F5 Networks

SSL Orchestrator 9.1 (BIG-IP 16.1.1)

Lab Guide (UDF3 Edition v1)

Participant Hands-on Lab Guide

Solutions for
an application world.

Last Updated: *12.2021*

k.stewart@f5.com

Table of Contents

WHAT IS THE F5 SSL ORCHESTRATOR?

F5 SSL Orchestrator (SSLO) provides an all-in-one appliance solution designed specifically to optimize the SSL infrastructure, provide security devices with visibility of SSL/TLS encrypted traffic, and maximize efficient use of that existing security investment. This solution supports policy-based management and steering of traffic flows to existing security devices, designed to easily integrate into existing architectures, and centralizes the SSL decrypt/encrypt function by delivering the latest SSL encryption technologies across the entire security infrastructure.

Multi-layered security

To solve specific security challenges, security administrators are accustomed to manually chaining together multiple point products, creating a barebones "security stack" consisting of multiple services. A typical stack may include components like Data Leak Prevention (DLP) scanners, Web Application Firewalls (WAF), Intrusion Prevention and Detection Systems (IPS and IDS), Malware Analysis tools, and more. In this model, all user sessions are provided the same level of security, as this "daisy chain" of services is hard-wired.

Dynamic service chaining

Dynamic service chaining effectively breaks the daisy chain paradigm by processing specific connections based on context provided by the Security Policy, that then allows specific types of traffic to flow through arbitrary chains of services. These service chains can include five types of services: layer 2 inline services, layer 3 inline services, receive-only services, ICAP services, and HTTP web proxy services.

Topologies

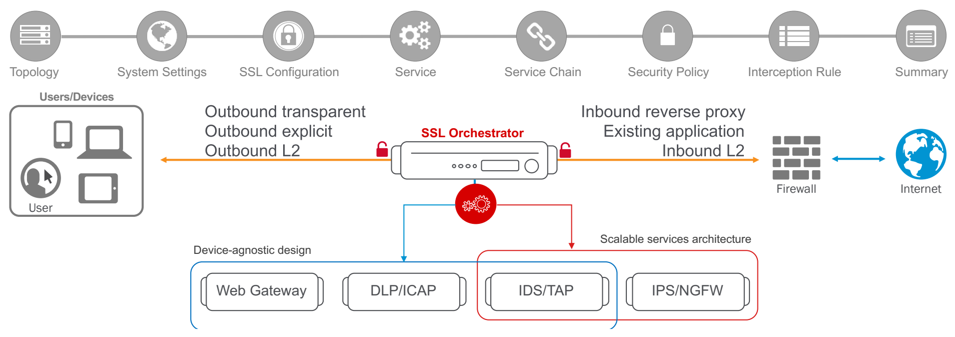
Different environments call for different network implementations. While some can easily support SSL visibility at layer 3 (routed), others may require these devices to be inserted at layer 2. SSL Orchestrator can support all of these networking requirements with the following topology options:

- Outbound transparent proxy
- Outbound explicit proxy
- Outbound layer 2
- Inbound reverse proxy
- Existing application
- Inbound layer 2

Security Policy

The SSLO Security Policy provides a rich set of context-aware methods to dynamically determine how best to optimize traffic flow through the security stack. Context can minimally come from the following:

- Source and destination address/subnet
- Source and destination port
- Source and destination IP geolocation
- Source and destination IP reputation
- URL filtering and IP intelligence - Subscriptions
- Host and domain name
- Source VLAN
- Protocol
- Server certificate subject/issuer/SAN
- TLS ClientHello Server Name (SNI)



WHAT'S NEW IN SSL ORCHESTRATOR?

SSL Orchestrator 4.0 provides significant architectural improvements over previous versions. Here are just a few of those updates:

- SSLO 4.0 replaces the complex iRules-based traffic classification and service chaining functions of previous versions with an Access per-request policy engine, providing much greater flexibility in traffic management options.
- SSLO 4.0 optimizes traffic flow through security services by replacing the complex "proxy hops" with a new "tee connector" – essentially a mid-proxy tap – that allows decrypted traffic to flow through security devices out-of-band from the main client-server proxy traffic. This is implemented as new "Service" and "Connector" profiles.
- SSLO 4.0 introduces new "split session" client and server SSL profiles, that are now responsible for carrying SNI signaling information across the inspection zone.
- SSLO 4.0 further optimizes traffic flow by reducing the amount of iRule data plane management, also making it easier to add customization iRules.
- SSLO 4.0 introduces three new network topologies. Along with the existing outbound transparent and explicit proxy flows, 4.0 now also supports inbound layer 3 (reverse proxy) inspection, and layer 2 transparent inbound and outbound topologies.

SSL Orchestrator 4.x also includes the following new functionality features:

- Explicit and transparent web proxy devices as an inline security service.
- Front-end explicit proxy authentication via APM integration (relies on existing SWG-Explicit access policy).
- FTPS (passive), SMTPS, POP3S, and IMAPS protocols inspection.
- ICAP advanced filtering via LTM CPM policy (relies on an existing CPM policy).
- URL filtering as a function of the Access per-request service chaining policy.
- Authentication headers - ability to define additional HTTP headers to pass to inline security services.
- vCMP support - ability to select existing VLANs for inbound and outbound to/from inline services.

SSL Orchestrator 5.x includes the following updates:

- Guided Configuration user experience, a complete refresh of the SSLO UI based on the Access Guided Configuration engine.
- Discreet "topology" definitions and the ability to define how SSLO listens for and processes traffic flows.
- Re-entrant, wizard-driven workflows. Based on the selected topology, SSLO 5.0 presents an intuitive workflow UI that walks the user through a simplified object creation process.

Note: VIPRION chassis platform support is not available in SSLO 4.0 and 5.0.

SSL Orchestrator 6.x includes the following updates:

- Transparent proxy captive portal authentication – In transparent forward proxy mode, an APM authentication profile (SWG-Transparent) can now be applied to perform captive portal-based client authentication.
- Reverse proxy (inbound SSLO) TLS 1.3 support – TLS 1.3 can now be handled on both client and server side of SSLO for inbound SSLO topologies.

- Service device monitor configuration – It is now possible to define the monitors applied to inline service definitions.
- Improved analytics dashboard – SSLO now provides a separate analytics dashboard with enhanced statistical information.
- VIPRION chassis support – SSLO can now function on VIPRION platforms, in both vCMP and non-vCMP configurations.
- Improved stability over previous versions

SSL Orchestrator 7.x adds the following new features:

- TLS 1.3 full proxy support for inbound and outbound flows – SSLO 6.0 included TLS 1.3 support for inbound (reverse proxy). This latest version now supports TLS 1.3 for outbound (forward proxy). A lab is dedicated to configuring TLS 1.3.
- Contextual security policies – In previous versions SSLO made no distinction between inbound and outbound flows for security policies, allowing inconsistent rule options to break traffic. SSLO 7.2 now creates separate inbound and outbound security policy types.
- Access to full IP Intelligence categories – This version provides access in the security policy to select specific IP Intelligence categories, versus simply 'good' or 'bad'.
- Update fix to URL category lookup when URLDB/SWG not provisioned – SSLO now correctly only queries custom URL categories if URLDB and/or SWG are not provisioned.
- Update fix to URL category lookup for custom categories – SSLO now correctly queries the categories directly based on http:// and https:// schemes. Previous versions only matched https:// URLs.
- Update fix to inline service load balancing – SSLO now correctly load balances inline services when port remapping is enabled.
- Strict Updates and modification enhancements – In previous versions when strict-updates was disabled on a configuration object, that object would become read-only in the SSLO UI. In SSLO 7.2, for most object types, strictness can be disabled and still editable in the SSLO UI. If any non-strict changes are made to the objects, deployment provides an option to keep those non-strict changes or overwrite. A lab is dedicated to strict updates modification.
- New HA Status UI – The HA Status UI provides a graphical view of HA state applicable to SSLO, including Gossip and Echo state. A lab is dedicated to configuring SSLO in HA mode.
- Several user interface, HA and upgrade stability enhancements – This SSLO version is mainly targeted at stability improvements, including UI, HA and upgrades.

SSL Orchestrator 8 adds the following new features:

- (8.0) Improved HA behavior – You can now upgrade SSLO instances without breaking HA.
- (8.0) Monitoring and remediation dashboard – The HA dashboard introduced in 7.x no supports a remediation option (button/link) to resolve any HA-related issues.
- (8.0) Strictness improvements – Expanding on the 7.x strict updates and modification enhancements (config merge), you can now preview the differences between existing and proposed configuration before committing.
- (8.0) Modify service network objects – You can now edit service network objects (VLANs, interfaces, self-IPs) directly within the security service configuration. You no longer have to delete and recreate a separate to change networking properties.
- (8.0) HA recovery script – Located at /usr/bin/ha-sync, this utility script can repair failed HA configurations.
- (8.1) Stability improvements – This is a bug fix release.

- (8.2) Data group security policy support – You can now define security policy rule matches using data sourced from data groups. Currently data group support is defined for client/server IP geolocation, client/server IP subnet match, and client/server port match as criteria in the security policy.
- (8.2) SWG as an inline security service (early access) – You can now insert F5 Secure Web Gateway (SWG) as an inline security service. SWG can be used to define transactional (per-request) policies for URL filtering, classification, and request/response analysis. This version is early access (hidden) as the underlying Forcepoint licensing updates are not baked in yet. A lab is dedicated to configuring SWG as an inline security service.

WHAT'S NEW IN SSLO 9?

SSL Orchestrator 9 adds the following new features:

- **(9.0) New session abort ending for blocked TLS session** – (STIP update – FDP_STIP_EXT.1.5) The SSL Orchestrator security policy contains a new abort option as a blocking action.

- **(9.0) SNI-based bypass based on client hello** – (STIP update – FDP_TEP_EXT.1.2) The security policy can now be configured to bypass TLS decryption based on SNI received in Client Hello, and before any server-side evaluation. In previous versions a connection would break in scenarios where TLS bypass by unique SNI/host was required for mutual TLS authentication. The elements of the security policy that perform this evaluation required a server-side "look", which would then break on the server side due to the client certificate requirement. In 9.0, a TLS bypass can be created, by SNI/host name, that will not break mTLS.

- **(9.0) Send SSL session logs using log publisher to one or more log destinations** – (STIP update – FAU_SAR.3) SSL Orchestrator can now log the details of the forged server certificate, and pass that to external log consumers via log publishers. This required STIP (CC) mode to be enabled in order for tmm to generate SSL session logs.

- **(9.0) Support authorityKeyIdentifier extension for certificates** – (STIP update – FDP_CER_EXT.1.2) SSL Orchestrator now includes an authorityKeyIdentifier (AKI) in the forged server certificate to aid in certificate path discovery at the client. Path discovery is the mechanism that a TLS client performs to find and build a complete chain of trust from the end-entity (leaf) certificate to the explicitly trusted root CA. In previous versions, the client would need to perform certificate path discovery based on subject and issuer common name string values, where the issuer name in a child certificate matches the subject name in the issuer certificate, and continued up to the self-signed root CA. This can cause issues in scenarios where a CA root certificate is "cross-signed" with another authority, creating two versions of the CA (a self-signed and issued certificate with the same common name). The AKI and SKI values are a SHA hash of a unique public key, making path discovery more reliable. In this case, instead of using subject and issuer string names, the authorityKeyIdentifier (AKI) value in the leaf certificate will match the subjectKeyIdentifier (SKI) of its parent, and continued up to the self-signed root CA.

- **(9.0) SSL profile switching and wildcard domain name matching from client hello SNI** – (STIP update – FDP_TEP_EXT.1.5) – This feature will allow an outbound topology to switch its SSL profiles based on client hello SNI matches. The most common use cases here might be switching a CA issuer for different tenants, or bypassing TLS for mutual TLS sessions by SNI host name. Only data-plane support in TMOS 16.1.0, configurable in SSL Orchestrator 9.2.

  Note: *STIP stands for "SSL/TLS Intercept Proxy" and is part of a larger US Federal NIAP (National Information Assurance Partnership) certification. F5 intends to achieve NIAP certification through a collection of enhancements per the standard to the SSL Orchestrator solution. As a result, F5 customers will achieve a higher level of assurance that the product meets the industry best practices in the SSL visibility solution space.*

- **(9.0) Gossip removal** – Gossip is the process whereby the SSL Orchestrator communicates HA sync information to its peer. This information is bidirectional and independent of typical BIG-IP "CMI" sync functions. In previous versions, gossip has proven to be an issue, particularly when configurations fail. In this release, the gossip sync function is removed and replaced with a new sync option (see source-of-truth update below).

- **(9.0) Support for local OCSP responder** – SSL Orchestrator now supports a local OCSP certificate revocation responder service for forged server certificates. The forged certificates can contain an authorityInfoAccess (AIA) attribute that points to a locally defined URL (a listening service on BIG-IP) that provides OCSP revocation status on the forged certificates.

- **(9.0) HTTP/2 (ALPN) support** – SSL Orchestrator now supports full-proxy HTTP/2 through the decrypted service chain. This is a "phase 1" feature where HTTP/2 is not de-multiplexed, so security services must also support decrypted HTTP/2. In phase 2, SSL Orchestrator will be able to de-mux to HTTP/1.1 inside the service chain.

- **(9.0) Verified Accept support** – When enabled, the system verifies that the pool member is available to accept the connection by sending the server a SYN before responding to the client's SYN with a SYN-ACK packet. SSL Orchestrator topologies are built on a standard virtual server type (with Verified Accept disabled). This could cause an issue in the scenario where an upstream firewall might block a specific IP and/or port, but the SSL Orchestrator closer to the client allows the connection (only to be blocked later at the firewall). Enabling Verified Accept allows the F5 to test for a valid server-side connection before completing the client-side handshake.

- **(9.0) Tabbed service catalog** – The security products in the SSL Orchestrator service catalog are now represented in a new de-cluttered tabbed interface, separated as inline layer 2, inline layer3, inline HTTP, ICAP, TAP, and a new F5 services tab. The F5 services tab will initially include SWG, but will eventually contain other F5 offerings, including WAF, tenant restrictions, IPS, and others.

- **(9.0) Per-topology DNS settings** – Explicit proxy topologies can now define independent DNS resolver settings.

- **(9.0) Pool member change from 6 to 20 with Standalone license** – SSL Orchestrator Standalone licensing has always limited the number of service pool members to 6 devices. This licensing update increases that number to 20.

- **(9.0) Custom category lookup performance enhancements** – Significant improvements have been made to increase custom category lookup performance.

- **(9.0) TCP keepalive passthrough** – The feature allows for dynamically setting TCP keepalive proxy settings, allowing an SSL Orchestrator interception rule to enable and proxy keepalive traffic. This is useful in situations where a client (ex. Citrix Workspace client) requires a keepalive to stay connected to the upstream server.

- **(9.0) Reject client with TLS alert if per-request policy reject ending is reached prior to SSL forward proxy triggering server-side TLS handshake** – On a Reject policy ending, SSL Orchestrator can now generate a

TLS alert in the client-side TLS handshake. In previous versions a Reject policy ending would either result in a TCP close or blocking page.

- **(9.0) Control plane re-architecture (source-of-truth and HA improvements)** – Significant improvements have come to the SSL Orchestrator control plane. In previous versions, the source-of-truth for the SSL Orchestrator configuration is JSON block storage (file-system objects). This separate source-of-truth (vs. native MCP configuration state) has created challenges, including some instability, configuration latency, and HA issues. In 9.0 the source-of-truth is still in separate JSON format, but now stored in iFile objects. This change allows SSL Orchestrator to utilize native MCP/CMI HA sync functions, and thus allows it to now support native automatic and incremental sync.

- **(9.0) Control plane re-architecture (removal of UI strictness)** – iApp strictness has always been a challenge in the SSL Orchestrator solution, at times making out-of-changes either complex, or impossible. In 9.0, the strictness lock icon has been removed from most objects, allowing you to freely make out-of-band changes that are honored by the SSL Orchestrator configuration throughout management and upgrade. The only significant object to retain strictness is the security policy, as moving between a constrained security policy ruleset and unconstrained visual policy editor would be untenable to manage.

