposes.

As shown in the picture above, SoapUI allows you to see the web service request and response data and it allows you to configure authentication schemes for the web service calls.

In our first test, the technical user name is is sapuser and we use its password credentials to authenticate our request in the ABAP back-end system.

In the next step, we want to switch from http to secure https requests. For that, we simply switch the "Transport Protocol" configuration for our web service in the SAOMANAGER from http to https and save our change in order to take effect.

If we repeat our http test, it fails with an authorization error "403 Forbidden", as was to be expected.
In our soap UI client we have to correct our URL to use https and our https port 44300 to get a positive response. In the “Raw” view of SoapUI error free responses are indicated by the “Http/1.1 200 ok” line.
The final step is to switch the authentication method from user/password to authentication through X.509 client certificates, which is more secure. No user/password information needs to be sent. To make this change, you simply check X.509 instead of userID/password authentication in the SOAMANAGER transaction.

Depending on how you created your web service endpoint in SOAMANAGER, you might see different endpoint configuration layouts. Earlier, we showed an endpoint configuration which has a security tab. Others have a “Provider Security” tab instead. In any event, you can configure the X.509 authentication in either of the layouts.

6.1.5 Web Service Authentication Testing

Before continuing, it is worth testing a web service call with authentication and authorization. You should know how to debug problems if errors occur, so we’ll use this section to demonstrate test tools and tracing capabilities in ABAP.

As mentioned before, an easy tool to use as a web service call client is SoapUI. To add a client X.509 certificate to the SSL configuration, the best advice is to use the Preference setting screen. You need to supply the X.509 certificate in a PKCS#12 format, which typically has the extension .p12 or .pfx. PKCS#12 contains the X.509 certificate itself, the CA root certificate and most importantly the private key the client needs to encrypt traffic. Private keys need to be kept very secure themselves because only they guarantee that no unauthorized third party can decrypt traffic. Therefore, PKCS#12 files are assigned a password when created and can be opened only with that password. The password needs to be added to the SoapUI configuration.
When we executed a call of our test web service the result looked as follows.

The signs of a successful call are the "HTTP/ 1.1 200 OK" response code and the sending of the X.509 client certificate as can be seen in the SSL Info tag. If you scroll down in the SSL info box, you can also find a "Peer Certificate", which is the ABAP server certificate we configured before on the ABAP-side in transaction STRUST.
A popular commercial tool many customers might be using is HP LoadRunner ©. A LoadRunner script for the same call as used in the SoapUI tool could look like the following:
One way to supply the X.509 certificate to LoadRunner is as 2 files: The certificate itself and the private key as well. Both files should be in PEM format. Again, you could use the XCA tool to create those files. If you replay this script with extended logging on you can find hints in the log that the client certificate is indeed used for the client/server communication (see red circle mark).

Another helpful tracing tool is the built-in ABAP engine in its ICM (Internet Connection Manager) component. The ABAP-ICM component is the web-server layer in front of the ABAP work Processes. The ICM is administrated through the transaction SMICM.

Choose Goto ➔ Trace Level ➔ Set to increase the trace level to 3 to trace web traffic in more detail. With Goto ➔ Trace File ➔ Reset, you can truncate the ICM trace file before you do a test. Then, after doing a test Web Service call, you can use Goto ➔ Trace File ➔ Display All to view the trace. Some key passages in the trace file to look out for are the shown in the following figure. You can see that the client certificate was received and that it is indeed the right one we wanted to send.
6.16 Configuring Authentication of a Trusted Reverse Proxy in an ABAP System

A reverse proxy itself is a client towards an SAP back-end system. Therefore, a trust relationship between the reverse proxy and the ABAP System itself can be configured for maximum security in large customer landscapes with potentially many different application servers, many reverse proxies and a vast company global intranet. In such situations, tying up the proxy to back-end connectivity itself is an additional security measure.

As with any client, the reverse proxy can authenticate itself towards an SAP ABAP back-end system with its own X.509 certificate. You need to create such a certificate yourself with your preferred CA, or use a tool like XCA. Being your own CA might be acceptable for such internal connections within your company network if you can administrate and secure your certificates and their private keys in an appropriate manner. Please consult your security experts for further details.

As usual, the CA’s root certificate used for signing the proxy’s X.509 certificate needs to be imported in the ABAP server’s key store with transaction STRUST. Just as an example, we created a Certificate Authority with Common Name “SAP ICC Proxy CA”. Then we created a client certificate with Common Name “Your Proxy” and signed it with our Certificate Authority “SAP ICC Proxy CA”. We will explain later how to use this client certificate for a reverse proxy’s client-side key store.

We exported the CA’s root certificate into a PEM formatted file from our XCA tool and imported them with the STRUST transaction into our ABAP System. After Import, our SSL Server PSE/key store looks like shown in the figure below.
In addition to importing the reverse proxies' client certificates, you need to add more system settings in the ABAP profile to enable the reverse proxy authentication through the proxy's client certificate. You can maintain the ABAP profile through transaction RZ10. The parameters to add are:

- `icm/HTTPS/trust_client_with_subject`
- `icm/HTTPS/trust_client_with_issuer`

For our example proxy certificates, typical parameter values would be:

- `icm/HTTPS/trust_client_with_subject = CN=Your Proxy,*`
- `icm/HTTPS/trust_client_with_issuer = CN=SAP ICC Proxy CA,*`

The "*" is just a wildcard allowing any content for subject and issuer after the CN (Common Name) value.

As we explain in section 6.2 it might be useful for testing and connectivity problem debugging purposes to configure a plain http connection between the reverse proxy and the backend system.

**Note:** A plain http connection between a reverse proxy and a backend system is NOT recommended for productive use for security reasons!