- **(9.1) Inbound Gateway Mode support** - SSL Orchestrator now provides SSL Visibility for inbound connections to servers behind BIG-IP in two modes.
    - Gateway Mode: The Gateway mode works like a router where a virtual uses a network address to process incoming connections for the range.
    - Application Mode: The Application mode works like a traditional LTM Virtual Server. It creates a virtual listening for a specific IP: Port and processes incoming connections for this IP.

- **(9.1) SNI switching with multi SNI support** - SNI switching allows a virtual server to contain multiple client SSL profiles, each with its end-entity certificates. The SSL Orchestrator UI now supports selecting multiple SSL profiles to the same virtual for both Inbound and Outbound explicit topologies.

- **(9.1) Verified Accept SSL profile optimization** - SSL Orchestrator now generates a single SSL profile instead of two profiles for Verified Handshake True (vht) and Verified Handshake False (vhf), greatly simplifying the SSL Orchestrator-generated configurations. By default, the verified Handshake will be enabled for Outbound traffic and disabled for Inbound traffic.

- **(9.1) Post Remap enhancement for SSL Orchestrator services** - Some security devices require HTTPS (443) traffic to be remapped to HTTP (80) for correct inspection. Previously, the remap setting was applied regardless of bypass/decrypt decisions. With this release, the bypass traffic is not remapped, and the Port remap applies only to the decrypted traffic.

- **(9.1) Port Lockdown for HA pair allows Custom port** - The BIG-IP system allows administrators to configure Port Lockdown settings for Self-IPs to reduce the attack surface by restricting incoming traffic. Previously, any setting besides "Allow All" or "Allow Default" caused the SSL Orchestrator GUI to malfunction and report High Availability failures. With this release, you can deploy/edit the SSL orchestrator configuration with the Port Lockdown settings set to Custom (TCP port 443).

Please refer to the official SSLO 9 release notes for detailed update information.

This lab guide and corresponding **UDF3** lab environment are prepared for **SSLO 9.1** on a **BIG-IP 16.1.2** instance, using a new "consolidated" lab architecture. All security services are now consolidated down to a single Ubuntu server instance using a Docker Compose environment.

About the consolidated architecture

The consolidated architecture is an update to previous versions that vastly reduces the resources used in the UDF environment. This is accomplished by consolidating the security services onto a single Ubuntu 18.04 server instance using Docker Compose.

The project is detailed here: https://github.com/kevingstewart/sslo-consolidated-services/tree/main/udf.

Layer 3, HTTP, and ICAP security services, plus web servers are created as Docker containers via Compose file. The layer 2 and TAP services are hosted directly on the Ubuntu host with dedicated interfaces.

To identify that the container services are running, SSH to the Ubuntu 18.04 server and issue the following command:

```
$ docker ps
```

There should be four services running. If that is not the case, or there are other issues, it may be easiest to delete and recreate the entire Docker Compose environment to return to a pristine deployment:

```
$ cd /home/ubuntu/build/sslo-consolidated-services-main/udf
$ ./security-service-rebuild.sh
```

This script will destroy all containers and images, then re-download the images and rebuild new containers.

Should you need to log into any of the Docker container services directly:

```
$ docker exec -it icap /bin/bash
$ docker exec -it explicit-proxy /bin/bash
$ docker exec -it layer3 /bin/bash
$ docker exec -it apache /bin/bash
```

SSL ORCHESTRATOR LAB ENVIRONMENT

The lab environment for this guide has provided some prerequisite settings that you should be aware of. These are provided to make the demo simpler. All of the following would need to be configured manually in another environment.

- **Client side VLAN and subnet are pre-defined** – This is the VLAN that an internal client connects to for outbound traffic flows. SSLO does not define the client-side VLAN(s) and self-IP(s). A web server also exists on the client side VLAN to facilitate an inbound (reverse proxy) use case – external client to an internal set of websites.

- **Outbound side VLAN and subnet are pre-defined** – This is the VLAN that traffic egresses from SSLO to the Internet gateway. SSLO does not define the server-side VLAN(s) and self-IP(s).

- **ICAP service VLAN and subnet are pre-defined** – SSLO does not define the networking for this service type, so it has been pre-created in this lab.

- **CA certificate and private key are installed** – This is the CA certificate and private key that are used to re-issue (forge) remote server certificates to internal clients for outbound traffic flows.

- **Server certificate and private key are installed** – For the inbound (reverse proxy) traffic flow use case, SSL traffic is terminated at the F5, and re-encrypted on the way to the internal application environment. A wildcard server certificate is installed to facilitate using any name under the "*.f5labs.com*" sub-domain.

Note: It is a security best practice to isolate security devices within the protected network enclaves provided by SSLO. Customers will often desire NOT to move or change existing security services. However, while possible with SSLO 4.0 and beyond, passing this decrypted traffic to points on an existing network architecture could create multiple points of data exposure. Usernames, passwords, credit card numbers and other sensitive information could be exposed to other devices on that network. Each inline layer 3 security service definition includes an "Auto Manage" option. This option, enabled by default, provides internal network settings for security services to use, so that only the interface (and 802.1q VLAN tag as needed) is required to be defined for the inbound and outbound interfaces. Should customers opt to not follow security best practices, or simply need different networking settings, you can disable the Auto Manage option and define all of the required inbound and outbound networking setting manually.

SSL Orchestrator	BIG-IP management IP	10.1.1.x (UDF-managed)	
	Gateway IP/DNS	10.1.20.1	
	Login	admin:admin | root:default	
	Interfaces	Client VLAN	1.1
		Outbound VLAN	1.2
		ICAP service	1.3 (tagged)
		Inline L3 services	1.3 (tagged)
		Inline HTTP services	1.3 (tagged)
		Web servers	1.3 (tagged)
		Inline L2 service inbound	1.4
		Inline L2 service outbound	1.5
		TAP service	1.6

Inline layer 2 service	Login	-- Ubuntu 18.04 server host -- br0 (bridge) tied to ens7 and ens8 interfaces on host
	Services	Suricata

Inline layer 3 service	Login	`$ docker exec -it layer3 /bin/bash`		
	Interfaces	Inbound interface	1.3 tag 60	198.19.64.30/25
		Outbound interface	1.3 tag 70	198.19.64.130/25
	Services	Suricata		

Explicit proxy service	Login	`$ docker exec -it explicit-proxy /bin/bash`		
	Interfaces	Inbound interface	1.3 tag 30	198.19.96.30/25
		Outbound interface	1.3 tag 40	198.19.96.130/25
	Services	Squid	Port 3128	

Receive-only service	Login	-- Ubuntu 18.04 server host -- ens9 interface tied to tap service on host
	MAC address	12:12:12:12:12:12 (arbitrary if directly connected)

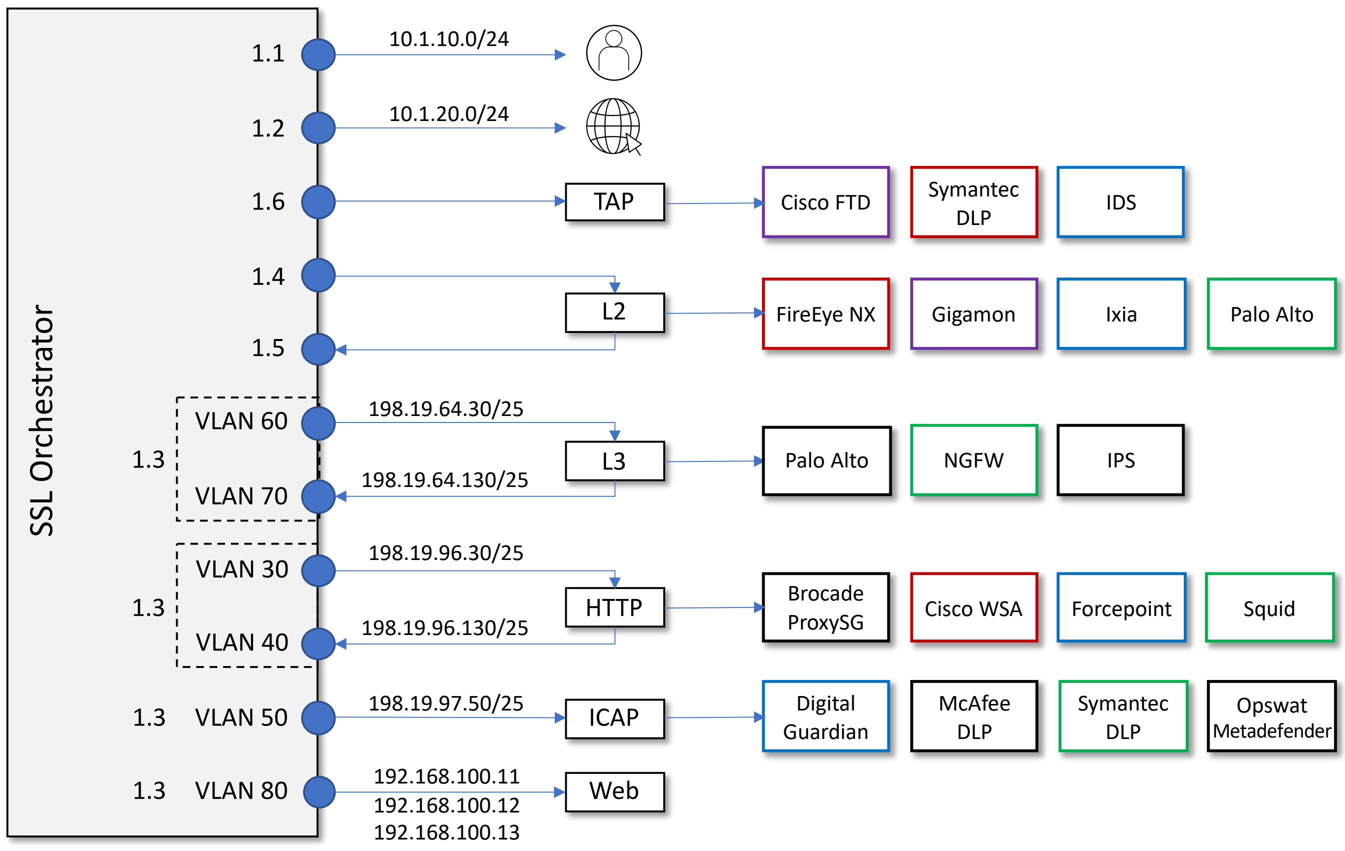
ICAP service	Login	`$ docker exec -it icap /bin/bash`
	IP Address:port	198.19.97.50:1344
	REQ/RES URLs	/avscan
	Services	ICAP Clamav

Internal web server	Login	`$ docker exec -it apache /bin/bash`
	IP Addresses *.f5labs.com	192.168.100.10 192.168.100.11 192.168.100.12 192.168.100.13
	Services	Apache2 (httpd 2.4)

Outbound client	Login	-- Ubuntu 18.04 client host --
	IP address	10.1.10.50 (RDP and SSH)

Inbound client	Login	-- Ubuntu 18.04 client host --
	IP address	10.1.20.50 (RDP and SSH)

The following is a visual representation of this UDF lab environment. The numbers inside the right edge of the SSL Orchestrator box indicate the port numbers assigned. The colored boxes to the right of the services indicate a few product examples for each respective service type.



LAB 1 – CREATE AN OUTBOUND LAYER 3 TOPOLOGY

The majority of enterprise transparent forward proxy configurations will involve a single or HA pair of F5 platforms performing the SSL visibility task. The SSL Orchestrator has been designed with that principle in mind and performs robust and dynamic service chaining of security devices. SSL Orchestrator now makes configuration of a single-box deployment simple and intuitive. Please follow the steps below to create a transparent forward proxy SSL Orchestrator configuration.

Step 1: Review the lab environment and map out the services and endpoints

Review the "SSL Orchestrator Lab Environment" section above. This lab will attach one of each type of security service (HTTP, ICAP, L2, L3, TAP) to SSLO for an outbound forward proxy traffic flow. Afterwards, an internal client will be able to access remote (Internet) resources through SSLO, providing decrypted, inspectable traffic to the security services.

- The client is attached to a *10.1.10.0/24* network and is assigned the IP *10.1.10.50*. This network is attached to the BIG-IP 1.1 interface.

- The **L2 device** is running on the Ubuntu 18.04 server host and configured to bridge its ens7 and ens8 interfaces. Its inbound VLAN (traffic to it) is attached to the BIG-IP *1.4* interface. Its outbound interface (traffic coming from it) is attached to the BIG-IP *1.5* interface. An instance of Suricata is running on the host and configured to listen on ens7.

- The **L3 device** is a Docker container configured to route between its eth1 and eth2 interfaces. Its inbound VLAN (traffic to it) is attached to the BIG-IP *1.3 (VLAN tag 60)* interface and has an IP of *198.19.64.30/25*. Its outbound interface (traffic coming from it) is attached to the BIG-IP *1.3 (VLAN tag 70)* interface and has an IP of *198.19.64.130/25*. Its default gateway is *198.19.64.245*, which will be a VLAN self-IP on the BIG-IP. The container is running Suricata.

- The **TAP** device is running on the Ubuntu 18.04 server host and configured to listen on the ens9 interface. That interface is attached to the BIG-IP *1.6* interface.

- The **DLP/ICAP** device is a Docker container configured with a single eth1 interface. That interface is attached to the BIG-IP *1.3 (VLAN tag 50)* interface and has an IP of *198.19.97.50 and listening on port 1344*. The container is running c-icap and Squid/Clamav.

- The **Explicit Proxy device** is a Docker container configured with Squid. Its interfaces are eth1 and eth2. Its inbound VLAN (traffic to it) is attached to the BIG-IP *1.3 (VLAN tag 30)* interface and has an IP of *198.19.96.30/25*. Its outbound interface (traffic coming from it) is attached to the BIG-IP *1.3 (VLAN tag 40)* interface and has an IP of *198.19.96.130/25*. Its default gateway is *198.19.96.245*, which will be a VLAN self-IP on the BIG-IP.

- The outbound network is attached to the BIG-IP *1.2* interface, in the *10.1.20.0/24* subnet, and has a gateway of *10.1.20.1*.

- The web servers are running in a Docker container and listening on *1.3 (VLAN tag 80)* at IP addresses *192.168.100.11*, *192.168.100.12*, and *192.168.100.13*, ports *80* and *443*.

- In the lab, client inbound, Internet outbound, and DLP VLANs and self-IPs are already created.

Step 2: Fulfill the SSL Orchestrator prerequisites

There are a number of objects that SSL Orchestrator does not create and expects to exist before deploying the iApp. You must create the following objects before starting the iApp:

- **Import the CA certificate and private key** – In order to terminate and re-encrypt outbound SSL traffic, SSL Forward Proxy must re-issue, or rather "forge" a new server certificate to the client. To perform this re-issuance process, the BIG-IP must possess a certificate authority (CA) certificate and associated private key. *This lab environment already has a subordinate CA certificate and private key installed*.

- **Create the client inbound VLAN and self-IP** – Create the VLAN and self-IP that connects the client to the BIG-IP. In this lab that's the *10.1.10.0/24* subnet and interface *1.1* on the BIG-IP. This lab environment already has this VLAN and self-IP created.

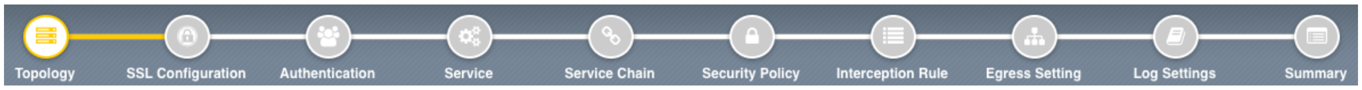
- **Create the Internet outbound VLAN and self-IP** – Create the VLAN and self-IP that connects the BIG-IP to the outbound Internet router. In this lab that's the *10.1.20.0/24* subnet and interface *1.2* on the BIG-IP. *This lab environment already has this VLAN and self-IP created*.

- **Create the DLP VLAN and self-IP** – If it is desired to isolate the DLP/ICAP device, create the VLAN and self-IP that connects the DLP device to the BIG-IP. In this lab that's the *198.19.97.0/25* subnet and interface *1.3 (VLAN tag 50)* on the BIG-IP. The DLP security device is listening on *198.19.97.50* and ICAP is listening on port *1344*. *This lab environment already has this VLAN and self-IP created*.

- **Create the default internet route for outbound traffic** – The iApp provides an option to leverage a defined gateway pool or use the system default route. If a gateway pool is not used, they system route table will need to have a default route used to reach Internet destination. *We'll use a gateway pool defined within SSLO*.

Step 3: Create the SSL Orchestrator deployment through Guided Configuration

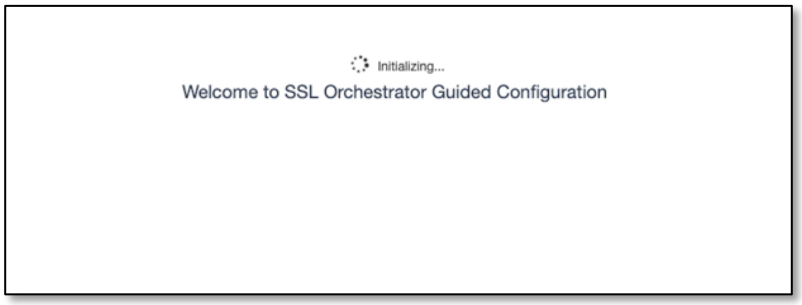
The SSL Orchestrator Guided Configuration presents a completely new and streamlined user experience. This workflow-based architecture provides intuitive, re-entrant configuration steps tailored to the selected topology.



The following steps will walk through the Guided Configuration (GC) to build a simple transparent forward proxy.

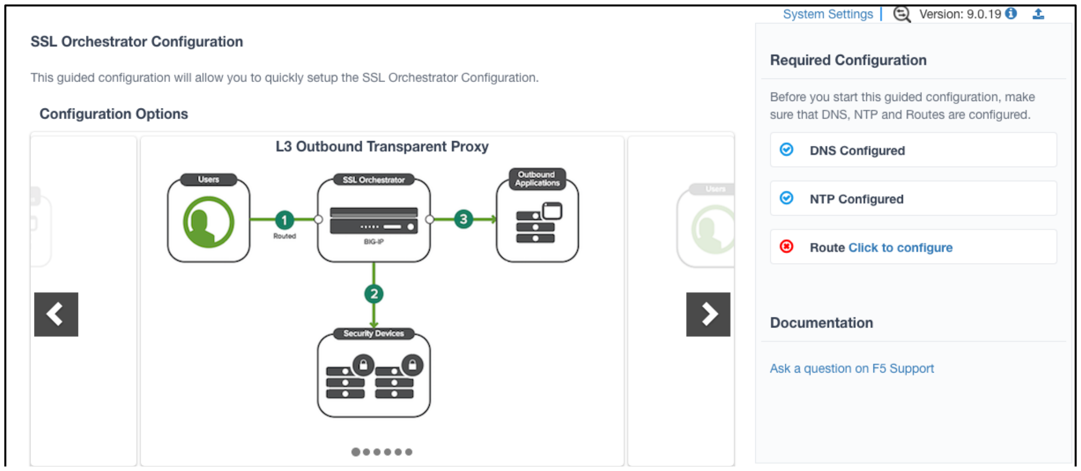
The following provides verbose details on each setting. For a more concise set of lab steps, without details, a lab 1 appendix is provided at the end of this guide.

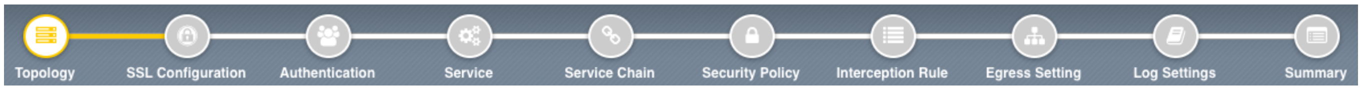
- **Initialization** – If this is the first time accessing SSLO in a new BIG-IP build, upon first access, GC will automatically load and deploy the built-in SSLO package.



- **Configuration review and prerequisites** – Take a moment to review the topology options and workflow configuration steps involved. Optionally satisfy any of the DNS, NTP and Route prerequisites from this page. Keep in mind, however, that aside from NTP, the SSLO GC will provide an opportunity to define DNS and route settings later in the workflow. No other configurations are required on this page, so click Next.

DNS settings have already been defined in this lab.


System Settings
Version: 9.0.19
SSL Orchestrator Configuration
This guided configuration will allow you to quickly setup the SSL Orchestrator Configuration.
Configuration Options
L3 Outbound Transparent Proxy
Users
SSL Orchestrator
Outbound Applications
Routed
BIG-IP
Security Devices
Required Configuration
Before you start this guided configuration, make sure that DNS, NTP and Routes are configured.
DNS Configured
NTP Configured
Route Click to configure
Documentation
Ask a question on F5 Support



- **Topology Properties** – SSLO creates discreet configurations based on the selected topology. An explicit forward proxy topology will ultimately create an explicit proxy listener and its relying transparent proxy listener, but the transparent listener will be bound only to the explicit proxy tunnel. If a subsequent transparent forward proxy topology is configured, it will not overlap the existing explicit proxy objects. The Topology Properties page provides the following options,

  The Protocol option presents four protocol types:

  - **TCP** – This option creates a single TCP wildcard interception rule for the L3 Inbound, L3 Outbound, and L3 Explicit Proxy topologies.

  - **UDP** – This option creates a single UDP wildcard interception rule for L3 Inbound and L3 Outbound topologies.

  - **Other** – This option creates a single any protocol wildcard interception rule for L3 Inbound and L3 Outbound topologies, typically used for non-TCP/UDP traffic flows.

  - **Any** – This option creates the TCP, UDP and non-TCP/UDP interception rules for outbound traffic flows.

  The SSL Orchestrator Topologies option page presents six topologies:

  - **L3 Explicit Proxy** – This is the traditional explicit forward proxy.

  - **L3 Outbound** – This is the traditional transparent forward proxy. An L3 outbound topology is effectively a “routed hop” configuration, where the SSLO topology listener becomes a routed path on the way to external (ie. Internet) resources.

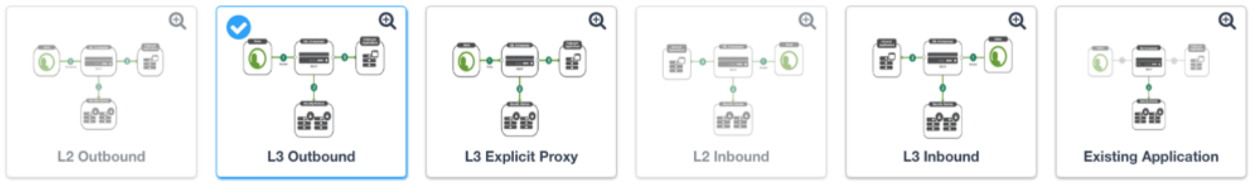
  - **L3 Inbound** – This is a reverse proxy configuration. Like the L3 outbound, L3 inbound is typically a routed hop configuration for traffic directed inbound. It can also behave as a traditional load-balanced application.

  - **L2 Inbound** – The layer 2 topology options insert SSLO as a bump-in-the-wire in an existing routed path, where SSLO presents no IP addresses on its outer edges. The L2 Inbound topology provides a transparent path for inbound traffic flows.

  - **L2 Outbound** – The layer 2 topology options insert SSLO as a bump-in-the-wire in an existing routed path, where SSLO presents no IP addresses on its outer edges. The L2 Outbound topology provides a transparent path for outbound traffic flows.

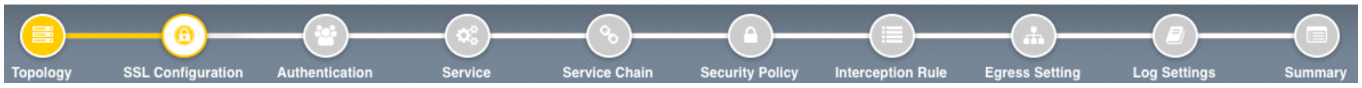
    It is important to distinguish SSLO’s layer 2 topology from those of other traditional layer 2 SSL visibility vendors. “True” layer 2 solutions like Blue Coat’s SSL visibility appliance (SSLVA) limit the types of devices that can be inserted into the inspection zone to layer 2 and below, and devices must be directly connected to the appliance. SSLO’s layer 2 topology only exists at the outer edges. Inside the inspection zone, full-proxy routing is still happening, so layer 3 and HTTP services can still function normally.

- **Existing Application** – This topology is designed to work with existing LTM applications. Whereas the L3 Inbound topology provides an inbound gateway function for SSLO, Existing Application works with LTM virtual servers that already perform their own SSL handling and client-server traffic management. The Existing Application workflow proceeds directly to service creation and security policy definition, then exits with an SSLO-type access policy and per-request policy that can easily be consumed by an LTM virtual server.



For this lab,

- **Name**: some name (ex. "demoL3")
- **Protocol**: Any – this will create separate TCP, UDP and non-TCP/UDP interception rules.
- **IP Family**: IPv4
- **Topology**: L3 Outbound
- Click Save & Next.



- **SSL Configurations** – This page defines the specific SSL settings for the selected topology, in this case a forward proxy, and controls both client-side and server-side SSL options. If existing SSL settings are available (from a previous workflow), it can be selected and re-used. Otherwise the SSL Configurations page creates new SSL settings for this workflow. The **[Advanced]** options below are available when "Show Advanced Settings" is enabled (top right). For this lab, create a new SSL profile,
  - **Client-side SSL**
    - **SNI Server Name (FQDN)**: Optionally specify the SNI Server Name (FQDN). If you attach multiple SSL profiles, one of them must be the default SNI: select the Default SNI check box. Only one default SNI is allowed. New in 9.1, multiple SSL profiles can be attached to a single topology, where the correct SSL profile is selected dynamically based on the incoming Client Hello Server Name Indication (SNI) extension. In a forward proxy scenario, this feature would typically be used to change attributes of the client SSL based on SNI, including client cipher properties and re-issuing CA.
    - **[Advanced] Processing Options**: SSLO now provides TLS 1.3 support for outbound topologies. TLS 1.3 configuration is described in a later lab, so for now leave this as is.

- **Cipher Type**: Cipher type can be a Cipher Group or Cipher String. If the former, select a previously-defined cipher group (from Local Traffic – Ciphers – Groups). If the latter, enter a cipher string that appropriately represents the client-side TLS requirement. For most environments, DEFAULT is optimal. For this lab, leave Cipher String selected.

- **Certificate Key Chain**: The certificate key chain represents the certificate and private key used as the "template" for forged server certificates. While re-issuing server certificates on-the-fly is generally easy, private key creation tends to be a CPU-intensive operation. For that reason, the underlying SSL Forward Proxy engine forges server certificates from a single defined private key. This setting gives customers the opportunity to apply their own template private key, and optionally store that key in a FIPS-certified HSM for additional protection. The built-in "default" certificate and private key uses 2K RSA and is generated from scratch when the BIG-IP system is installed. The pre-defined default.crt and default.key can be left as is. Click Done.

- **CA Certificate Key Chain**: An SSL forward proxy must re-sign, or "forge" remote server certificate to local clients using a local certificate authority (CA) certificate, and local clients must trust this local CA. This setting defines the local CA certificate and private key used to perform the forging operation. Click the pencil icon to Edit, then select subrsa.f5labs.com for both Certificate and Key, and click Done.

  SSL Settings minimally require RSA-based template and CA certificates but can also support Elliptic Curve (ECDSA) certificates. In this case, SSLO would forge an EC certificate to the client if the TLS handshake negotiated an ECDHE_ECDSA cipher. To enable EC forging support, add both an EC template certificate and key, and EC CA certificate and key.

- **ALPN**: The Proxy ALPN checkbox option specifies the ALPN hello extension to be proxied to the server by SSL Forward Proxy, allowing client to negotiate with server for protocols like HTTP/2. If the Proxy ALPN checkbox is not selected, then HTTP1.1 will be utilized by default.

- **Server-side SSL**

  - **[Advanced] Processing Options**: SSLO now provides TLS 1.3 support for outbound topologies. TLS 1.3 configuration is described in a later lab, so for now leave this as is.

  - **Cipher Type**: Cipher type can be a Cipher Group or Cipher String. If the former, select a previously-defined cipher group (from Local Traffic – Ciphers – Groups). If the latter, enter a cipher string that appropriately represents the server-side TLS requirement. For most environments, DEFAULT is optimal.

  - **[Advanced] Bypass on Handshake Alert**: This setting allows the underlying SSL Forward Proxy process to bypass SSL decryption if an SSL handshake error is detected on the server side. It is recommended to leave this disabled.

  - **[Advanced] Bypass on Client Certificate Failure**: This setting allows the underlying SSL Forward Proxy process to bypass SSL decryption if it detects a Certificate request message from the server, as in when a server requires mutual certificate authentication. It is recommended to leave this disabled.

The above two Bypass options can create a security vulnerability. If a colluding client and server can force an SSL handshake error, or force client certificate authentication, they can effectively bypass SSL inspection. It is recommended that these settings be left disabled.

- **Trusted Certificate Authority**: Browser vendors routinely update the CA certificate stores in their products to keep up with industry security trends, and to account for new and revoked CAs. In the SSL forward proxy use case, however, the SSL visibility product now performs all server-side certificate validation, in lieu of the client browser, and should therefore do its best to maintain the same industry security trends. BIG-IP ships with a CA certificate bundle that maintains a list of CA certificates common to the browser vendors. However, a more comprehensive bundle can be obtained from the F5 Downloads site. For this lab, select the built-in ca-bundle.crt.

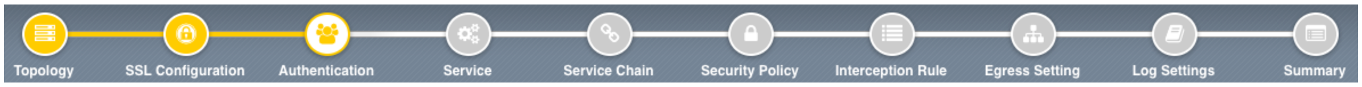
- **[Advanced] Expire Certificate Response**: SSLO performs validation on remote server certificates and can control what happens if it receives an expired server certificate. The options are **drop**, which simply drops the traffic, and **ignore**, which mirrors an expired forged certificate to the client. The default and recommended behavior for forward proxy is to drop traffic on an expired certificate.

- **[Advanced] Untrusted Certificate Authority**: SSLO performs validation on remote server certificates and can control what happens if it receives an untrusted server certificate, based on the Trusted Certificate Authority bundle. The options are **drop**, which simply drops the traffic, and **ignore**, which allows the traffic and forges a good certificate to the client. The default and recommended behavior for forward proxy is to drop traffic on an untrusted certificate.

- **[Advanced] OCSP Certificate Validator**: This setting selects an existing or can create a new OCSP profile for server-side Online Certificate Status Protocol (OCSP) and OCSP stapling. With this enabled, if a client issues a Status Request message in its Client Hello message (an indication that it supports OCSP stapling), SSLO will issue a corresponding Status Request message in its server-side TLS handshake. SSLO will then forge the returned OCSP stapling response back to the client. If the server does not respond with a staple but contains an Authority Info Access (AIA) field that points to an OCSP responder URL, SSLO will perform a separate OCSP request. The returned status is then mirrored in the stapled client-side TLS handshake.

- **[Advanced] CRL Certificate Validator**: This setting selects an existing or can create a new CRL profile for server-side Certificate Revocation List (CRL) validation. With this enabled, SSLO attempts to match server certificates to locally cached CRLs.

- Click Save & Next.



- **Authentication List** – In 9.0 a new Authentication List workflow exists to create authentication mechanisms for a topology. In this initial release, the Authentication List simply contains an OCSP Responder configuration. The list of authentication mechanisms will grow in subsequent versions.