If you'd use a plain http connection temporarily for testing the icm/HTTPS/* parameters would not apply. However, it is still possible to forward the original SAP Cloud client certificate via http headers to the backend system for authentication. While you can't define a trust relationship with an http connection between the
backend system and the reverse proxy you can still configure the backend system to recognize the original cloud certificate through the following profile parameter setting:

```
icm/accept_forwarded_cert_via_http = TRUE```

Please do not forget to disable this parameter or to set it to FALSE when you switch to https connectivity between the reverse proxy and the backend system.

You'd need to restart your ABAP system in order to have these parameter settings take effect.

6.1.7 Certificate Life Cycle Management

As described in the 6.1 sections, a number of certificates for different purposes have to be implemented in NetWeaver ABAP-based systems and, as is described next, in your reverse proxy. Even on the SAP Cloud solution side, you have client certificates to maintain.

Certificates have an expiration date. For security reasons, this date should not be set in the distant future. Consequently, certificates need to be renewed every so often, which leads to the need to establish certificate life-cycle management procedures in your company.

If a certificate is not tied into a regular maintenance process, the consequences for the productive use of an application can be severe. It might be that your cloud-on-premise integration scenario is running fine for a year or some long time until the date a crucial certificate for your technical cloud connectivity expires. At that point, access to your application scenario is revoked for all users, which effectively causes a productive use (unplanned) downtime for the scenario. Worse: typical high availability and disaster recovery precautions your IT organizations might have made can't protect you against the consequences of expired certificates.

Therefore, we strongly recommend that you establish active certificate life-cycle management for all application and network components involved in your productively used application scenarios.

For SAP NetWeaver ABAP-based back-end systems, the report SSF_ALERT_CERTEXPIRE is provided to check on all implemented certificates. Please refer to OSS note 572035  “Warning about expired security certificates” for further details.

6.2 Implementing a Secure Reverse Proxy

As described before, a reverse proxy in the path between a customer's back-end application system and the SAP Cloud solutions is a major security precaution to protect a customer's back-end system and the confidential business data kept in such systems. Many customers have already built a “de-militarized network security zone (DMZ)” between their inner company and application system network and the Internet. The DMZ is the right place to put a reverse proxy. Many customers have deployed Application Delivery Controllers (ADCs) in their DMZ. Such ADCs offer reverse proxy services among many other load balancing, security and sometimes network performance enhancing services. ADCs are often specialized hardware appliances with capacity for potentially hundreds of reverse proxies as virtual end-points of application systems towards the Internet. Therefore, it is often possible to just configure one more virtual proxy end-point in such an ADC for your SAP on-premise back-end application to SAP Cloud integration scenario.

Another option is to use the SAP Web Dispatcher component as reverse proxy. The SAP Web Dispatcher (WD) is a component of the SAP NetWeaver platform and therefore available to all SAP Business Suite component customers without additional license fees. The WD is a software component and as such has to be deployed on a physical or virtual server on top of an operating system in your DMZ.
Both options, extending use of existing third party reverse proxy solutions as well as deploying WD, are supported by SAP. For third party solutions, we recommend that you use products that have been certified by SAP and offer the necessary features for client certificate forwarding for authentication. Since Cloud to on-premise application integration is a relatively new feature, necessary client certificate handling tests were not available during older certification tests. Please ask your ADC vendor for further details of features they are supporting, in particular about the X509 certificate forwarding.

The SAP Web Dispatcher supports all needed features, including x509 certificate forwarding.

6.2.1 Configuring Third Party Reverse Proxies

Third party ADCs as reverse proxies are supported by their respective vendors, so we can't give any detailed advice about how to configure them. In this section, we can give just a list of configuration steps that need to be performed. The next section will be about how to configure SAP WebDispatcher after its deployment and might serve as a template on how to configure a third party reverse proxy in an analogous way.

Network security setups follow common standards that are inherently complex. You might try to set up everything in one go if you feel experienced enough. However, if something does not work, troubleshooting can be time consuming. A step-by-step approach, where you gradually add security features, could be a better choice. This section describes that step-by-step approach. Note that, during the steps, your back-end system might be exposed to the Internet without sufficient security for a short time. So it might be a good idea to prepare the configuration of a reverse proxy for a productively used system in a test environment.

The steps to perform for the reverse proxy setup are:

1. Assign the external public Internet address you will use for remote access to your virtual reverse proxy endpoint of reverse proxy instance. This often means you need to set up network address translation (NATing) in your Internet facing router to translate an Internet address into the internal IP address of your reverse proxy in the DMZ.
2. Set up a simple http (no encryption) reverse proxy connection from the Internet to the back-end system in order to verify proper network routing configuration across the different Internet, DMZ, and application network security zones. Note that you have to use different ports for http and https requests, for which you have to configure firewalls differently as you go through the steps. Use the ABAP ping test described in section 6.1 for verifying proper network routing configuration.
3. Switch your reverse proxy to only accept https requests from the Internet side and tighten your Internet facing firewall to only allow requests on the address and https port for your reverse proxy to pass. Your reverse proxy will act as a server to requests coming from the SAP Cloud. Therefore, you need to set up signed server certificates in your reverse proxy in way similar to that described in section 6.1 for the ABAP server. The connection between reverse proxy and you ABAP back-end system remains an insecure http connection. Use the ABAP HTTPS ping test to test a secure connection to your reverse proxy with proper http forwarding to the ABAP system.
4. Configure the reverse proxy to expect the SAP Cloud's X.509 client certificate. Requests without X.509 client certificates should be rejected by the reverse proxy and not forwarded to the SAP back-end system. Further, you should import the CA root certificate of the CA which signed the client certificate into your reverse proxy. The reverse proxy should then validate incoming client certificates with the CA root certificate and reject all client certificates not being signed by the right CA. For this step, you need to get the SAP Cloud client's CA root certificate, which might be specific to your Cloud tenant, from the SAP Cloud solution. The CA root certificate needs to be imported in the reverse proxy's server-side key store.
5. Configure your reverse proxy to forward the SAP Cloud X.509 client certificate in the http header of the request it forwards to the SAP back-end system. Cross check with section 6.1.3 configurations that the right CA root certificate used for signing the client certificate is used on reverse proxy and ABAP-side together.
6. You should secure the network traffic between the reverse proxy and the back-end system to prevent threats from inside your company's network as well. In addition, you can configure the reverse proxy to be authenticated by the ABAP back-end system as a "trusted intermediary" through its own X.509 client certificate. For this step, you need to configure the reverse proxy to use its own X.509 certificate for the
https connection to the back-end system. Details of the ABAP-side configuration steps are described in the 6.1.6 section as well.

6.2.2 The Easy Way to Configure SAP WebDispatcher as a Reverse Proxy

Section 6.1 already described how you can integrate any reverse proxy product, which offers the necessary security features, into the trust relationships and network path between SAP Cloud and ABAP-based on-premise systems. A lot of that work and the methodologies we learned can be used for a simplified deployment of the SAP Web Dispatcher (WD) as a reverse proxy.

Start with downloading the WD software package from the SAP Service Marketplace. Search for Web Dispatcher in http://service.sap.com/swdc to locate the right package for your operating system. The SAP Web Dispatcher is always backward compatible and can be upgraded independently from back-end system upgrades (see OSS note 908097). For new installations, we recommend that you always use the newest WD version (at time of writing, WD 7.41 is the latest one).

Find and download the SAPCAR unpacking tool in the same way. Unpack the software package into some directory (use command sapcar -xvf <WD software package>). Details of the WD installation are described in https://service.sap.com/~form/sapnet?_SCENARIO=01100035870000000202&_SHORTKEY=00200797470000089947&_OBJECT=011000358700000121752008E

Create an initial configuration for WD using the bootstrap command as shown below.

The bootstrap processing creates the sapwebdisp.pfi configuration profile of the SAP Web Dispatcher and an administrative user to a web administration interface of the Web Dispatcher. You might want to immediately note down the administrative user name and password you entered for later use (see red mark in the picture above). If

At the end of the bootstrap processing, the Web Dispatcher is running and should be connected with simple http to your ABAP system. You can perform the ping test performed earlier:

http://<your host.domain:port>/sap/bc/ping

to verify proper Web Dispatcher set up so far.

Note that you don't have to configure a list of all ABAP nodes ("central + dialog instances") in Web Dispatcher, but rather just specify the central message server connectivity parameters. SAP WebDispatcher then automatically queries the ABAP Message server about the other nodes and auto-configures itself based on the message server response. That way, any needed load balancing configuration happens as well. If you are interested in the details of the ABAP Message Server API, please see http://help.sap.com/saphelp_nw73ehp1/helpdata/en/67/f2a88355de4690a429a25ef83c89b2/content.htm

Similar to the ABAP server configurations, you have to deploy the SAP security software libraries as a separate step. See the Web Dispatcher installation document for more details on this step.

The next big task is to supply your Web Dispatcher deployment with server and client-side key stores and certificates and private keys for them. The easiest way to do so is to re-use the key stores we already configured in your ABAP system as described in the previous sections. SAP Web Dispatcher uses the same SAP .PSE format for key stores as ABAP systems do. You can export key stores from ABAP and re-use them for the SAP Web Dispatcher.

NOTE: ABAP .PSE key store files contain private keys and should be kept very confidential and secured in your organization. Since the SAP ABAP and Web Dispatcher instances both belong to your organization, it is OK to share the private keys between those systems with the advantages of easier setup and maintenance. However, if sharing private keys between systems is not permitted according to your company’s internal security guidelines, you need to create separate .PSE key stores for SAP Web Dispatcher with their own set of private keys and certificates. We describe the creation of separate .PSE key stores in the next section.

For Web Dispatcher acting as a server towards the SAP Cloud solution, you can simply use the same SSL server PSE/ key store as the ABAP server. That way Web Dispatcher gets a server certificate with its private key and all client and Certificate Authority root certificates at once and you don’t need to re-assemble such server key store for SAP Web Dispatcher again.

You still need also a Web Dispatcher client-side key store with the Web Dispatcher’s own client certificate and its private key. You can use the ABAP transaction STRUST to build and later administrate such Client SSL PSE again. For that, choose Environment → SSL Client Identities and confirm a pop-up that the next table maintenance screen is for a shared cross ABAP client table.

NOTE: ABAP .PSE key store files contain private keys and should be kept very confidential and secured in your organization. Since the SAP ABAP and Web Dispatcher instances both belong to your organization, it is OK to share the private keys between those systems with the advantages of easier setup and maintenance. However, if sharing private keys between systems is not permitted according to your company’s internal security guidelines, you need to create separate .PSE key stores for SAP Web Dispatcher with their own set of private keys and certificates. We describe the creation of separate .PSE key stores in the next section.

For Web Dispatcher acting as a server towards the SAP Cloud solution, you can simply use the same SSL server PSE/ key store as the ABAP server. That way Web Dispatcher gets a server certificate with its private key and all client and Certificate Authority root certificates at once and you don’t need to re-assemble such server key store for SAP Web Dispatcher again.

You still need also a Web Dispatcher client-side key store with the Web Dispatcher’s own client certificate and its private key. You can use the ABAP transaction STRUST to build and later administrate such Client SSL PSE again. For that, choose Environment → SSL Client Identities and confirm a pop-up that the next table maintenance screen is for a shared cross ABAP client table.
You can create a new table entry by clicking "New Entries". In our example above, we created the client identity SAP_WD as “Client PSE for SAP WebDispatcher. When you return to the STRUST main screen, you will find a new entry for the Client PSE you just created. It is marked in yellow in the left hand PSE list in the following picture. The description string in the PSE name might be truncated if your description is long.

In the very same way as you created the SSL server PSE as described in section 6.1.2, you have now to right click your new client PSE entry to start creating first a self-signed certificate and private key for it and then to sign that certificate with your preferred Certificate Authority. Remember that your certificate signing response needs to contain the CA root certificate as well being chained in a PEM format.

After signing the "own certificate", you need to import the root certificate of the CA you used for signing the ABAP server's SSL server certificate too. When this PSE will be used as client PSE for the SAP Web Dispatcher, Web Dispatcher needs to know the ABAP server's CA's root certificate so that it can verify the authenticity of the ABAP server certificate. After importing that CA certificate, don't forget to add it to the certificate list and to save this PSE change.

At the end of the SSL client PSE creation and signing process, your client PSE for the SAP Web Dispatcher should look like the picture shown above.

Now you are ready to export the SSL server and client PSEs. Simply select one of the PSE's to export at a time and choose PSE→Export to perform the export to a file. You may use the following naming conventions:
• SAPSSLS.pse for the SSL server PSE
• SAPSSLC.PSE for the SSL client PSE

Turn back to your Web Dispatcher deployment and create a new subdirectory “sec” in your Web Dispatcher’s main directory. Then copy your server and client PSE files into that directory.

In the next step, you need to configure the Web Dispatcher profile for SSL and SAP Cloud certificate forwarding. As before, multiple steps with tests in between will be shown so that mistakes are avoided and, if needed, troubleshooting becomes easier.

Open the sapwebdisp.pfl file in an editor like Window’s notepad and extend the initial bootstrap configuration as shown below.