  - **OCSP Responder** – You can configure a Local Online Certificate Status Protocol (OCSP) Responder and associate a Local OCSP Responder to a virtual server. OCSP is an Internet protocol used to obtain the revocation status of a digital certificate. When the validity of a certificate is requested, an OCSP request is sent to an OCSP Responder and checks the specific certificate with a trusted certificate authority. This results in an OCSP response being sent back of good, revoked, or unknown. To configure OCSP, you must select TCP or Any as your Protocol and either L2 Outbound, L3 Outbound, or L3 Explicit Proxy as your SSL Orchestrator topology from the Topology Properties screen. To create a new authentication, click Add. The Authentication Properties screen appears where you can select OCSP Responder (for the Client). Click OCSP Responder and click Add. The Authentication Properties screen appears where you can configure a new OCSP Responder. Later when configuring the Interception Rule, you may select from the Authentication section OCSP Responder list to associate a Local OCSP Responder into the Interception Rule. This action adds a new iRule to the virtual server. In addition, you may configure authentication using the mini-flow Authentication tab without creating a topology and may utilize the existing iRule item-selector to select the OCSP iRule.

    Essentially, this OCSP Responder configuration creates an OCSP service on the client side. An iRule is then added to the interception rule VIP that inserts the FQDN into the forged server certificate that should resolve to the destination IP address defined in the setting, which is itself a separate virtual server with OCSP profile configuration. The applied SSL settings must also contain a valid server side OCSP configuration in order to report back actual revocation state to the OCSP profile.

    - **FQDN**: Enter an FQDN that will be injected into the authorityInfoAccess (AIA) field of the forged server certificate. This should resolve to the virtual server address listed below.

    - **Source**: Enter a source address filter here, as required. Otherwise leave as 0.0.0.0%0/0 to allow access from any source.

    - **Destination Address/Mask**: Enter the destination address and mask here that will match the FQDN value in the forged AIA value.

    - **Port**: Enter the destination port here for the OCSP service. This will almost always be unencrypted HTTP on port 80.

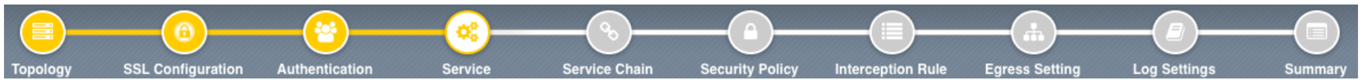
    - **VLANs**: Select the client facing VLAN.

    - **SSL Configuration**: Select the SSL configuration previously created for this topology workflow.

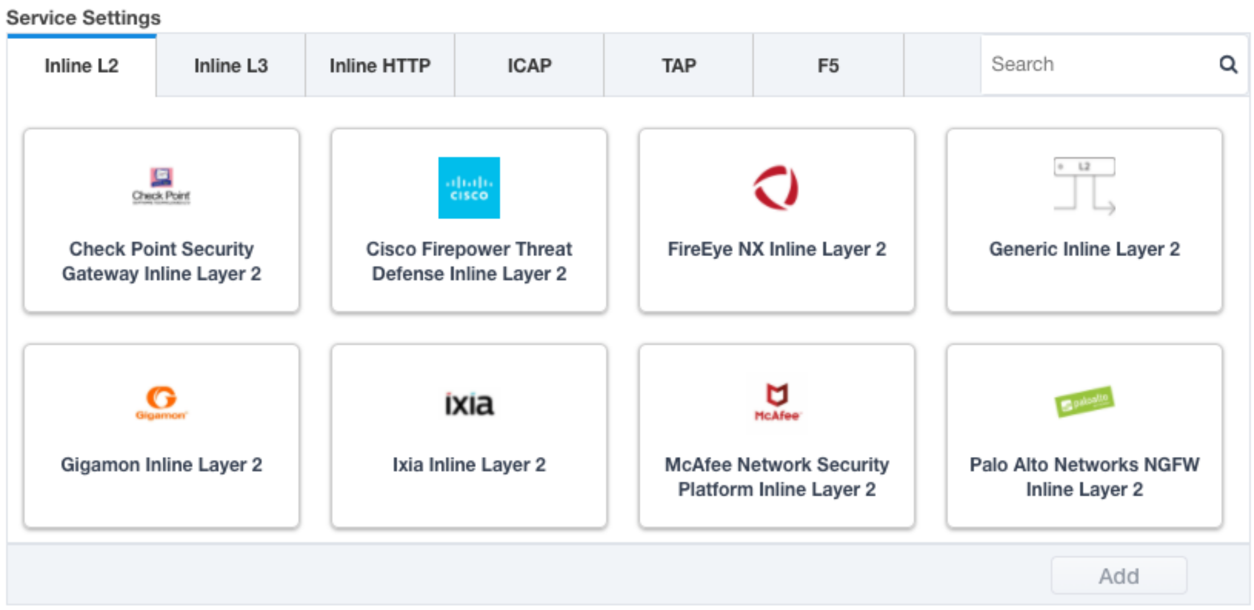
    - **OCSP Profile**: Select **Create New** or **Use Existing** as required.

- **Max Age in Seconds**: Enter a value in seconds for max age of the OCSP response.
- **Nonce**: Enable or disable OCSP response nonce, as required.
- **[Advanced] Client TCP Profile**: Select an alternate client-side TCP profile, as required.
- **[Advanced] Server TCP Profile**: Select an alternate server-side TCP profile, as required.
- **[Advanced] HTTP Profile**: Select an alternate HTTP profile, as required.

- Click Save & Next.



- **Services List** – The Services List page is used to define security services that attach to SSLO. The SSLO Guided Configuration now includes a tabbed services catalog that contains common product integrations. Beneath each of these catalog options is one of the five basic service types. The service catalog also provides "generic" security services. Depending on screen resolution, it may be necessary to scroll down to see additional services.


Service Settings
Inline L2
Inline L3
Inline HTTP
ICAP
TAP
F5
Search
Check Point Security Gateway Inline Layer 2
Cisco Firepower Threat Defense Inline Layer 2
FireEye NX Inline Layer 2
Generic Inline Layer 2
Gigamon Inline Layer 2
Ixia Inline Layer 2
McAfee Network Security Platform Inline Layer 2
Palo Alto Networks NGFW Inline Layer 2
Add

This lab will create one of each type of security service. Click Add Service, then either select a service from the catalog and click Add, or simply double-click the service to go to its configuration page.

- **Inline layer 2 service** – Select the FireEye NX Inline Layer 2 service from the catalog and click Add, or simply double-click the FireEye NX Inline Layer 2 service, or any other Inline Layer 2 service in the catalog.
  - **Name**: Provide a unique name to this service (example "FEYE").

- **Network Configuration** – Paths define the network interfaces that take inspectable traffic to the inline service and receive traffic from the service. Click Add.
    - **Ratio**: Inline security services are natively load balanced, so this setting defines a ratio, if any for the load balanced pool members. Enter 1.
    - **From BIGIP VLAN**: This is the interface taking traffic to the inline service. Select the Create New option, enter a unique name (ex. FEYE_in), select the F5 interface connecting to the inbound side of the service, and add a VLAN tag value if required. For this lab, select interface 1.4.
    - **To BIGIP VLAN**: This is the interface receiving traffic from the inline service. Select the Create New option, enter a unique name (ex. FEYE_out), select the F5 interface connecting to the outbound side of the service, and add a VLAN tag value if required. For this lab, select interface 1.5.
    - Click Done.
- **Device Monitor**: Security service definitions can now specify custom monitors. For this lab, leave it set to the default /Common/gateway_icmp.
- **Service Action Down**: SSLO also natively monitors the load balanced pool of security devices, and if all pool members fail, can actively bypass this service (**Ignore**), or stop all traffic (**Reset**, **Drop**). For this lab, leave it set to Ignore.
- **[Advanced] Internal IP Offset**: This setting specifies an internal IP Offset to generate the Internal IPv4 and IPv6 addresses. For High Availability (HA) configurations, the Internal IP Offset index numbers must be the same on all HA devices (Active and Standby). For scaled standalone configurations where non-HA standalone SSLO appliances are scaled behind a separate load balancer, the Internal IP Offset index numbers must be different across all devices.
- **Enable Port Remap**: This setting allows SSLO to remap the port of HTTPS traffic flowing across this service. This is advantageous when a security service defines port 443 traffic as encrypted HTTPS and natively ignores it. By remapping HTTPS traffic to, say, port 8080, the security service will inspect the traffic. For this lab, enable (check) this option and enter a port value (ex. 8080).
- **iRules**: SSLO now allows for the insertion of additional iRule logic at different points. An iRule defined at the service only affects traffic flowing across this service. It is important to understand, however, that these iRules must not be used to control traffic flow (ex. pools, nodes, virtual servers, etc.), but rather should be used to view/modify application layer protocol traffic. For example, an iRule assigned here could be used to view and modify HTTP traffic flowing to/from the service. Additional iRules are not required, however, so leave this empty.
- Click Save.

- **Inline layer 3 service** – Select the Generic Inline Layer 3 service from the catalog and click Add, or simply double-click the Generic Inline Layer 3 service.

    - **Name**: Provide a unique name to this service (example "IPS").

    - **IP Family**: This setting defines the IP family used with this layer 3 service. Leave it set to IPv4.

    - **Auto Manage Addresses**: When enabled the Auto Manage Addresses setting provides a set of unique, non-overlapping, non-routable IP addresses to be used by the security service. If disabled, the To and From IP addresses must be configured manually. It is recommended to leave this option enabled (checked).

      In environments where SSLO is introduced to existing security devices, it is a natural tendency to not want to have to move these devices. And while SSLO certainly allows it, by not moving the security devices into SSLO-protected enclaves, customers run the risk of exposing sensitive decrypted traffic, unintentionally, to other devices that may be connected to these existing networks. It is therefore highly recommended, and a security best practice, to remove SSLO-integrated security devices from existing networks and place them entirely within the isolated enclave created and maintained by SSLO.

    - **To Service Configuration**: The "To Service" defines the network connectivity from SSLO to the inline security device.
        - **To Service**: With the Auto Manage Addresses option enabled, this IP address will be pre-defined, therefore the inbound side of the service must match this IP subnet. With the Auto Manage Addresses option disabled, the IP address must be defined manually. For this lab, leave the 198.19.64.7/25 address intact.

        - **VLAN**: Select the Create New option, provide a unique name (ex. IPS_in), select the F5 interface connecting to the inbound side of the service, and add a VLAN tag value if required. For this lab, select interface 1.3 and VLAN tag 60.

    - **Service Down Action**: SSLO also natively monitors the load balanced pool of security devices, and if all pool members fail, can actively bypass this service (**Ignore**), or stop all traffic (**Reset**, **Drop**). For this lab, leave it set to Ignore.

    - **Security Devices - L3 Devices**: This defines the inbound-side IP address of the inline layer 3 service, used for routing traffic to this device. Multiple load balanced IP addresses can be defined here. Click Add, enter 198.19.64.30, then click Done.

    - **Device Monitor**: Security service definitions can now specify custom monitors. For this lab, leave it set to the default /Common/gateway_icmp.

    - **From Service Configuration**: The "From Service" defines the network connectivity from the inline security device to SSLO.

        - **From Service**: With the Auto Manage Addresses option enabled, this IP address will be pre-defined, therefore the outbound side of the service must match this IP subnet. With the Auto Manage Addresses option disabled, the IP address must be defined manually. For this lab, leave the 198.19.64.245/25 address intact.

        - **VLAN**: Select the Create New option, provide a unique name (ex. IPS_out), select the F5 interface connecting to the outbound side of the service, and add a VLAN tag value if required. For this lab, select interface 1.3 and VLAN tag 70.

    - **Enable Port Remap**: This setting allows SSLO to remap the port of HTTPS traffic flowing across this service. This is advantageous when a security service defines port 443 traffic as encrypted HTTPS and natively ignores it. By remapping HTTPS traffic to, say, port 8181, the security service will inspect the traffic. For this lab, enable (check) this option and enter a port value (ex. 8181).

    - **Manage SNAT Settings**: SSLO now defines an option to enable SNAT (source NAT) across an inline layer 3/HTTP service. The primary use case for this is horizontal SSLO scaling, where independent SSLO devices are scaled behind a separate load balancer but share the same inline layer 3/HTTP services. As these devices must route back to SSLO, there are now multiple SSLO devices to route back to. SNAT allows the layer 3/HTTP device to know which SSLO sent the packets for proper routing. SSLO scaling also requires that the Auto Manage option be disabled, to provide separate address spaces on each SSLO. For this, leave it set to None.

    - **iRules**: SSLO now allows for the insertion of additional iRule logic at different points. An iRule defined at the service only affects traffic flowing across this service. It is important to understand, however, that these iRules must not be used to control traffic flow (ex. pools, nodes, virtual servers, etc.), but rather should be used to view/modify application layer protocol traffic. For example, an iRule assigned here could be used to view and modify HTTP traffic flowing to/from the service. Additional iRules are not required, however, so leave this empty.

    - Click Save.

- **Inline HTTP service** – An inline HTTP service is defined as an explicit or transparent proxy for HTTP (web) traffic. Select the Cisco WSA HTTP Proxy service from the catalog and click Add, or simply double-click the Cisco WSA HTTP Proxy service, or any other HTTP Proxy service in the catalog.

    - **Name**: Provide a unique name to this service (example "Proxy").

    - **IP Family**: This setting defines the IP family used with this layer 3 service. Leave it set to IPv4.

    - **Auto Manage Addresses**: When enabled the Auto Manage Addresses setting provides a set of unique, non-overlapping, non-routable IP addresses to be used by the security service. If disabled, the To and From IP addresses must be configured manually. It is recommended to leave this option enabled (checked).

In environments where SSLO is introduced to existing security devices, it is a natural tendency to not want to have to move these devices. And while SSLO certainly allows it, by not moving the security devices into SSLO-protected enclaves, customers run the risk of exposing sensitive decrypted traffic, unintentionally, to other devices that may be connected to these existing networks. It is therefore highly recommended, and a security best practice, to remove SSLO-integrated security devices from existing networks and place them entirely within the isolated enclave created and maintained by SSLO.

- **Proxy Type**: This defines the proxy mode that the inline HTTP service is in. For this lab, set this option to Explicit.

- **To Service Configuration**: The "To Service" defines the network connectivity from SSLO to the inline security device.

    - **To Service**: With the Auto Manage Addresses option enabled, this IP address will be pre-defined, therefore the inbound side of the service must match this IP subnet. With the Auto Manage Addresses option disabled, the IP address must be defined manually. For this lab, leave the 198.19.96.7/25 address intact.

    - **VLAN**: Select the Create New option, provide a unique name (ex. Proxy_in), select the F5 interface connecting to the inbound side of the service, and add a VLAN tag value if required. For this lab, select interface 1.3 and VLAN tag 30.

- **Service Down Action**: SSLO also natively monitors the load balanced pool of security devices, and if all pool members fail, can actively bypass this service (**Ignore**), or stop all traffic (**Reset**, **Drop**). For this lab, leave it set to Ignore.

- **Security Devices: HTTP Proxy Devices** – This defines the inbound-side IP address of the inline HTTP service, used for passing traffic to this device. Multiple load balanced IP addresses can be defined here. For a transparent proxy HTTP service, only an IP address is required. For an explicit proxy HTTP service, the IP address and listening port is required. Click Add, enter 198.19.96.30 for the IP Address, and 3128 for the Port, then click Done.

- **Device Monitor**: Security service definitions can now specify custom monitors. For this lab, leave it set to the default /Common/gateway_icmp.

- **From Service Configuration**: The "From Service" defines the network connectivity from the inline security device to SSLO.

    - **From Service**: With the Auto Manage Addresses option enabled, this IP address will be pre-defined, therefore the outbound side of the service must match this IP subnet. With the Auto Manage Addresses option disabled, the IP address must be defined manually. For this lab, leave the 198.19.96.245/25 address intact.

    - **VLAN**: Select the Create New option, provide a unique name (ex. Proxy_out), select the F5 interface connecting to the outbound side of the service, and add a VLAN tag value if required. For this lab, select interface 1.3 and VLAN tag 40.

- **Manage SNAT Settings**: SSLO now defines an option to enable SNAT (source NAT) across an inline layer 3/HTTP service. The primary use case for this is horizontal SSLO scaling, where independent SSLO devices are scaled behind a separate load balancer but share the same inline layer 3/HTTP services. As these devices must route back to SSLO, there are now multiple SSLO devices to route back to. SNAT allows the layer 3/HTTP device to know which SSLO sent the packets for proper routing. SSLO scaling also requires that the Auto Manage option be disabled, to provide separate address spaces on each SSLO. For this, leave it set to None.

  - **Authentication Offload**: When an Access authentication profile is attached to an explicit forward proxy topology, this option will present the authenticated username value to the service as an X-Authenticated-User HTTP header. For this lab, leave it disabled (unchecked).

  - **iRules**: SSLO now allows for the insertion of additional iRule logic at different points. An iRule defined at the service only affects traffic flowing across this service. It is important to understand, however, that these iRules must not be used to control traffic flow (ex. pools, nodes, virtual servers, etc.), but rather should be used to view/modify application layer protocol traffic. For example, an iRule assigned here could be used to view and modify HTTP traffic flowing to/from the service. Additional iRules are not required, however, so leave this empty.

  - Click Save.

- **ICAP service** – An ICAP service is an RFC 3507-defined service that provides some set of services over the ICAP protocol. Select the Digital Guardian DLP ICAP service from the catalog and click Add, or simply double-click the Digital Guardian DLP ICAP service, or any other ICAP service in the catalog.
  - **Name**: Provide a unique name to this service (example "DLP").

  - **IP Family**: This setting defines the IP family used with this layer 3 service. Leave it set to IPv4.

  - **ICAP Devices**: This defines the IP address of the ICAP service, used for passing traffic to this device. Multiple load balanced IP addresses can be defined here. Click Add, enter 198.19.97.50 for the IP Address, and 1344 for the Port, and then click Done.

  - **ICAP Headers**: Select either **Default** or **Custom** to specify additional ICAP headers. To add custom headers, select Custom, otherwise leave as Default.

  - **OneConnect**: The F5 OneConnect profile improves performance by reusing TCP connections to ICAP servers to process multiple transactions. If the ICAP servers do not support multiple ICAP transactions per TCP connection, do not enable this option. For this lab, leave the OneConnect setting enabled.

  - **Request URI Path**: This is the RFC 3507-defined URI request path to the ICAP service. Each ICAP security vendor will differ with respect to request and response URIs, and preview length, so it is important to review the vendor's documentation. In this lab, enter /avscan.

  - **Response URI Path**: This is the RFC 3507-defined URI response path to the ICAP service. Each ICAP security vendor will differ with respect to request and response URIs, and preview length, so it is important to review the vendor's documentation. In this lab, enter /avscan.

    - **Preview Max Length(bytes)**: This defines the maximum length of the ICAP preview. Each ICAP security vendor will differ with respect to request and response URIs, and preview length, so it is important to review the vendor's documentation. A zero-length preview length implies that data will be streamed to the ICAP service, similar to an HTTP 100/Expect process, while any positive integer preview length defines the amount of data (in bytes) that are transmitted first, before streaming the remaining content. The ICAP service in this lab environment does not support a complete stream, so requires a modest amount of initial preview. In this lab, enter 524288.

    - **Service Down Action**: SSLO also natively monitors the load balanced pool of security devices, and if all pool members fail, can actively bypass this service (**Ignore**), or stop all traffic (**Reset**, **Drop**). For this lab, leave it set to Ignore.

    - **HTTP Version**: This defines whether SSLO sends HTTP/1.1 or HTTP/1.0 requests to the ICAP service.

    - **ICAP Policy**: An ICAP policy is a pre-defined LTM CPM policy that can be configured to control access to the ICAP service based on attributes of the HTTP request or response. ICAP processing is enabled by default, so an ICAP CPM policy can be used to disable the request and/or response ADAPT profiles.

    - Click Save.

- **TAP service** – A TAP service is a passive device that simply receives a copy of traffic. Select the Cisco Firepower Threat Defense TAP service from the catalog and click Add, or simply Double-click the Cisco Firepower Threat Defense TAP service, or any other TAP service in the catalog.

    - **Name**: Provide a unique name to this service (example "TAP").

    - **Mac Address**: For a tap service that is not directly connected to the F5, enter the device's MAC address. For a tap service that is directly connected to the F5, the MAC address does not matter and can be arbitrarily defined. For this lab, enter 12:12:12:12:12:12.

    - **VLAN**: This defines the interface connecting the F5 to the TAP service. Click Create New and provide a unique name (ex. TAP_in).

    - **Tag**: Enter a VLAN tag value, as required.

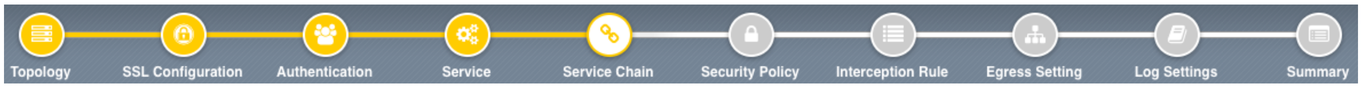
    - **Interface**: Select the 1.6 interface.

    - **Enable Port Remap**: This setting allows SSLO to remap the port of HTTPS traffic flowing to this service. For this lab, leave the option disabled (unchecked).

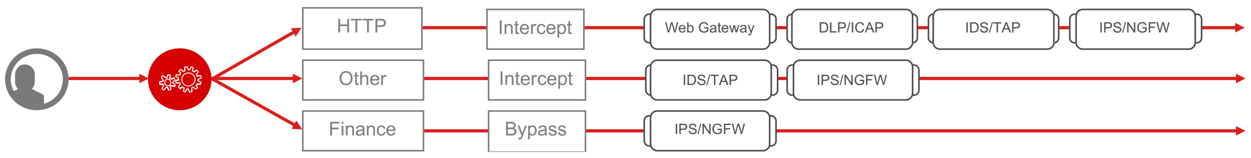
    - Click Save.

- **F5 service** – This tab represents the F5 services that can be attached to the service chain. In 9.0 this includes the F5 Secure Web Gateway (SWGaaS). Please see the section on configuring SWG for more detailed information.

- Click Save & Next.



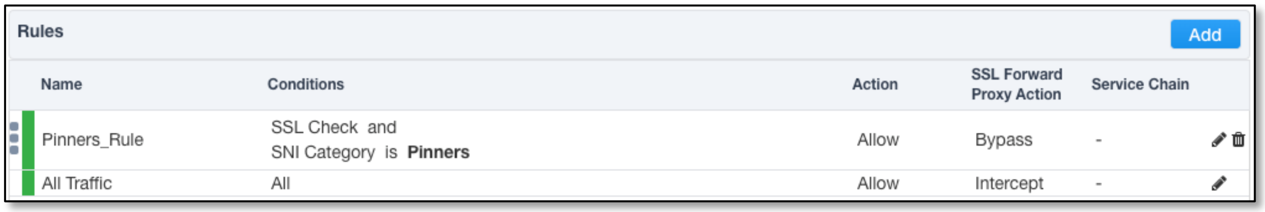
- **Service Chain List** – Service chains are arbitrarily-ordered lists of security devices. Based on environmental requirements, different service chains may contain different re-used sets of services, and different types of traffic can be assigned to different service chains. For example, HTTP traffic may need to go through all of the security services, while non-HTTP traffic goes through a subset, and traffic destined to a financial service URL can bypass decryption and still flow through a smaller set of security services.



  - Click Add to create a new service chain containing all of the security services.
    - **Name**: Provide a unique name to this service (ex. "all_services").
    - **Services**: Select any number of desired service and move them into the **Selected Service Chain Order** column, optionally also ordering them as required. In this lab, select all of the services.
    - Click Save.
  - Click Add to create a new service chain for just the L2 (ex. FireEye) and TAP services.
    - **Name**: Provide a unique name to this service (ex. "L2_services").
    - **Services**: Select the inline layer 2 (ex. FireEye) and TAP services.
    - Click Save.
  - Click Save & Next.

- **Security Policy** – security policies are the set of rules that govern how traffic is processed in SSLO. The "actions" a rule can take include,
  - Whether or not to allow the traffic (allow/block)
  - Whether or not to decrypt the traffic (intercept/bypass)
  - Which service chain (if any) to pass the traffic through

  The SSLO Guided Configuration presents an intuitive rule-based, drag-and-drop user interface for the definition of security policies.



Rules

Add

Name	Conditions	Action	SSL Forward Proxy Action	Service Chain
Pinners_Rule	SSL Check and SNI Category is **Pinners**	Allow	Bypass	-
All Traffic	All	Allow	Intercept	-

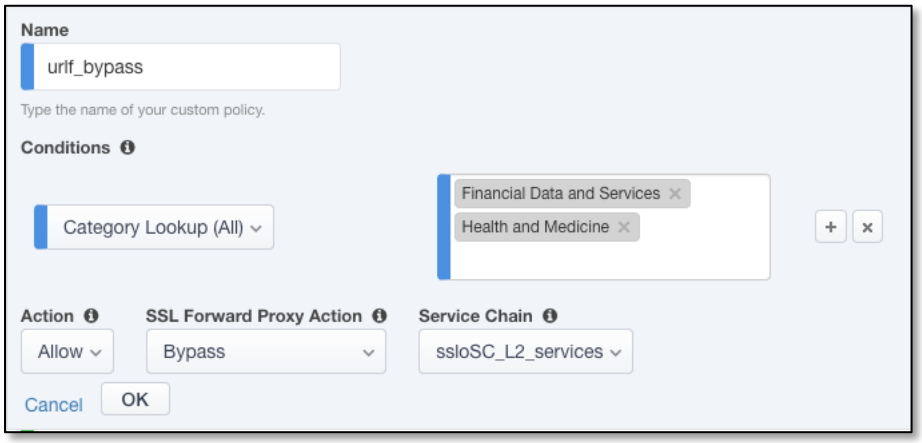
In the background, SSLO maintains these security policies as visual per-request policies. If traffic processing is required that exceeds the capabilities of the rule-based user interface, the underlying per-request policy can be managed directly.

Note that once the per-request policy is manipulated, the rules-based interface can no longer be used.

For the lab, create an additional rule to bypass SSL for “Financial Data and Services” and “Health and Medicine” URL categories.

- Click Add to create a new rule.
  - **Name**: Provide a unique name for the rule (ex. “urlf_bypass”).
  - **Conditions**
    - **Category Lookup (All)**: Add Financial Data and Services and Health and Medicine.

      The Category Lookup (All) condition provides categorization for TLS SNI, HTTP Connect and HTTP Host information.
  - **Action**: Select Allow.
  - **SSL Forward Proxy Action**: Select Bypass.
  - **Service Chain**: Select the L2/TAP service chain.
  - Click OK.



Name: urlf_bypass

Type the name of your custom policy.

Conditions: Category Lookup (All) — Financial Data and Services, Health and Medicine

Action: Allow

SSL Forward Proxy Action: Bypass

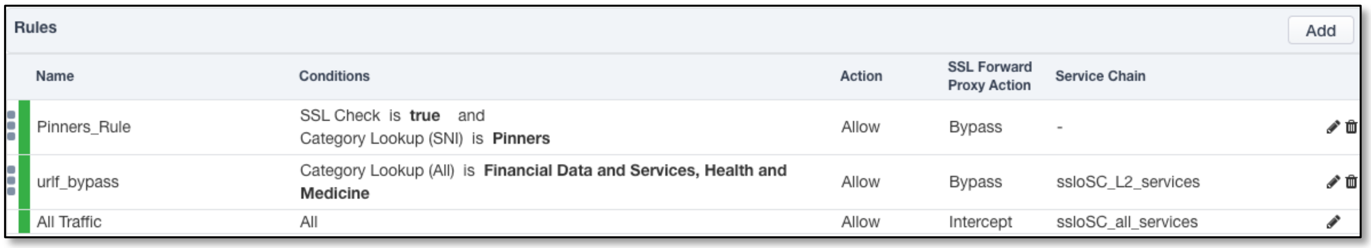
Service Chain: ssloSC_L2_services

Notice in the list of rules that the **All Traffic** rule intercepts but does not send traffic to a service chain. For the lab, edit this rule to send all intercepted traffic to a service chain.

- Click the pencil icon to edit this rule.

- Service Chain – Select the service chain containing all of the services.

- Click OK.



Rules

Name	Conditions	Action	SSL Forward Proxy Action	Service Chain
Pinners_Rule	SSL Check is **true** and Category Lookup (SNI) is **Pinners**	Allow	Bypass	-
urlf_bypass	Category Lookup (All) is **Financial Data and Services, Health and Medicine**	Allow	Bypass	ssloSC_L2_services
All Traffic	All	Allow	Intercept	ssloSC_all_services

The SSL Orchestrator 9.0 security policy contains new traffic conditions:

- **Server Certificate (Issuer DN)**: This condition evaluates the issuer distinguishedName (DN) value from the original certificate.

- **Server Certificate (SANs)**: This condition evaluates the subjectAlternativeName (SAN) values from the original certificate.

- **Server Certificate (Subject DN)**: This condition evaluates the subject distinguishedName (DN) value from the original certificate.

- **Server Name (TLS ClientHello)**: This condition evaluates the Server Name Indication (SNI) extension value in the client-side ClientHello message from the client. This condition is most useful to create a pure TLS bypass condition for specific types of traffic that may break under normal TLS bypass conditions, like mutual TLS. As security policy rules are nested, it is also important that this rule be placed above other rules that evaluate other SSL and traffic properties, like URL category matches.

The SSL Orchestrator 9.0 security policy also contains a new "**abort**" action on matched traffic flows. The abort ending will generate a different result based on where in the flow the abort is triggered:

- TCP reset for TCP
- ICMP Port Unreachable for UDP
- TLS Alert (Handshake Failure) for TLS Client Hello

- **Server Certificate Status Check**: This option inserts additional security policy logic to validate the remote server certificate and return a blocking page to the user if the certificate is untrusted or expired. One or both of the Certificate Response options on the SSL Configuration page (Expire Certificate Response and Untrusted Certificate Response) must be set to 'ignore'. SSLO will "mask" the server certificate's attributes in order to present a blocking page with a valid forged certificate. For this lab, either option (enabled or disabled) is acceptable.

- **Proxy Connect**: This option is only available for the outbound explicit proxy topology and defines an upstream proxy chain. With this setting enabled, SSLO can egress to an upstream (external) explicit proxy gateway. An example of this might be a cloud gateway solution. Select a pool that points to the upstream explicit proxy, and optionally any (HTTP Basic) credentials needed to access this proxy. For this lab, leave this setting disabled.

Click Save & Next.

- **Interception Rule** – Interception rules are based on the selected topology and define the "listeners", analogous to LTM virtual servers, that accept and process different types of traffic (ex. TCP, UDP, other). The resulting LTM virtual servers will bind the SSL settings, VLANs, IPs, and security policies created in the topology workflow.
    - **Source Address**: The source address field provides a filter for incoming traffic based on source address and/or source subnet. It is usually appropriate to leave the default 0.0.0.0%0/0 setting applied to allow traffic from all addresses to be processed. As of version 9.0, a source address list is also configurable here. Address Lists are configured under Shared Objects in the BIG-IP UI.
    - **Destination Address/Mask**: The destination address/mask field provides a filter for incoming traffic based on destination address and/or destination subnet. As this is a transparent forward proxy configuration, it is appropriate to leave the default 0.0.0.0%0/0 setting applied to allow all outbound traffic to be processed. As of version 9.0, a destination address list is also configurable here. Address Lists are configured under Shared Objects in the BIG-IP UI.
    - **[Custom/Advanced] Port**: The port field provides a filter for incoming traffic based on destination port.
    - **Ingress Network – VLANs**: This defines the VLANs through which traffic will enter. For a transparent forward proxy topology, this would be a client-side VLAN. Select client-vlan.
    - **Protocol Settings – Verified Accept**: When enabled, the system verifies that the pool member is available to accept the connection by sending the server a SYN before responding to the client's SYN with a SYN-ACK packet. SSL Orchestrator topologies are built on a standard virtual server type (with Verified Accept disabled). This could cause an issue in the scenario where an upstream firewall might block a specific IP and/or port, but the SSL Orchestrator closer to the client allows the connection (only to be blocked later at the firewall). Enabling Verified Accept allows the F5 to test for a valid server-side connection before completing the client-side handshake. Enable or disable this setting as required.
    - **[Custom/Advanced] Client TCP Profile**: This setting allows for selection of a custom client-side TCP profile.
    - **[Custom/Advanced] Server TCP Profile**: This setting allows for selection of a custom server-side TCP profile.
    - **[Custom/Advanced] SSL Configurations**: This setting allows for selection of a custom SSL configuration.