```plaintext
# Profile generated by sapwebdisp -bootstrap
## Profile Extended for Security
## Trace Level: 1=production, 3=most detail
rdisp/TRACE = 3

# unique instance number
SAWSYSTEM = 01

# add default directory settings
DIR_EXECUTABLE = .

# Accessibility of Message Servers
rdisp/mshost = crm371
ms/http_port = 8100

## SAP Web Dispatcher Parameter
wdisp/auto_refresh = 25
wdisp/max_servers = 100

wdisp/shm_attach_mode = 6

# configuration for small scenario
icm/max_conn      = 100
icm/max_sockets   = 1024
icm/req_queue_len = 300
icm/min_threads   = 5
icm/max_threads   = 15
mpi/total_size_MB = 20

#maximum number of concurrent connections to one server
wdisp/HTTP/max_pooled_con = 100

wdisp/HTTPS/max_pooled_con = 100

## SAP Web Dispatcher Ports
icm/server_port_0 = PROT=HTTP,PORT=80

## Open an https listening port towards the Internet
icm/server_port_1 = PROT=HTTPS, PORT=443

## SAP Web Dispatcher Web Administration
icm/HTTP/admin_0 = PREFIX=/sapdisp/admin,DOCROOT=./admin,AUTHFILE=icmauth.txt

##configuration for enabling the use of the SAP Crypto software library
ssl/ssl_lib = C:\sunlight\webdispatcher\sapcrypto.dll

##specify the SSL server PSE to be used for outbound Internet communication
ssl/server_pse = C:\sunlight\webdispatcher\sec\SAPSSLS.pse
```
With this configuration you enable Web Dispatcher to process secure https requests.

You need to restart Web Dispatcher every time you change its profile in order for the changes to take effect. If Web Dispatcher runs on Windows in command prompt, you can use CTR+Pause/ Break keys on your keyboard to stop Web Dispatcher. Use the following command to start it:

\[Your Web Dispatcher Directory path\]>sapwebdisp.exe pf=\.\sapwebdisp.pfl

Web Dispatcher should start without error messages. It writes a log/trace file named dev_webdisp, which can be useful for troubleshooting in case of an error.

Perform an https ping test:

https://<your host.domain:port>/sap/bc/ping

You should get a proper response. If you check the server certificate on the response, it should match the one you configured originally for your SSL server PSE/ key store.

In the next configuration step, you should configure the SAP Cloud certificate authentication and forwarding to the ABAP back-end. Add the following lines at the end of your Web Dispatcher spawebdisp.pfl file:

##ask for and authenticate SAP Cloud X.509 client certificate
icm/HTTPS/verify_client = 1

##forward an SAP Cloud client X.509 certificate in the http header to the ABAP system
##via http
icm/HTTP/forward_ccert_as_header = TRUE

This step is a bit of a temporary workaround for the purpose of verifying that Web Dispatcher is indeed forwarding the SAP Cloud client certificate correctly. The X.509 based authentication is not an option offered in the SOAMANGAER transport security configuration. In order to use http in between Web Dispatcher and the ABAP system, we need to switch to use of http and user/ password authentication for an intermediate test as shown below.

![Configuration of Web Service](image-url)
The SoapUI tool allows you to configure user/password authentication as well.

With these settings in place, you get correct responses to your test web service calls and it is worthwhile to check the Web Dispatcher and ABAP transaction SMICM traces with highest trace level 3 enabled.

In the Web Dispatcher trace, you can find the following characteristic entries:

```
[Thr 2720] Client Certificate available (FCPath-Len= 1)
[Thr 2720] New session (TLSv1.0)
[Thr 2720] HexDump of native SSL session ID { &buf= 0284F93C, buf_len= 32 } 
[Thr 2720] 00000: 20 Db 25 9c b4 e7 ec a9 4f 50 6c b4 a9 42 45 f8 .%..... OPl..BE.
[Thr 2720] 00010: 83 32 39 55 fa cb e0 17 90 73 e3 e4 db 50 e8 27 .s...P.
[Thr 2720] Base64-Dump of peer certificate (len=962 bytes)
[Thr 2720] -----BEGIN CERTIFICATE-----
MIIDvjCCAqagAwIBAgIBAjANBgkqhkiG9w0BAQUFADBlMQswCQYDVQQGEwJVUzEL...
CgYDVQQLEwNJQ0MxGTAXBgNVBAMTEFNBUCBJQ0MgQ0xvdWQgQ0EwHhcNMTQwNDA5...
-----BEGIN CERTIFICATE-----
MIIDvjCCAqagAwIBAgIBAjANBgkqhkiG9w0BAQUFADBlMQswCQYDVQQGEwJVUzEL...
CgYDVQQLEwNJQ0MxGTAXBgNVBAMTEFNBUCBJQ0MgQ0xvdWQgQ0EwHhcNMTQwNDA5...
```

```
[Thr 2720] hSmWz15Wyay51y31qXLMvdCSnxTPdy1Y8dKx5+YXqXYB012tFqL2Enue2D1VYY0
[Thr 2720] TRZ2iJV4jQ1GYCYB1zV6E19a16dt.dsoLbBmx404KK7b0221BDQQFQX/bxXyJYK
[Thr 2720] ex88R0Ed5uiBrSQpQlJnAL/DfeWbsUm03SOM3pfcrggryY87sRNLfzoRbCuorXK
[Thr 2720] MR1Wc8riDrDGJKDqtY1WHLQXAmAQ28FQF32lm3SvctEfqjLRQAeUu9awcqa4MBVb
[Thr 2720] +caAZA09OMDC1GzcQcn78b1A+LmyoYJfUSB7t77KAd7vEqCp05wOIDAQABo28w
[Thr 2720] bTAMBgNVHRMBAf8EAjAAMB0GAIuDgQWBB72BrV24y1Q8rmhm5cK12D1Ly9cTAL
```
This entry tells us that our SoapUI client, which stands for the SAP Cloud client in our testing, is sending its X.509 certificate. In the ABAP system's SMICM transaction's trace, you can see the incoming http-post request of the web service call in clear text. Note the header field ssl_client_cert... This shows that the original SAP Cloud client certificate was indeed sent along by the SAP Web Dispatcher as http header.

Two more related header fields can be seen if you scroll down a bit in the trace.
The Web Dispatcher configuration should be finalized in the next step. For that, you should restore the transport security of your test web service to use of https and X.509 client certificate authentication. Remove the user/password authentication in your SoapUI test tool as well.

Then edit the following modifications into the Web Dispatcher’s sapwebdisp.pfl profile:

Replace the line:
```
icm/HTTP/forward_ccert_as_header = TRUE
```
with the following line because we are switching from http to https use between Web Dispatcher and ABAP system:
```
icm/HTTPS/forward_ccert_as_header = TRUE
```

Add the following lines:

```
##define the client key store SAP WebDispatcher should use to find its X.509 certificate
ssl/client_pse = C:\sunlight\webdispatcher\sec\SAPSSLC.pse

## enable secure communication between SAP Web Dispatcher and ABAP system
## 0=SSL offloading (default), 1=use SSL to back-end if request is SSL, 2=always use SSL towards back-end.
wdisp/ssl_encrypt = 1

##This parameter determines which X.509 client certificate of the SAP Web Dispatcher can be used with which application servers.
##The following values are permitted:
##0: No certificate
##1: Default certificate from the PSE specified through the ssl/client_pse parameter
##2: Use the certificate specified in the wdisp/ssl_cred parameter.
wdisp/ssl_auth = 1

##The internal hostname in URL used by WebDispatcher to forward requests to the ABAP server is different from the Common Name we used in the ABAP server certificate.
```
## Either configure to ignore that mismatch or change the
## ABAP SSL server certificate to have a CN name equal
to the ABAP system's internal fully qualified domain name.

```plaintext
wdisp/ssl_ignore_host_mismatch = TRUE
```

Save the file, restart Web Dispatcher and test again.

Your web service request should get proper responses and be properly authenticated in the Web Dispatcher as reverse proxy and the ABAP system. You might do some last checking in the Web Dispatcher and ABAP-SMICM trace files. The Web Dispatcher trace file contains the following entries after a successful web service request routing:

```
[Thr 2584] -----BEGIN CERTIFICATE-----
[Thr 2584] MIIDvjCCAqagAwIBAgIBAjANBgkqhkiG9w0BAQUFADB1MQswCQYDVQQGEwJVUzEL
[Thr 2584] -----END CERTIFICATE-----
[Thr 2584] Subject DN: CN=SomeClientID, OU=SAP ICC Cloud, O=SAP ICC, L=Palo Alto,
SP=CA, C=US
[Thr 2584] Issuer DN: CN=SAP ICC Cloud CA, OU=ICC, O=SAP, L=Palo Alto, SP=CA, C=US
```

Web Dispatcher is getting the SAP Cloud client certificate and is authenticating it. A bit further down in the trace you find:

```
[Thr 2584] -----BEGIN CERTIFICATE-----
[Thr 2584] MIIDtzCCAp+gAwIBAgIBATANBgkqhkiG9w0BAQUFADB1MQswCQYDVQQGEwJVUzEL
[Thr 2584] -----END CERTIFICATE-----
[Thr 2584] SapSSLGetPeerInfo2(sssl_hdl=0280F060)==SAP_O_K
[Thr 2584] ... = 
[Thr 2584] HttpModHandler: add cert to headers: cert_array_len=2, cipher_id_len=2,
cipher_size=128
```

The SAP Cloud client certificate is added to the http header of the forwarding request to the SAP back-end system. Again, a bit further down in the trace you'd find:

```
[Thr 2584] SapSSLSetTargetHostname(sssl_hdl=034AE5B0)==SAP_O_K
[Thr 2584] in: hostname = "crm371.dummy.nodomain"
```

This is the internal, fully qualified domain name configured for the ABAP system

```
[Thr 2584] Server's List of trusted CA DNames (from cert-request message):
[Thr 2584] #1 "CN=SAP ICC CA, OU=ICC, O=SAP, L=Palo Alto, SP=CA, C=US"
[Thr 2584] #2 "CN=SAP ICC Cloud CA, OU=ICC, O=SAP, L=Palo Alto, SP=CA, C=US"
[Thr 2584] #3 "CN=SAP ICC Proxy CA, OU=SAP ICC, O=SAP, L=Palo Alto, SP=CA, C=US"
```

Note the three Certificate Authorities Web Dispatcher is recognizing through having their root certificates in its key stores. #1 is the root certificate of the CA used for signing the ABAP server certificate. #2 is the root certificate of the CA used for signing the SAP Cloud X.509 client certificate. #3 is the root certificate of the CA used for signing Web Dispatchers X.509 client certificate. #1 and #3 have been imported through the Web Dispatcher's client-side key store (parameter ssl/ client_pse). #2 is part of the server-side key store.