- **[Custom] L7 Profile Type**: This setting allows for selection of a custom layer 7 profile type.

- **[Custom] L7 Profile**: This setting allows for selection of a custom layer 7 profile, depending on the L7 Profile Type selection.

- **Security Policy Settings – Access Profile**: The Access Profile selection is exposed for both explicit and transparent forward proxy topology deployments. In transparent forward proxy mode, this allows selection of an access policy to support captive portal authentication (covered later in this guide). For this lab, leave this setting as default.

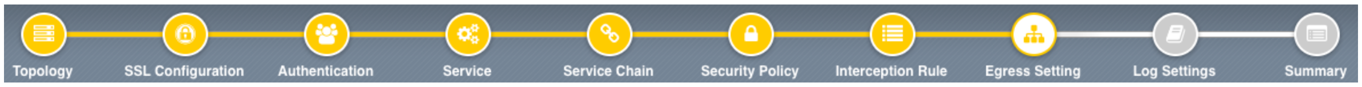
- **Security Policy Settings – Access Profile Scope**: Profile scope generally applies to authentication. When a transparent proxy captive portal authentication profile is assigned to the topology, the Named scope allows authentication information from the captive portal authentication to be available here. Without authentication, the scope should be left as Public.

- **Authentication**: This setting defines additional authentication schemes to be attached. In the initial 9.0 release, this simply contains the OCSP Responder configuration created on the Authentication page of the workflow.

- **L7 Interception Rules – Protocols**: FTP and email protocol traffic are all "server-speaks-first" protocols, and therefore SSLO must process these separately from typical client-speaks-first protocols like HTTP. This selection enables processing of each of these protocols, which create separate port-based listeners for each. As required, selectively enable the additional protocols that need to be decrypted and inspected through SSLO.

- **Pool**: This setting would not generally be used in a forward proxy scenario, as it enables address translation (destination NAT) to a set of pool members.

- Click Save & Next.



- Egress Setting – Traffic egress settings are now defined per-topology and manage both the gateway route and outbound SNAT settings.

  - **Manage SNAT Settings**: Enables per-topology instance SNAT settings. For this lab, select Auto Map.

  - **Gateways**: Enables per-topology instance gateway routing. Options are to use the system default route, to use an existing gateway pool, or to create a new gateway. For this lab, select Create New.

  - **IPv4 Outbound Gateways**: When creating a new gateway, this section provides the ratio and gateway address settings.

  - **Ratio**: Multiple gateway IP addresses are load balanced in an LTM pool, and the ratio setting allows SSLO to proportion traffic to the gateway members, as required. A ratio on 1 for all members evenly distributes the load across them. For this lab, select 1.

- **Address**: This is the next hop gateway IP address. For this lab, enter 10.1.20.1.

- Click Save & Next.

- **Log Settings** – log settings are defined per-topology. In environments where multiple topologies are deployed, this can help to streamline troubleshooting by reducing debug logging to the affected topology. Multiple discreet logging options are available:

  - **Per-Request Policy**: provides log settings for security policy processing. In Debug mode, this log facility produces an enormous amount of traffic, so it is recommended to only set Debug mode for troubleshooting. Otherwise, the most appropriate setting is Error to only log error conditions.

  - **FTP**: specifically logs error conditions for the built-in FTP listener when FTP is selected among the additional protocols in the Interception Rule configuration. The most appropriate setting is Error to only log error conditions.

  - **IMAP**: specifically logs error conditions for the built-in IMAP listener when IMAP is selected among the additional protocols in the Interception Rule configuration. The most appropriate setting is Error to only log error conditions.

  - **POP3**: specifically logs error conditions for the built-in POP3 listener when POP3 is selected among the additional protocols in the Interception Rule configuration. The most appropriate setting is Error to only log error conditions.

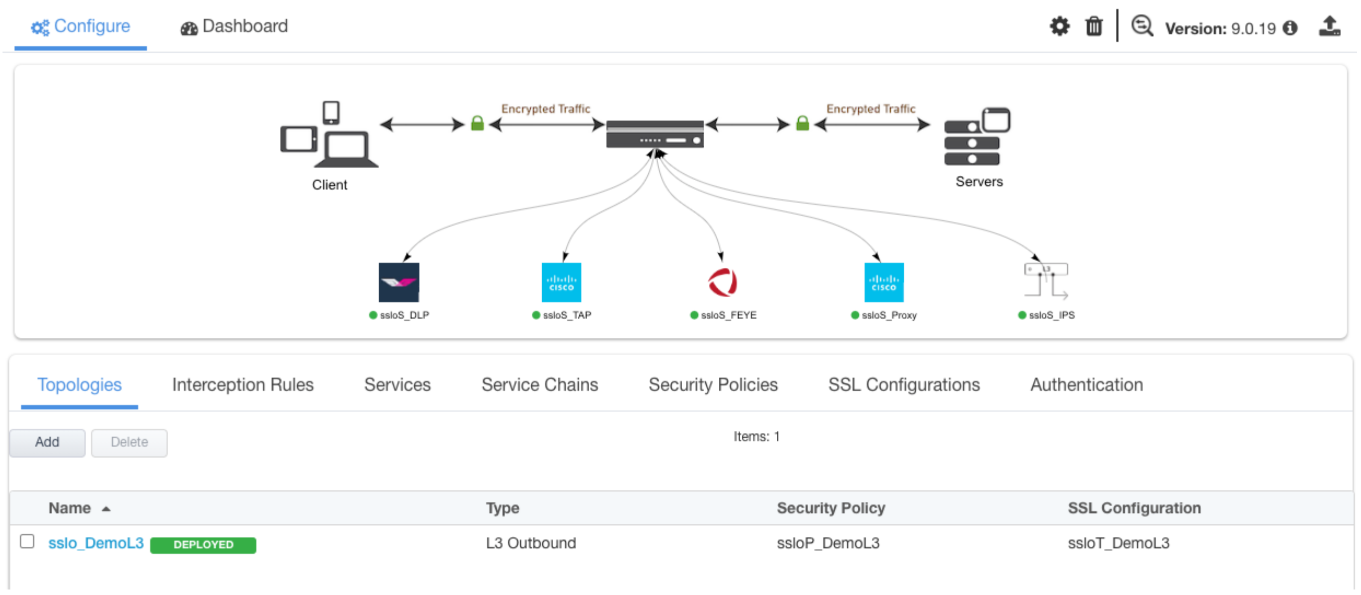
  - **SMTP**: specifically logs error conditions for the built-in SMTP listener when SMTP is selected among the additional protocols in the Interception Rule configuration. The most appropriate setting is Error to only log error conditions.

  - **SSL Orchestrator Generic**: provides log settings for generic SSLO processing. If Per-Request Policy logging is set to Error, and SSL Orchestrator Generic is set to Information, only the SSLO packet summary will be logged. Otherwise, the most appropriate setting is Error to only log error conditions.

  Click Save & Next.

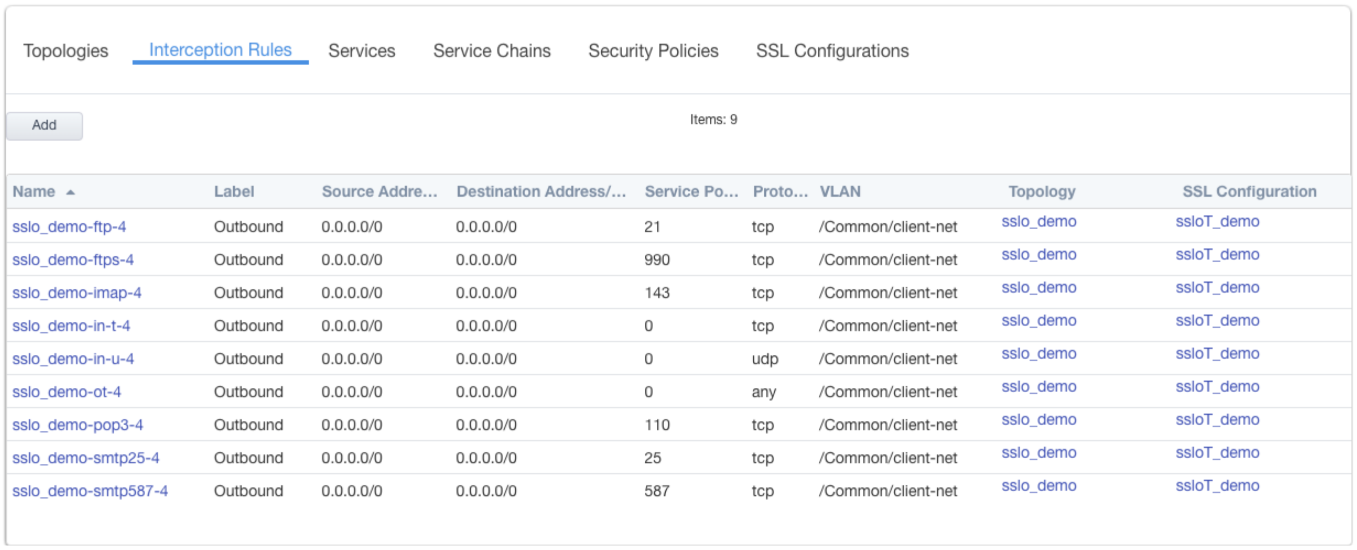
- **Summary** – the summary page presents an expandable list of all of the workflow-configured objects. To expand the details for any given setting, click the corresponding arrow icon on the far right. To edit any given setting, click the corresponding pencil icon. Clicking the pencil icon will send the workflow back to the selected settings page. When satisfied with the defined settings, click Deploy.

Upon successfully deploying the configuration, SSL Orchestrator will now display a **Configure** view.



Name	Type	Security Policy	SSL Configuration
sslo_DemoL3 DEPLOYED	L3 Outbound	ssloP_DemoL3	ssloT_DemoL3

The **Interception Rules** tab shows the listeners that were created per the selected topology.



Name	Label	Source Addre...	Destination Address/...	Service Po...	Proto...	VLAN	Topology	SSL Configuration
sslo_demo-ftp-4	Outbound	0.0.0.0/0	0.0.0.0/0	21	tcp	/Common/client-net	sslo_demo	ssloT_demo
sslo_demo-ftps-4	Outbound	0.0.0.0/0	0.0.0.0/0	990	tcp	/Common/client-net	sslo_demo	ssloT_demo
sslo_demo-imap-4	Outbound	0.0.0.0/0	0.0.0.0/0	143	tcp	/Common/client-net	sslo_demo	ssloT_demo
sslo_demo-in-t-4	Outbound	0.0.0.0/0	0.0.0.0/0	0	tcp	/Common/client-net	sslo_demo	ssloT_demo
sslo_demo-in-u-4	Outbound	0.0.0.0/0	0.0.0.0/0	0	udp	/Common/client-net	sslo_demo	ssloT_demo
sslo_demo-ot-4	Outbound	0.0.0.0/0	0.0.0.0/0	0	any	/Common/client-net	sslo_demo	ssloT_demo
sslo_demo-pop3-4	Outbound	0.0.0.0/0	0.0.0.0/0	110	tcp	/Common/client-net	sslo_demo	ssloT_demo
sslo_demo-smtp25-4	Outbound	0.0.0.0/0	0.0.0.0/0	25	tcp	/Common/client-net	sslo_demo	ssloT_demo
sslo_demo-smtp587-4	Outbound	0.0.0.0/0	0.0.0.0/0	587	tcp	/Common/client-net	sslo_demo	ssloT_demo

In the above,

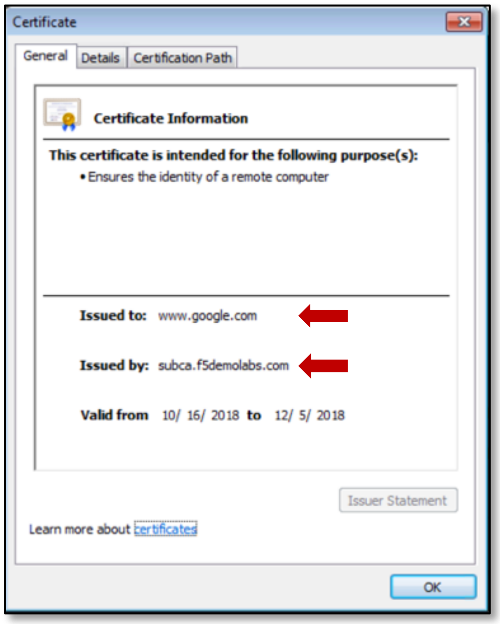
- The **-in-t-4** listener defines normal TCP IPv4 traffic.
- The **-in-u-4** listener defines normal UDP IPv4 traffic.
- The **-ot-4** listener defines normal non-TCP/non-UDP IPv4 traffic.
- The **-ftp**, **-ftps**, **-pop3**, **-smtp25** and **-smtp587** listeners create paths for each respective protocol.

This completes the configuration of SSL Orchestrator as a transparent forward proxy. At this point an internal client should be able to browse out to external (Internet) resources, and decrypted traffic will flow across the security services.

Step 4: Test the solution

To test the deployed solution, use the following options:

- **Server certificate test**
  Open a browser on the client system and navigate to any remote HTTPS site, for example, https://www.google.com. Once the site opens in the browser, check the server certificate of the site and verify that it has been issued by the local CA configured in SSLO. This confirms that the SSL forward proxy functionality enabled by SSL Orchestrator is working correctly.



This certificate is intended for the following purpose(s):
• Ensures the identity of a remote computer

Issued to: www.google.com

Issued by: subca.f5demolabs.com

Valid from 10/ 16/ 2018 to 12/ 5/ 2018

Learn more about certificates

- **Decrypted traffic analysis on the F5**
  Perform a tcpdump on the F5 system to observe the decrypted clear text traffic. This confirms SSL interception by SSLO.

```
tcpdump –lnni [interface or VLAN name] -Xs0
```

  As a function of adding a new service, the UI requires a name for each (source and destination) network. SSL Orchestrator will then create separate source and destination VLANs for inline security devices, and those VLANs will be encapsulated within separate application service paths. For example, given an inline layer 2 service named "FEYE" with its "From BIGIP VLAN" named "**FEYE_in**", and its "To BIGIP VLAN" named "**FEYE_out**", its corresponding BIG-IP VLANs would be accessible via the following syntax:

  ssloN_ *+ [network name] +* ***.app/ssloN_*** *+ [network name]*

  Example:
  ssloN_FEYE_in.app/ssloN_FEYE_in
  ssloN_FEYE_out.app/ssloN_FEYE_out

  A tcpdump on the source side VLAN of this FireEye service would therefore look like this:

```
tcpdump -lnni ssloN_FEYE_in.app/ssloN_FEYE_in -Xs0
```

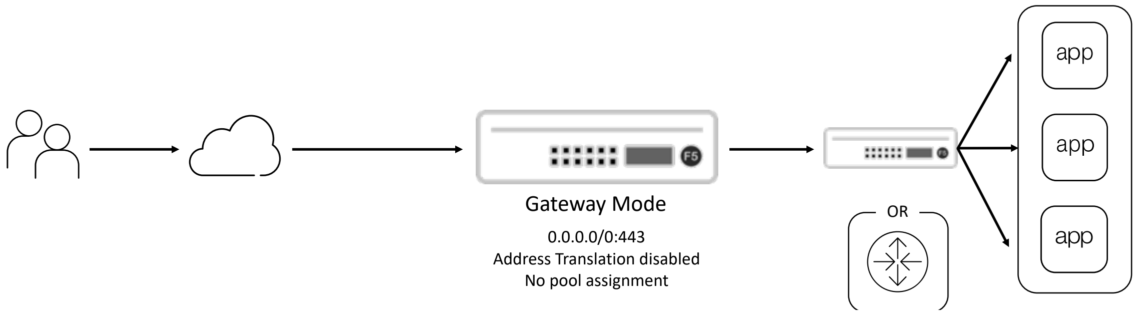
  The security service VLANs and their corresponding application services are all visible from the BIG-IP UI under Network -> VLANs.