```
[Thr 2584] -----BEGIN CERTIFICATE-----
[Thr 2584] MIIDwTCCAqmgAwIBAgIBCDANBgkqhkiG9w0BAQUFADBfMQswCQYDVQQGEwJVUzEL
[Thr 2584] -----END CERTIFICATE-----
[Thr 2584] Subject DN: CN=ec2-54-208-16-57.compute-1.amazonaws.com, OU=CRM, O=SAP Web AS, O=SAP ICC CA, C=US
```
Web Dispatcher received the ABAP server certificate.

```plaintext
[Thr 2584]   Issuer  DN: CN=SAP ICC CA, OU=ICC, O=SAP, L=Palo Alto, SP=CA, C=US
[Thr 2584]   Current Cipher: TLS_RSA_WITH_AES128_CBC_SHA
[Thr 2584]   MatchTargetName("crm371.dummy.nodomain", CN="ec2-54-208-16-57.compute-1.amazonaws.com") # MSMatch
[Thr 2584]   << ERROR: SapsSSLSessionStart(sssl_hdl=034AE5B0)==SSSLERR_SERVER_CERT_MISMATCH
```

The ABAP server certificate's Common Name (CN) field value doesn't match the ABAP server's internal fully qualified domain name. This mismatch will be ignored due to the wdisp/ssl_ignore_host_mismatch = TRUE parameter setting in the Web Dispatcher profile.

The ABAP-side trace of the transaction SMICM shows interesting details too:

It shows that the ABAP system receives the reverse Proxy's X.509 Client certificate. Through this certificate, the reverse proxy is authenticated as "trusted intermediary" (below).
Right below you find:

```
# Profile generated by sapwebdisp bootstrap
## Profile Extended for Security

## Trace Level: 1=production, 3=most detail
rdisp/TRACE = 1

# unique instance number
SAPSYSTEM = 01

# add default directory settings
DIR_EXECUTABLE = .
DIR_INSTANCE = .

# Accessibility of Message Servers
rdisp/mshost = crm371
ms/http_port = 8100

# SAP Web Dispatcher Parameter
wdisp/auto_refresh = 25
wdisp/max_servers = 100
wdisp/shm_attach_mode = 6

# configuration for small scenario
icm/max_conn = 100
icm/max.Sockets = 1024
icm/req_queue_len = 300
icm/min_threads = 5
icm/max_threads = 15
mpi/total_size_MB = 20

#maximum number of concurrent connections to one server
wdisp/HTTP/max_pooled_con = 100
wdisp/HTTPS/max_pooled_con = 100

# SAP Web Dispatcher Ports. You can disable this port for better security by deleting
# this line and renaming the next icm/server_port_1 parameter to icm/server_port_0
```

This means that the original SAP Cloud X.509 certificate has been found in the http header of the incoming request. This certificate is extracted and forwarded to an ABAP work process where it will be used for authentication and authorization.

This concludes the easy Web Dispatcher configuration. For use in productive systems, you should reduce the trace level from 3 to 1. As a summary, please find the complete Web Dispatcher configuration profile below again.
icm/server_port_0 = PROT=HTTP,PORT=80

## Open an https listening port towards the Internet
icm/server_port_1 = PROT=HTTPS, PORT=443

# SAP Web Dispatcher Web Administration
icm/HTTP/admin_0 = PREFIX=/sap/wdisp/admin,DOCROOT=./admin,AUTHFILE=icmauth.txt

##configuration for enabling the use of the SAP Crypto software library
ssl/ssl_lib = C:\sunlight\webdispatcher\sapcrypto.dll

##specify the SSL server PSE to be used for outbound Internet communication
ssl/server_pse = C:\sunlight\webdispatcher\sec\SAPSSLS.pse

##ask for and authenticate SAP Cloud X.509 client certificate
icm/HTTPS/verify_client = 1

##forward an SAP Cloud client X.509 certificate in the http header to the ABAP system
icm/HTTPS/forward_ccert_as_header = TRUE

##define the client key store SAP WebDispatcher should use to find its X.509
##certificate
ssl/client_pse = C:\sunlight\webdispatcher\sec\SAPSSLC.pse

## enable secure communication between SAP Web Dispatcher and ABAP system
## 0=SSL offloading (default), 1=use SSL to back-end if request is SSL, 2=always use
## SSL towards back-end.
wdisp/ssl_encrypt = 1

##This parameter determines which X.509 client certificate of the SAP Web Dispatcher
##can be used with which application servers.
##The following values are permitted:
##0: No certificate
##1: Default certificate
##2: Use the certificate specified in the wdisp/ssl_cred parameter.
wdisp/ssl_auth = 1

##do something about SSSLERR_SERVER_CERT_MISMATCH
##The external hostname in URL is different from the internal host name of the ABAP
##system. Either tolerate that or change the
##ABAP SSL server certificate to have a CN name equal to the ABAP system's fully
##qualified domain name.
wdisp/ssl_ignore_host_mismatch = TRUE

### 6.2.3 Advanced SAP Web Dispatcher Configuration Options

The SAP Web Dispatcher offers many more features than described in the previous sections; these can tighten up
security even further. Its capacity can be configured for different load requirements and its key stores can be
created and maintained in different ways.

In this section, we will focus on a different way to configure key stores. Information about Web Dispatcher
capacity sizing can be found at
Advanced security configuration features, such as URL filtering, are described in chapter 4 onwards of this guide:
https://websmp201.sap-ag.de/~form/sapnet?_SCENARIO=01100035870000000202&_SHORTKEY=00200797470000089947&_OBJECT=011000358700000121752008E
For the easy Web Dispatcher configuration, we outsourced the Web Dispatcher key store creation to the ABAP system for simplification. We can re-use the ABAP-server-side key store as Web Dispatcher server-side key store and we can use the graphical ABAP transaction STRUST to create the Web Dispatcher's client-side key store.

However, there might be situations where:

- The engineer configuring and operating Web Dispatcher has no access rights or there are other hindrances to accessing the STRUST transaction in the ABAP system.
- You do not want to re-use key stores of the ABAP back-end application system as key stores for a reverse proxy for security governance rule reasons in your company.
- You don’t use Web Dispatcher but another product as reverse proxy but you do want to re-use parts of the key stores of the ABAP layer and need to extract private keys and certificates from them as common PEM or other common formats, so that you can import them into your proxy’s key stores.
- You want to automate key store creation and maintenance steps.
- You need to protect key store files themselves with a password for added security. You grant an operating system user single-sign on rights to open protected key stores such that only that operating system user can run SAP Web Dispatcher.

Therefore, with SAP Web Dispatcher, SAP provides a command line tool sapgenpse.exe to create and manipulate SAP key store files in SAP's own .pse format. Sapgenpse.exe is included in the SAP Cryptographic library download. How to use sapgenpse is described at http://help.sap.com/saphelp_nw70/helpdata/en/a6/f19a3dc0d82453e10000000a114084/content.htm and http://help.sap.com/saphelp_nwmobile711/helpdata/en/56/a92f3ae689f058e10000000a11402f/content.htm?frameset=/en/41/Ifa63acd1e1a49e10000000a11402f/frameset.htm.

A quick overview of sapgenpse options can be gained by simply executing sapgenpse -h, see below.

![A quick overview of sapgenpse options](image)

In the following, we list a number of useful operations that might be performed with the sapgenpse tool. For demonstration purposes, we created a sapgenpse_test directory and copied need program and PSE files into it.