- **Decrypted traffic analysis on the security services**
  Depending on the type of security service, it may easier to log into the console shell and run a similar tcpdump capture on the inbound or outbound interface, to tail its capture logs, or to log into its management UI and capture analytics. A tcpdump capture usually requires root or sudo access.

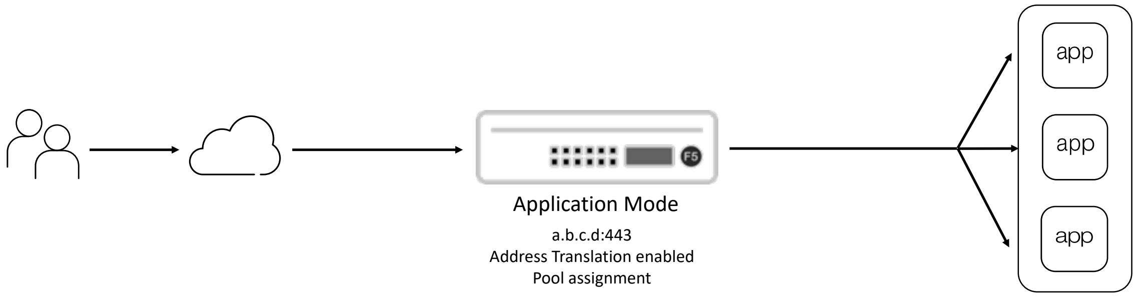
```
tcpdump -lnni [interface] -Xs0
```

LAB 2 – CREATE AN INBOUND LAYER 3 TOPOLOGY

The SSL Orchestrator inbound L3 topology generally describes two slightly different use cases for inbound traffic. The default use case is a "gateway" mode. This is where SSLO sits in front of a separate ADC/load balancer or routed path. The primary attributes of the gateway mode are a wildcard (0.0.0.0/0) listener, no pool assignment and disabled address translation (destination NAT). Destination addresses from the external client therefore point to a location *behind* SSLO.



The alternate use case is an "application" mode, where the client's destination address terminates at the F5. The primary attributes of the application mode are an IP-specific listener address, backend server pool assignment, and enabled address translation (destination NAT). An SSLO inbound topology in application mode behaves very much like a typical F5 BIG-IP reverse proxy virtual server, except that SSLO manages the complete configuration workflow, including mandatory re-encryption to the backend services.



The following lab instructions provide for both SSLO inbound L3 use cases.

This lab will consist of an abbreviated set of steps, as some of the objects created in Lab 1 (services and service chains) will be fully re-usable here. If any of these objects have not been created, please review Lab 1 for more detailed configuration instructions.

Step 1: Review the lab diagram and map out the services and endpoints

Specifically, note that in this lab there is a web server on the internal network (the client's network in this case) that external users want to get to. An external client desktop exists on the external/outbound network, that accesses these resources through SSLO.

- The external client is attached to a *10.1.20.0/24* network and is assigned the IP *10.1.20.50*. This network is attached to the BIG-IP 1.2 interface.

- The web server is a Docker container on the Ubuntu 18.04 server configured with Apache2, and listens on four addresses:
    - `192.168.100.10`
    - `192.168.100.11`
    - `192.168.100.12`
    - `192.168.100.13`

- In lieu of a separate DNS server in the lab, the external client has static /etc/hosts entries that map the above addresses to the following URLs, respectively:

192.168.100.10	test0.f5labs.com
192.168.100.11	test1.f5labs.com
192.168.100.12	test2.f5labs.com
192.168.100.13	test3.f5labs.com
10.1.20.120	www.f5labs.com

  The first four addresses point the client *through* the SSLO inbound topology in gateway mode. The last address points the client *to* the SSLO inbound topology in application mode.

- The external client is also configured with a static route that points 10.1.10.0/24 destinations *through* the F5 outbound-side self-IP. This is for lab testing in lieu of a separate router.

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    0      0        0 ens6
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 ens5
10.1.10.0       0.0.0.0         255.255.255.0   UG    0      0        0 ens6
10.1.20.0       0.0.0.0         255.255.255.0   U     0      0        0 ens7
192.168.100.1   10.1.20.10      255.255.255.0   UG    0      0        0 ens7
```

- A wildcard (*.f5labs.com) server certificate and private key have been installed on the SSL Orchestrator.

The external client has two options for accessing the internal websites: via wildcard (0.0.0.0/0) gateway, and direct IP listener. The lab will explore both options below.

Note: SSL Orchestrator sends all traffic through an inline layer 3 or HTTP device in the same direction – entering through the inbound interface. While still technically possible via customization, SSLO does not support dual routing through an inline layer 3/HTTP service.

SSL Orchestrator 9.1 introduces gateway mode support, where:

- **Gateway** mode works like a router, where the topology uses a network address to process incoming connections.

- **Application** mode works like a traditional LTM virtual server. It creates a topology listening on a specific IP:port and processes incoming connections for this IP.

The following steps will follow an application mode deployment first, and then create a separate gateway mode deployment.

Step 2: Configure an L3 inbound SSLO deployment – Application Mode

In this scenario, an SSLO L3 inbound listener is configured in application mode. It will listen on a dedicated single IP:port. The **[Advanced]** options below are available when “Show Advanced Settings” is enabled (top right). Follow the L3 Inbound topology workflow to build this solution. In the SSL Orchestrator dashboard view, select the Topologies tab (bottom) and click Add.

- **Configuration review and prerequisites** – Take a moment to review the topology options and workflow configuration, then click Next.

- **Topology Properties**
    - **Name**: Provide some name (ex. “demoInApp”)
    - **Protocol**: TCP
    - **IP Family**: IPv4
    - **Topology**: Select L3 Inbound
    - **Mode**: Select Application
    - Click Save & Next

- **SSL Configuration** – An inbound topology requires different SSL settings.
    - Click Show Advanced Setting
    - **Client-side SSL**
        - **SNI Server Name (FQDN)**: Optionally specify the SNI Server Name (FQDN). If you attach multiple SSL profiles, one of them must be the default SNI: select the Default SNI check box. Only one default SNI is allowed. New in 9.1, multiple SSL profiles can be attached to a single topology, where the correct SSL profile is selected dynamically based on the incoming Client Hello Server Name Indication (SNI) extension. In a reverse proxy scenario, this feature would typically be used to select a different server certificate based on incoming SNI.
        - **[Advanced] Processing Options**: SSLO now provides TLS 1.3 support for outbound topologies. TLS 1.3 configuration is described in a later lab, so for now leave this as is.
        - **Cipher Type**: Cipher String
        - **Cipher String**: DEFAULT
        - **Certificate Key Chain**: The certificate key chain represents the certificate and private key of an endpoint server instance (the target of a remote client’s request). In a gateway-mode configuration, this would typically be a wildcard of Subject Alternative Name (SAN) certificate in the event the SSLO inbound listener was intended to service multiple sites. In

this lab a wildcard certificate has been provided. Select the pencil icon to edit, then select the wildcard.f5labs.com certificate and private key and click Done.

SSL Settings minimally require RSA-based template and CA certificates but can also support Elliptic Curve (ECDSA) certificates.

- **Server-side SSL**
    - **[Advanced] Processing Options**: SSLO now provides TLS 1.3 support for outbound topologies. TLS 1.3 configuration is described in a later lab, so for now leave this as is.
    - **Cipher Type**: Cipher String
    - **Cipher String**: DEFAULT
    - **Trusted Certificate Authority**: As an inbound solution, the server-side SSL would be pointing to internal servers. While definitely possible to perform validation against internal server certificates, it is likely less important to do so. Leave this setting as is.
    - **Expire Certificate Response**: Assuming no internal certificate validation is needed, the default **drop** setting will cause the connection to fail, so leave this at Ignore.
    - **Untrusted Certificate Authority**: Assuming no internal certificate validation is needed, the default **drop** setting will cause the connection to fail, so leave this at Ignore.
    - **[Advanced] OCSP Certificate Validator**: Assuming no internal certificate validation is needed, any OCSP configuration will cause the connection to fail, so leave this as is.
    - **[Advanced] CRL Certificate Validator**: Assuming no internal certificate validation is needed, any CRL configuration will cause the connection to fail, so leave this as is.
- Click Save & Next.

- **Services List** – The same services can be leveraged here, so simply click Save & Next.
- **Service Chain List** – The same service chains can be leveraged here, so simply click Save & Next.
- **Security Policy** – The security policy requirements are specific to each organization, though an inbound security policy would likely be less complex than an outbound policy. Minimally assign a service chain to the existing **All Traffic** rule and click Save & Next.

Note: SSL Orchestrator sends all traffic through an inline layer 3 or HTTP device in the same direction – entering through the inbound interface. While still technically possible via customization, SSLO does not support dual routing through an inline layer 3/HTTP service.

- **Interception Rule**
    - **Application mode** – Interception rule listening on a dedicated IP, port 443, with any server certificate. Clients route to SSLO.
        - Show Advanced Setting
        - **Source Address**: 0.0.0.0%0/0
        - **Destination Address/Mask**: 10.1.20.120/32
        - **Port**: 443
        - **Security Policy Settings – Access Profile**: The Access Profile selection is exposed for inbound topologies to allow for insertion of per-session access and authentication processing. For this lab, leave this setting as default.
        - **Ingress Network – VLANs**: Select the outbound-vlan (this is the server-side VLAN).
        - **Protocol Settings – Client TCP Profile**: Allows setting a custom client-side TCP profile.
        - **Protocol Settings – Server TCP Profile**: Allows setting a custom server-side TCP profile.
        - **Protocol Settings – SSL Configuration**: Allows setting a custom SSL setting.
        - **L7 Profile Type**: This setting enables or disables HTTP processing.
        - **L7 Profile**: If the above option is set to HTTP, this option selects a specific HTTP profile.
        - **Pool**: Select the web-https-pool (pre-created server pool).

  Click Save & Next

- **Egress Settings** – Traffic egress settings are now defined per-topology and manage both the gateway route and outbound SNAT settings.
    - **Manage SNAT Settings**: Enables per-topology instance SNAT settings. For this lab, select Auto Map.

  Click Save & Next

- **Log Settings** – The Log Settings page presents a per-topology configuration for log settings. For this lab, these settings can be left as default. Click Save & Next

- **Summary** – The summary page presents an expandable list of all of the workflow-configured objects. To expand the details for any given setting, click the corresponding arrow icon on the far right. To edit any given setting, click the corresponding pencil icon. Clicking the pencil icon will send the workflow back to the selected settings page. When satisfied with the defined settings, click Deploy.

Step 3: Configure an L3 inbound SSLO deployment – Gateway Mode

In this scenario, an SSLO L3 inbound listener is configured in gateway mode. It will listen on a range of IP addresses. The **[Advanced]** options below are available when "Show Advanced Settings" is enabled (top right). Follow the L3 Inbound topology workflow to build this solution. In the SSL Orchestrator dashboard view, select the Topologies tab (bottom) and click Add.

- **Configuration review and prerequisites** – Take a moment to review the topology options and workflow configuration, then click Next.

- **Topology Properties**
  - **Name**: Provide some name (ex. "demoInGW")
  - **Protocol**: TCP
  - **IP Family**: IPv4
  - **Topology**: Select L3 Inbound
  - **Mode**: Select Gateway
  - Click Save & Next

- **SSL Configuration** – An inbound topology requires different SSL settings.
  - Click Show Advanced Setting
  - **Client-side SSL**
    - **SNI Server Name (FQDN)**: Optionally specify the SNI Server Name (FQDN). If you attach multiple SSL profiles, one of them must be the default SNI: select the Default SNI check box. Only one default SNI is allowed. New in 9.1, multiple SSL profiles can be attached to a single topology, where the correct SSL profile is selected dynamically based on the incoming Client Hello Server Name Indication (SNI) extension. In a reverse proxy scenario, this feature would typically be used to select a different server certificate based on incoming SNI.
    - **[Advanced] Processing Options**: SSLO now provides TLS 1.3 support for outbound topologies. TLS 1.3 configuration is described in a later lab, so for now leave this as is.
    - **Cipher Type**: Cipher String
    - **Cipher String**: DEFAULT
    - **Certificate Key Chain**: The certificate key chain represents the certificate and private key of an endpoint server instance (the target of a remote client's request). In a gateway-mode configuration, this would typically be a wildcard of Subject Alternative Name (SAN) certificate in the event the SSLO inbound listener was intended to service multiple sites. In

this lab a wildcard certificate has been provided. Select the pencil icon to edit, then select the wildcard.f5labs.com certificate and private key and click Done.

SSL Settings minimally require RSA-based template and CA certificates but can also support Elliptic Curve (ECDSA) certificates.

- **Server-side SSL**
  - **[Advanced] Processing Options**: SSLO now provides TLS 1.3 support for outbound topologies. TLS 1.3 configuration is described in a later lab, so for now leave this as is.
  - **Cipher Type**: Cipher String
  - **Cipher String**: DEFAULT
  - **Trusted Certificate Authority**: As an inbound solution, the server-side SSL would be pointing to internal servers. While definitely possible to perform validation against internal server certificates, it is likely less important to do so. Leave this setting as is.
  - **Expire Certificate Response**: Assuming no internal certificate validation is needed, the default **drop** setting will cause the connection to fail, so leave this at Ignore.
  - **Untrusted Certificate Authority**: Assuming no internal certificate validation is needed, the default **drop** setting will cause the connection to fail, so leave this at Ignore.
  - **[Advanced] OCSP Certificate Validator**: Assuming no internal certificate validation is needed, any OCSP configuration will cause the connection to fail, so leave this as is.
  - **[Advanced] CRL Certificate Validator**: Assuming no internal certificate validation is needed, any CRL configuration will cause the connection to fail, so leave this as is.
- Click Save & Next.

- **Services List** – The same services can be leveraged here, so simply click Save & Next.
- **Service Chain List** – The same service chains can be leveraged here, so simply click Save & Next.
- **Security Policy** – The security policy requirements are specific to each organization, though an inbound security policy would likely be less complex than an outbound policy. Minimally assign a service chain to the existing **All Traffic** rule and click Save & Next.

Note: SSL Orchestrator sends all traffic through an inline layer 3 or HTTP device in the same direction – entering through the inbound interface. While still technically possible via customization, SSLO does not support dual routing through an inline layer 3/HTTP service.

- **Interception Rule**
    - **Gateway mode** – Interception rule listening on a range of IP addresses, port 443, with any server certificate. Clients route to SSLO.
        - Show Advanced Setting
        - **Source Address**: 0.0.0.0%0/0
        - **Destination Address/Mask**: 0.0.0.0%0/0
        - **Port**: 443
        - **Security Policy Settings – Access Profile**: The Access Profile selection is exposed for inbound topologies to allow for insertion of per-session access and authentication processing. For this lab, leave this setting as default.
        - **Ingress Network – VLANs**: Select the outbound-vlan (this is the server-side VLAN).
        - **Protocol Settings – Client TCP Profile**: Allows setting a custom client-side TCP profile.
        - **Protocol Settings – Server TCP Profile**: Allows setting a custom server-side TCP profile.
        - **Protocol Settings – SSL Configuration**: Allows setting a custom SSL setting.
        - **L7 Profile Type**: This setting enables or disables HTTP processing.
        - **L7 Profile**: If the above option is set to HTTP, this option selects a specific HTTP profile.
        - **Pool**: Select the web-https-pool (pre-created server pool).

  Click Save & Next

- **Egress Settings** – Traffic egress settings are now defined per-topology and manage both the gateway route and outbound SNAT settings.
    - **Manage SNAT Settings**: Enables per-topology instance SNAT settings. For this lab, select Auto Map.
    - **Gateways**: Enables per-topology instance gateway routing. Options are to use the system default route, to use an existing gateway pool, or to create a new gateway. For this lab, select Default Route.

  Click Save & Next

- **Log Settings** – The Log Settings page presents a per-topology configuration for log settings. For this lab, these settings can be left as default. Click Save & Next

- **Summary** – The summary page presents an expandable list of all of the workflow-configured objects. To expand the details for any given setting, click the corresponding arrow icon on the far right. To edit any given setting, click the corresponding pencil icon. Clicking the pencil icon will send the workflow back to the selected settings page. When satisfied with the defined settings, click Deploy.

Step 4: Testing

For gateway mode testing, the lab's inbound desktop client includes static Hosts entries that match the real IPs of the internal web server,

• test0.f5labs.com	=	192.168.100.10
• test1.f5labs.com	=	192.168.100.11
• test2.f5labs.com	=	192.168.100.12
• test3.f5labs.com	=	192.168.100.13

and a static persistent route that points 10.1.10.0/24 traffic to the BIG-IP outbound (external) VLAN self-IP (10.1.20.100). To test the **gateway mode** use case, open a browser or cURL to one of the above URLs.