```
C:\SunLight\WebDispatcher>sapgenpse -h
SAPGENPSE tool for creation and management of PSE-files and SS0-credentials
Usage: sapgenpse [-h] [sub-optional] ...
Usage: sapgenpse -m <name>where\<postlogin_web.dll [command] [sub-optional] ...
(command) must be one of:
    gen_pse     create new PSE and/or PKCS#10 certification request
                (same as "get_pse")
    export_pse  export user certificate of a PSE
    export_pse12 export a PKCS#12 digital ID transport file
    print_p12   print a PKCS#12 digital ID transport file
    print_pse   print a PSE certificate
    get_my_name show attributes of the user certificate in a PSE
    fail_login  show address of the user certificate in a PSE
    maintain_pk  show/add/delete trusted keys/certs in PKList of PSE
    print_pse   export the user certificate of a PSE
    pass_pse    create new PSE from PKCS#10 private key plus certs
    getting "-h" after a (command) will show further help information
```

In the following, we list a number of useful operations that might be performed with the sapgenpse tool. For demonstration purposes, we created a sapgenpse_test directory and copied need program and PSE files into it.
Then we opened a command window and performed the operations shown in the next sub-sections.

6.2.3.1 Listing the content of an ABAP PSE-key store file

The command is:

`sapgenpse maintain_pk -l -p <PSE file path/name>`

When we apply this to the Server key store file of the ABAP and Web Dispatcher as described in section 6.2.3, we get the following output:

```
C:\sapgenpse_test>sapgenpse maintain_pk -l -p SAPSSLS.pse
maintain_pk for PSE "C:\PSE\SAPSSLS.pse"
*** Object <PKList> is of the type <PKList_OID> ***
1. -------------------------------------------------------------
   Version:         2 (X.509v3-1996)
   SubjectName:     CN=Your Proxy, OU=SAP, O=SAP ICC Proxy CA
   IssuerName:      CN=SAP ICC Proxy CA, OU=SAP ICC, O=SAP, L=Palo Alto, SP=CA, C=US
   SerialNumber:    03
   Validity         -  NotBefore:   Tue Apr 22 21:37:00 2014 (140422213700Z)
   -  NotAfter:     Sat Apr 22 21:37:00 2023 (230422213700Z)
   Public Key Fingerprint: 7E00 5969 004E 22D4 400A 2727 17FD 4BF7
   SubjectKey:      Algorithm RSA (OID 1.2.840.113549.1.1.1), NULL
   Certificate extensions:
   Subject Key Identifier:       BFB1 E56F 7094 A960 F9E2 0A30 53A2 EE16 2D1C 02FE
   Key Usage:                    digitalSignature keyEncipherment dataEncipherment
   Basic Constraints:            NOT allowed to act as a CA !
   Private extensions:
   netscape-cert-type (OID 2.16.840.1.113730.1.1):
   BitString (3 bits):      0 A0
   netscape-comment (OID 2.16.840.1.113730.1.13):
   IA5String:            |xca certificate                 |
2. -------------------------------------------------------------
   Version:         2 (X.509v3-1996)
   SubjectName:     CN=SAP ICC Cloud CA, OU=ICC, O=SAP, L=Palo Alto, SP=CA, C=US
   IssuerName:      CN=SAP ICC Cloud CA, OU=ICC, O=SAP, L=Palo Alto, SP=CA, C=US
   SerialNumber:    01
   Validity         -  NotBefore:   Wed Apr 09 00:32:00 2014 (140409003200Z)
   -  NotAfter:     Tue Apr 09 00:32:00 2024 (240409003200Z)
   Public Key Fingerprint: 072B BEFA BA85 34CC 67CF 011E B7E7 1225
   SubjectKey:      Algorithm RSA (OID 1.2.840.113549.1.1.1), NULL
   Certificate extensions:
   Subject Key Identifier:       3195 10CE CC95 61FF A5E9 D953 8B77 C9A6 6F29 047B
   Key Usage:                    digitalSignature keyAgreement keyCertSign
   Basic Constraints:            allowed to act as a CA !
   Private extensions:
   netscape-cert-type (OID 2.16.840.1.113730.1.1):
   BitString (8 bits):      0 07
   netscape-comment (OID 2.16.840.1.113730.1.13):
   IA5String:            |xca certificate                 |
3. -------------------------------------------------------------
   Version:         2 (X.509v3-1996)
   SubjectName:     CN=SomeClientID, OU=SAP ICC Cloud, O=SAP ICC, L=Palo Alto, SP=CA, C=US
   IssuerName:      CN=SAP ICC Cloud CA, OU=ICC, O=SAP, L=Palo Alto, SP=CA, C=US
```
SerialNumber: 02
Validity - NotBefore: Wed Apr 09 00:52:00 2014 (140409005200Z)
NotAfter: Sun Apr 09 00:52:00 2023 (230409005200Z)
Public Key Fingerprint: 2164 9DEE 4213 0E60 86C9 23AA 9362 71AD
SubjectKey: Algorithm RSA (OID 1.2.840.113549.1.1.1), NULL
Certificate extensions:
  Subject Key Identifier: F606 8576 E01C A241 2A6E 866E 5C28 8643 88BC BD71
  Key Usage: digitalSignature keyEncipherment dataEncipherment
  Basic Constraints: NOT allowed to act as a CA!
Private extensions:
  netscape-cert-type (OID 2.16.840.1.113730.1.1):
    BitString (3 bits): 0 A0
  netscape-comment (OID 2.16.840.1.113730.1.13):
    IA5String:
      | xca certificate |
4. ------------------------------------------------------------
Version: 2 (X.509v3-1996)
SubjectName: CN=SAP ICC Proxy CA, OU=SAP ICC, O=SAP, L=Palo Alto, SP=CA, C=US
IssuerName: CN=SAP ICC Proxy CA, OU=SAP ICC, O=SAP, L=Palo Alto, SP=CA, C=US
SerialNumber: 01
Validity - NotBefore: Tue Apr 22 00:30:00 2014 (140422003000Z)
NotAfter: Mon Apr 22 00:30:00 2024 (240422003000Z)
Public Key Fingerprint: 6388 262B 73F2 2808 ED9F 16A5 92EE 6152
SubjectKey: Algorithm RSA (OID 1.2.840.113549.1.1.1), NULL
Certificate extensions:
  Subject Key Identifier: 6889 2C52 EA47 B66C 3238 278D 0037 6800 689A A655
  Key Usage: digitalSignature keyCertSign cRLSign
  Basic Constraints: allowed to act as a CA!
Private extensions:
  netscape-cert-type (OID 2.16.840.1.113730.1.1):
    BitString (8 bits): 0 07
  netscape-comment (OID 2.16.840.1.113730.1.13):
    IA5String:
      | xca certificate |
This command lists only the client and CA root certificates we added to the server PSE. To see the server certificate itself, use the following command:

```
sapgenpse get_my_name -p <PSE file path\name>
```

For instance:

```
C:\sapgenpse_test>.sapgenpse get_my_name -p sapssls.pse
No SSO for USER "Administrator"
with PSE file "C:\sapssls.pse"
Subject : CN=ec2-54-208-16-57.compute-1.amazonaws.com, OU=CRM, OU=SAP Web AS, O=SAP ICC CA, C=US
Issuer  : CN=SAP ICC CA, OU=ICC, O=SAP, L=Palo Alto, SP=CA, C=US
Serialno: 08
KeyInfo : RSA, 2048-bit
Validity - NotBefore: Tue Apr 08 17:33:00 2014 (140408173300Z)
NotAfter: Sat Apr 08 17:33:00 2023 (230408173300Z)
```

### 6.2.3.2 Extracting Certificates from PSE Files

When you use a third party proxy, you might want to use the same server certificate as the ABAP server for simplicity and as was suggested for the easy Web Dispatcher deployment. As mentioned, before a standard transportable key store format is the PKCS#12 format. The command to export the server certificate from the ABAP PSE files into PKCS#12 files is:

```
sapgenpse export_p12 -p <PSE file path\name> -z <Password for PKCS#12 file> -C 0 <PKCS#12 file path\name>
```
Example:

C:\sapgenpse_test> sapgenpse export_p12 -p .\sapssls.pse -z abc -C 0 sapssls.p12

There is no further output from this command but you should find a .p12 file in your test directory now. You can use this file for import into your reverse proxy’s server keystore if your proxy accepts PKCS#12 files. Otherwise, you can import this file into the XCA tool we mentioned earlier.

As you can see in the XCA screenshot, the PSE exported a PKCS#12 file contained the server certificate, the private key of the server certificate, and the root certificate of the CA we used for signing the server certificate.

6.2.3.3  Building a Web Dispatcher server-side server side key store (PSE) from scratch with sapgenpse

As mentioned at the beginning of section 6.2.3, there are a number of circumstances where you might want to create a key store or PSE file for the SAP Web Dispatcher with the simple command line tool sapgenpse. The general approach is the same as creating a PSE in the ABAP STRUST transaction:

1. The initial PSE creation step creates a private/public key pair and a self-signed certificate
2. You need to extract a certificate signing request for your self-signed certificate
3. Use that signing-request file with your preferred Certificate Authority service or tool to get a certificate signing response and the root certificate of your CA.
4. Import the signing response into your PSE so that your certificate gets signed.
5. Add any number of client certificate and their CA root certificate to the PSE in order to allow Web Dispatcher to check for client certificates
For step 1 you can use the following command:

```
sapgenpse get_pse -s 2048 -a RSA -p <new PSE file path\name> -x <PSE password> -r <signature request file path\name> "<distinguished name parameters>
```

An example with parameters used earlier in this paper would be:

```
sapgenpse get_pse -s 2048 -a RSA -p SAPSSLS.pse -x secret -r .\srequest.txt "CN= ec2-54-208-16-57.compute-1.amazonaws.com, OU=SAP Web AS, O=SAP ICC CA, C=US"
```

Use the sequest.txt file in the example to create your signed server certificate with your CA (step 3). Use chained PEM format for the response so that your root certificate is included.

As step 4, import the signing response with the following command:

```
sapgenpse import_own_cert -c <response file path\name> -p <PSE file path\name> -x <PSE password>
```

For our example, the command looks like this:

```
sapgenpse import_own_cert -c .\sresponse.txt -p .\SAPSSLS.pse -x secret
```

The last step is to import the root certificates of the Certificate Authorities used to sign the ABAP back-end’s server certificate and the SAP Cloud’s client certificate. For the import of these root certificates, use the following command:

```
sapgenpse maintain_pk -a "<Certificate file path\name>" -p <PSE file path\name> -x <PSE password>
```

This has to be executed twice for the 2 root certificates to import:

```
sapgenpse maintain_pk -a "\.\ABAPRootCA.cer" -p .\SAPSSLS.pse -x secret
sapgenpse maintain_pk -a "\.\SAPCloudRootCA.cer" -p .\SAPSSLS.pse -x secret
```

### 6.2.3.4 Building a Web Dispatcher client-side key store (PSE) from scratch with sapgenpse

For the SAP Web Dispatcher client-side key store, you just have to repeat steps 1 through 4 described in the previous section. There are no additional root certificates to be imported, so step 5 does not apply. Be sure to name the client-side PSE file differently than the server-side one. The typical name is SAPSSLC.pse. Another detail to watch out for is to request a client certificate signing from your Certificate Authority service. In general, CA root certificates, server certificates, and client certificates can be distinguished for signing with some CAs.

### 6.2.4 Desirable Security and Other Configurations for the SAP Web Dispatcher

The major reason for having a reverse proxy and not connecting a business application back-end system directly to the Internet is security. There are many different attack vectors and about as many defense mechanisms. The network encryption and authentication schemes we described before are a must-have foundation. There are a
number of additional security measures. A reverse proxy can also cache static content, which can help to reduce request load to a back-end system. We list such features briefly in this section. More details can be found in the Web Dispatcher administration documentation on the SAP Help Portal: http://help.sap.com/saphelp_nw74/helpdata/en/48/99d142ee2b73e7e10000000a42189b/content.htm?frameset=/en/48/99d142ee2b73e7e10000000a42189b/frameset.htm&current_toc=/en/ed/2429371ec14c23a7508affa1280d07/plain.htm&node_id=15&show_children=true#jump15

6.2.4.1 Logging

For logging requests through the SAP Web Dispatcher, you may set two profile parameters for server and client-side traffic respectively. Here’s an example:

```
icm/HTTP/logging_0 = PREFIX=/, LOGFILE=access_log-%y-%m-%d, MAXSIZEKB=10000, SWITCHTF=day, LOGFORMAT=CLFMOD

icm/HTTP/logging_client_0 =  PREFIX=/, LOGFILE= access_client_log-%y-%m-%d, MAXSIZEKB=10000, SWITCHTF=day, LOGFORMAT= SAPSMD
```

This configuration starts new log files every day and tracks all requests (PREFIX=/) in different log formats. Please refer to http://help.sap.com/saphelp_nw74/helpdata/en/48/406e93ca2331c3e10000000a42189d/content.htm?frameset=/en/52/22dbaaeaf843beb0382eb70bcb29e/frameset.htm&current_toc=/en/f9/b6a344af114f95943d23c1a56d23c1a5692/plain.htm&node_id=26&show_children=false for the many ways log file writing can be customized to your needs.

6.2.4.2 Caching

Typically, 90% or more of the content of a web site is static, meaning objects like java scripts, jpg files … which don’t change often and are user session independent. Rather than requesting those objects time and again from a back-end server, they can be cached in a reverse proxy to lighten the request load on the back-end system. An example configuration for SAP Web Dispatcher caching is:

```
icm/HTTP/server_cache_0 = PREFIX=/, CACHEDIR=./cache
```

You need to add a subdirectory cache to your WD deployment directory, into which Web Dispatcher persists the content it can cache.

More details about WD caching can be found here http://help.sap.com/saphelp_nw74/helpdata/en/48/9cebc918a273e9e10000000a42189b/content.htm?frameset=/en/48/9cebc918a273e9e10000000a42189b/frameset.htm&current_toc=/en/ed/2429371ec14c23a7508affa1280d07/plain.htm&node_id=54

6.2.4.3 Filtering Requests

Two important security concepts are multilayered defense and minimizing the attack surface. Multi-layer defense means that you implement redundant and complementary security features into multiple components in the network traffic path from a client to a server. Such components are firewalls, reverse proxy and the back-end system itself. The idea is that if one defense layer is overcome by an attacker, there are still others behind it blocking the attacker. Thereby, simpler filters, which don’t require much processing power and
therefore can handle potentially high attack volumes, for instance from a denial of service attack, are put at the outer edges of a datacenter to the Internet, vs. the most intelligent filters are closer to the back-end or inside the back-end system itself, for instance the user authorization for data access.

Minimizing the attack surface is the idea to close up as many entry paths to your back-end system as possible and to leave only those paths open that are absolutely required. Filters are used in each of the multiple defense layers. They can be set up as whitelists, meaning they first disallow everything and then open up explicitly defined access points, or as blacklists, which allow everything and then exclude certain access points. Some filter examples are:

- Firewalls typically let pass only certain protocol requests on certain ports. That way, they might, for example, block telnet access, which mostly is reserved to internal system administrators only.
- Reverse proxies can apply URL filters and more.
- Back-end application systems could restrict access only to certain authenticated outside clients as we showed before with the x509 certificate configurations.

In the SAP Web Dispatcher, filtering is configured through the icm/HTTP/auth_<xx> parameter. You can filter requests according to the following criteria:

- URL
- Client IP address
- Server IP address
- User name/user group and password
- String search in the URL

For the use case of SAP Cloud to customer on-premise application integration, it would make sense to use the first 3 filters. For the example systems used in this paper, the parameter settings would look like this:

```
##filters
#URL filter set to /. Only secure requests like https://<domain:port>/ are passed on#
icm/HTTP/auth_0 = PREFIX=/, PERMFILE=perm_sec.txt

#client and server IP address filter are checked, see perm.txt file
icm/HTTP/auth_1 = PREFIX =/, PERMFILE=perm.txt
```

Multiple filters can be defined and are applied to incoming traffic in order of their enumeration. For performance reasons, simple filters should come first. The PREFIX defines the URL path to which a filter applies. "/" means just any URL. The filter action is defined in permission files. For the perm_sec.txt files we chose as example:

```
# P/D/S <URI pattern> <USER> <GROUP> <CLIENT IP> <SERVER IP>
S /  
S */
```

These two lines allow only https requests to pass. Insecure http requests would be rejected. Ensure that you have no empty lines between comment ("#") and filter lines.

The second filter perm.txt file could look as follows:

```
# P/D/S <URI pattern> <USER> <GROUP> <CLIENT IP> <SERVER IP>
P /sap/ * * 54.x.x.x 10.y.y.y
P /sap/* * * 54.x.x.x 10.y.y.y
#allow localhost request to go through for admin access
P /sap/ * * 127.0.0.1 *
P /sap/* * * 127.0.0.1 *
```

"P" stands for permit, so this file contains only whitelist entries. First access is restricted to URLs starting with / sap. The "*" wildcards define that there are no user and group restrictions; we then added Client IP and Server IP restrictions. Any request not matching one of the P rules is denied.

The intention of a narrow client and server IP restriction is to just allow the SAP Cloud solution as Client to talk to the SAP Web Dispatcher as server. No other Client would be allowed. You can get the client IP address of an SAP Cloud solution from the documentation of that solution.
While such source IP filtering is a good security feature required by many customers, it needs careful consideration, since IP addresses can change as they are passed on through network devices on the traffic pass. For instance, it might be that the SAP Web Dispatcher can be reached from the Internet through having its own Internet Server IP address opened up. However, special Internet edge routers your company might have, might perform a Network Address Translation (NATing), which changes the external server IP address to the internal IP address of the Web Dispatcher server. Likewise, if your company would use another proxy in front of SAP Web Dispatcher, the client IP the SAP Web Dispatcher gets might not be that of an SAP Cloud solution any longer but that of the proxy. In conclusion, you should set up server and client IP address filters only in close cooperation with your network group, since settings can depend on the network environment around your SAP landscape. Another pitfall of source and client IP filtering is that you might accidentally lock out the access to the Web Dispatcher administration and monitoring web site. Therefore, we permitted also access from "localhost", meaning a browser running on the Web Dispatcher server itself is allowed to have access to the admin websites through the admin URL: https:// localhost/ sap/ wdisp/ admin

There are many more filtering and other rules that can be defined through icm/ HTTP/ auth. <xx> configurations, see http:// help.sap.com/ saphelp_nw74/ helpdata/en/ 48/ 3edf38c1d272d2e10000000a42189c/ content.htm for more details and examples.

6.2.4.4 Modification Rules

Modifying web requests in various ways is another task most reverse proxies offer. The SAP Web Dispatcher can be configured for modifications through the following parameter:

#using modification functions
icm/ HTTP/mod_<xx> = PREFIX=/, FILE=action.txt

This parameter works similarly to the filtering parameter described in the previous section. There can be multiple enumerated modification rules. They are applied to the URL path defined as PREFIX value and the type of modification is defined in a file, here called action.txt. A simple example of such file content is:

# Modification rules for WebDisp
# set http Host header
SetHeader Host <reverse proxy host>

This example patches a minor inconsistency reverse proxies can cause. They might just forward http headers as they get them from an outside client to the server, whereas in fact the reverse proxy is the http host of the forwarding request to a back-end server. The SetHeader modification allows to override such inconsistency. An alternative we described earlier is to configure reverse proxies and back-end systems for more tolerance against http host header mismatches through the parameter:

wdisp/ssl_ignore_host_mismatch = TRUE

6.2.5 SAP Web Dispatcher Monitoring and Administration Web Interface

Over time, the SAP Web Dispatcher web-administration and monitoring capabilities are extended. The current release at the time of writing this paper is 7.41. One of the major new features is key store/ PSE management capabilities, which can be used instead of the sapgenpse command line tool described in section 6.2.3

The default URL to call for the web administration interface is:

https://<WD address:admin port>/sap/wdisp/admin
You are prompted for the administrative user and password for login. After login, you can explore various functions through a left hand navigation plane.

For instance, the above picture shows the PSE Management site. For more details on SAP Web Dispatcher web-administration see http://help.sap.com/saphelp_nw74/helpdata/en/48/80c48a109a1b5ae10000000a42189c/content.htm