```
curl -vk https://test0.f5labs.com
```

For application mode testing, create a static Hosts entry in /etc/hosts for,

- www.f5labs.com = 10.1.20.120

To test an **application mode** use case, open a browser or cURL to the provided URL.

```
curl -vk https://www.f5labs.com
```

LAB 3 – CREATE AN EXPLICIT FORWARD PROXY TOPOLOGY

SSL Orchestrator creates discreet, non-overlapping interception rules (listeners) based on the selected topology. For example, the explicit forward proxy workflow minimally creates an explicit proxy listener and relying transparent proxy listener attached to the explicit proxy tunnel. If a separate transparent proxy workflow was created, the resulting listener would not conflict with or overlap the existing transparent proxy listener. Therefore, assuming a transparent forward proxy already exists from Lab 1, the following workflow will create a separate set of non-overlapping listeners to satisfy an explicit forward proxy use case.

This lab will consist of an abbreviated set of steps, as all of the objects created in Lab 1 (SSL settings, services, service chains and security policies) will be fully re-usable here. If any of these objects have not been created, please review Lab 1 for more detailed configuration instructions.

Step 1: Review the lab diagram and map out the services and endpoints

Review the same step in Lab 1 for more details. This lab uses the exact same environment, so SSL settings, services, service chains and security policy will be re-used.

Step 2: Configure an explicit proxy SSLO deployment through Guided Configuration

- **Configuration review and prerequisites** – Take a moment to review the topology options and workflow configuration, then click Next.
- **Topology Properties**
  - **Name**: Provide some name (ex. "demoEP")
  - **Protocol**: TCP
  - **IP Family**: IPv4
  - **Topology**: Select L3 Explicit Proxy
  - Click Save & Next
- **SSL Configurations** – The existing outbound SSL settings from Lab 1 can be re-used here.
  - **SSL Profile**: Use Existing, select existing outbound SSL settings.
  - Click Save & Next

Whenever repurposing a topology setting, a warning will appear, "There are other configuration items that are referencing this item. Editing this item will affect the referencing ones mentioned below". Click OK to acknowledge.

- **Authentication List** – In 9.0 a new Authentication List workflow exists to create authentication mechanisms for a topology. In this initial release, the Authentication List simply contains an OCSP Responder configuration. The list of authentication mechanisms will grow in subsequent versions.

- **Services List** – There are no new services to create.
  - Click Save & Next
- **Service Chain List** – There are no new service chains to create.
  - Click Save & Next
- **Security Policy** – The existing outbound Security Policy from Lab 1 can be re-used here.
  - **Type**: Use Existing, select existing outbound SSL settings.
  - **Proxy Connect**: This option is only available for the outbound explicit proxy topology and defines an upstream proxy chain. With this setting enabled, SSLO can egress to an upstream (external) explicit proxy gateway. An example of this might be a cloud gateway solution. Select a pool that points to the upstream explicit proxy, and optionally any (HTTP Basic) credentials needed to access this proxy. For this lab, leave this setting disabled.
  - Click Save & Next
- **Interception Rule** – An explicit proxy requires a unique IP address and port listener.
  - **IPv4 Address**: 10.1.10.150
  - **Port**: 3128
  - **Access Profile**: If enabling explicit proxy authentication, select an existing SWG-Explicit access profile here.
  - **DNS Resolver**: In SSL Orchestrator 9.0 you can define a separate DNS resolver configuration for each explicit proxy topology. If you haven't already created a unique DNS resolver, select the "/Common/ssloGS_global.app/ssloGS-net-resolver". If you modify the global DNS settings in SSL Orchestrator (detailed in step 3 below), it will update this configuration.
  - **Authentication - OCSP Responder**: If you created a client-side OCSP authentication profile earlier in the workflow, you can select that here.
  - **VLANs**: client-vlan
  - Click Save & Next
- **Egress Setting** – Traffic egress settings are now defined per-topology and manage both the gateway route and outbound SNAT settings.
  - **Manage SNAT Settings**: Enables per-topology instance SNAT settings. For this lab, select Auto Map.
  - **Gateways**: Enables per-topology instance gateway routing. Options are to use the system default route, to use an existing gateway pool, or to create a new gateway. For this lab, select Use Existing Gateway Pool, then select the "-ex-pool-4" gateway pool.
  - Click Save & Next

- **Log Settings** – Log settings are defined per-topology. In environments where multiple topologies are deployed, this can help to streamline troubleshooting by reducing debug logging to the affected topology.

- **Summary** – The summary page presents an expandable list of all of the workflow-configured objects. To expand the details for any given setting, click the corresponding arrow icon on the far right. To edit any given setting, click the corresponding pencil icon. Clicking the pencil icon will send the workflow back to the selected settings page. When satisfied with the defined settings, click Deploy.

Step 3: Add DNS and Logging settings

Minimally an explicit proxy requires DNS settings. To enable this for the L3 Explicit topology, in the SSLO UI click System Settings icon (gear) on the top right of the Configure view.

- **DNS Query Resolution**: Select Local Forwarding Nameserver.

- **Local Forwarding Nameserver(s)**: enter 10.1.20.1.

- **Gateway Configuration**: If using a remote DNS forwarder, that is, anything not on a local subnet to the F5 (ex. 8.8.8.8), a route must also be configured to allow SSLO to reach this remote DNS. If a system gateway route has already been defined, select Default Route here. Otherwise, select Create New and enter the local gateway route. In this lab that's 10.1.20.1.

- Click Deploy to commit the changes.

Testing – Configure the browser to use 10.1.10.150:3128 for explicit proxy access. An explicit proxy request can also be done using command-line cURL:

```
curl -vk --proxy 10.1.10.150:3128 https://www.example.com
```

LAB 4 – CREATE AN EXISTING APPLICATIONS TOPOLOGY

SSL Orchestrator defines an existing application as a typical reverse proxy LTM virtual server, performing its own SSL handling (SSL profiles) and traffic management (assigned pool). The Existing Application SSLO topology therefore only needs to create the components that this virtual server can consume, specifically the services, service chains, and security policy. The Existing Application SSLO workflow skips SSL management, interception rules and egress settings, and ultimately produces an SSLO-type per-request policy that can be attached to an existing LTM virtual server.

This lab will consist of an abbreviated set of steps, as all of the relevant objects created in Lab 1 (services, service chains and security policies) will be fully re-usable here. If any of these objects have not been created, please review Lab 1 for more detailed configuration instructions.

Step 1: Review the lab diagram and map out the services and endpoints

Review the same step in Lab 1 for more details. This lab uses the exact same environment, so SSL settings, services, service chains and security policy will be re-used.

Step 2: Create an LTM application

For the lab, create a simple LTM application,

- **Create a pool** – Use one (or multiple) of the internal webserver IPs and select port 80. This is different than the webserver pool used in the SSLO inbound topology lab. In that lab re-encryption to the backend applications is mandatory. In the Existing Application topology, the traffic management and SSL processing are processed by the LTM application virtual server, so SSL offload (no re-encryption) is supported.
    - 192.168.100.10:80
    - 192.168.100.11:80
    - 192.168.100.12:80
    - 192.168.100.13:80
- **Create a client SSL profile** – Use the wildcard.f5labs.com certificate and private key.
- **Create an LTM virtual server** – Use the following basic settings,
    - **Destination Address/Mask**: 10.1.20.125
    - **Service Port**: 443
    - **HTTP Profile**: http
    - **SSL Profile (Client)**: wildcard.f5labs.com SSL profile
    - **VLANs and Tunnels**: outbound-vlan VLAN
    - **Source Address Translation**: Auto Map
    - **Pool**: previously created pool

- **Test access to the LTM virtual server** – The webserver should be accessible via HTTPS request to the LTM virtual server. The inbound client has an /etc/hosts entry for "app.f5labs.com" that points to 10.1.20.125.

Step 3: Configure an Existing Application deployment through Guided Configuration

- **Configuration review and prerequisites** – Take a moment to review the topology options and workflow configuration, then click Next.
- **Topology Properties**
  - **Name**: Provide some name (ex. "existing_app_1")
  - **IP Family**: IPv4
  - **Topology**: Select Existing Application
  - Click Save & Next
- **Services List** – There are no new services to create.
  - Click Save & Next
- **Services Chain List** – There are no new service chains to create.
  - Click Save & Next
- **Security Policy** – The security policy requirements are specific to each organization, though an inbound security policy would likely be less complex than an outbound policy. Minimally assign a service chain to the existing **All Traffic** rule and click Save & Next.

Note: SSL Orchestrator sends all traffic through an inline layer 3 or HTTP device in the same direction – entering through the inbound interface. While still technically possible via customization, SSLO does not support dual routing through an inline layer 3/HTTP service.

- **Summary** – The summary page presents an expandable list of all of the workflow-configured objects. To expand the details for any given setting, click the corresponding arrow icon on the far right. To edit any given setting, click the corresponding pencil icon. Clicking the pencil icon will send the workflow back to the selected settings page.
  - When satisfied with the defined settings, click Deploy.

Step 4: Attach the SSLO objects to an existing LTM application

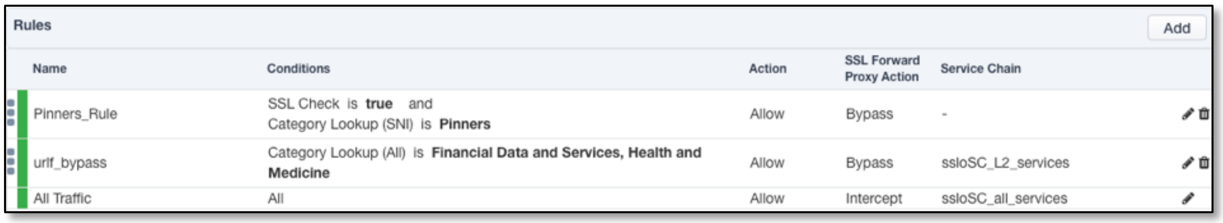
The Existing Application topology workflow produces a single SSLO per-request policy. To attach this to the LTM virtual server, edit the virtual server properties,

- **Access Policy (Access Profile**): Attach the single "ssloDefault_accessProfile".
- **Access Policy (Per-Request Policy)**: Attach the existing application per-request policy.

Test this configuration by browsing to the internal resource and verifying that decrypted traffic flows to the inline security services.

Lab 5 – Manage the SSL Orchestrator Security Policy

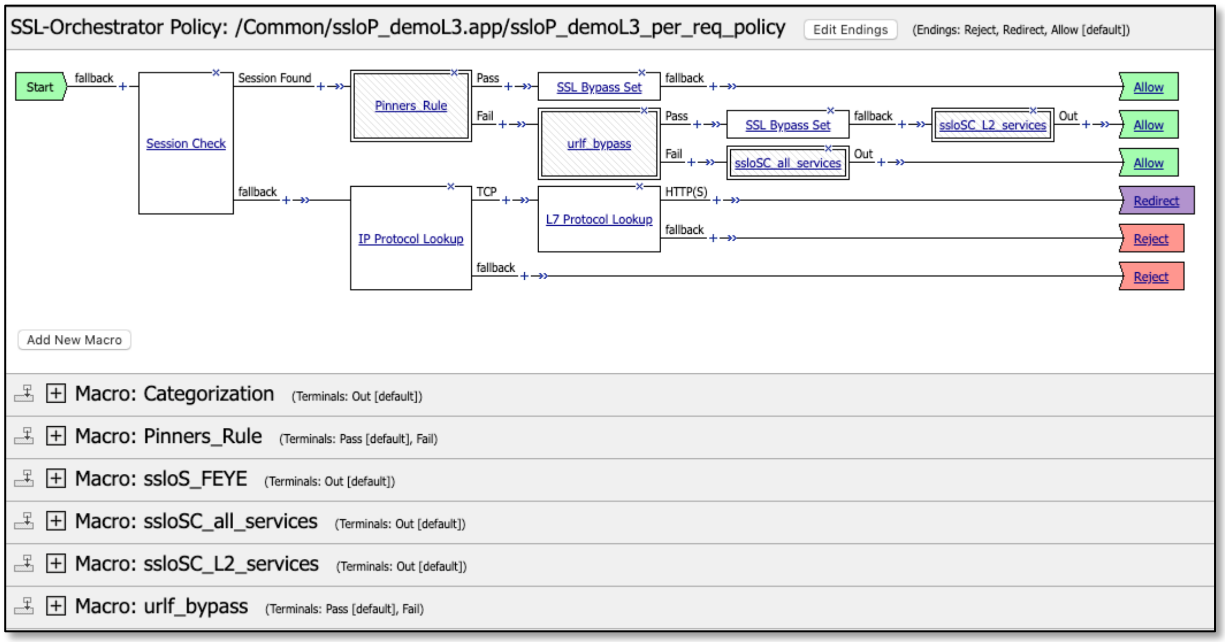
SSL Orchestrator provides a rich, interactive, rules-based security policy through the Guided Configuration.



Rules

Name	Conditions	Action	SSL Forward Proxy Action	Service Chain
Pinners_Rule	SSL Check is **true** and Category Lookup (SNI) is **Pinners**	Allow	Bypass	-
urlf_bypass	Category Lookup (All) is **Financial Data and Services, Health and Medicine**	Allow	Bypass	ssloSC_L2_services
All Traffic	All	Allow	Intercept	ssloSC_all_services

The security policy itself is a front-end to an access per-request engine that converts the rules into visual elements in this policy. Also note that the order of rules affects the order of events in the visual policy. Rules are read top-to-bottom and converted into corresponding visual agents nesting from left to right.



SSL-Orchestrator Policy: /Common/ssloP_demoL3.app/ssloP_demoL3_per_req_policy Edit Endings (Endings: Reject, Redirect, Allow [default])

Macro: Categorization (Terminals: Out [default])

Macro: Pinners_Rule (Terminals: Pass [default], Fail)

Macro: ssloS_FEYE (Terminals: Out [default])

Macro: ssloSC_all_services (Terminals: Out [default])

Macro: ssloSC_L2_services (Terminals: Out [default])

Macro: urlf_bypass (Terminals: Pass [default], Fail)

While security policy rules work well for most traffic processing scenarios, it may be necessary to go beyond their capabilities and manipulate the visual per-request policy directly.

Keep in mind, however, that the rules engine converts rules to visual elements in one direction only. It cannot convert visual elements back to rules, therefore once the visual per-request policy has been manipulated, the Guided Configuration security policies user interface will no longer be available.

This lab will explore some of the different options for manipulating SSLO security policies.

Step 1: Review and edit the existing security policy rules

In the SSLO dashboard view, navigate to the Security Policies tab and click on a security policy (Name). The Guided Configuration will present the rules engine previously seen as part of the topology workflow. New rules can be added, and existing rules edited. Notice also that the “All Traffic” rule is anchored to the security policy and cannot be moved or removed. This is the default action rule for the policy, similar to a default deny rule in a firewall policy.

By default, it Intercepts (decrypts) traffic, but does not send traffic to any service chain. This can be edited to Intercept, bypass or block (reject), and to send traffic to a service chain.

Additional rules can use **AND** (Match All) or **OR** (Match Any) logic to create complex decisions. Review the **Conditions** options to see the possibilities.

Step 2: Review and edit the visual per-request policy

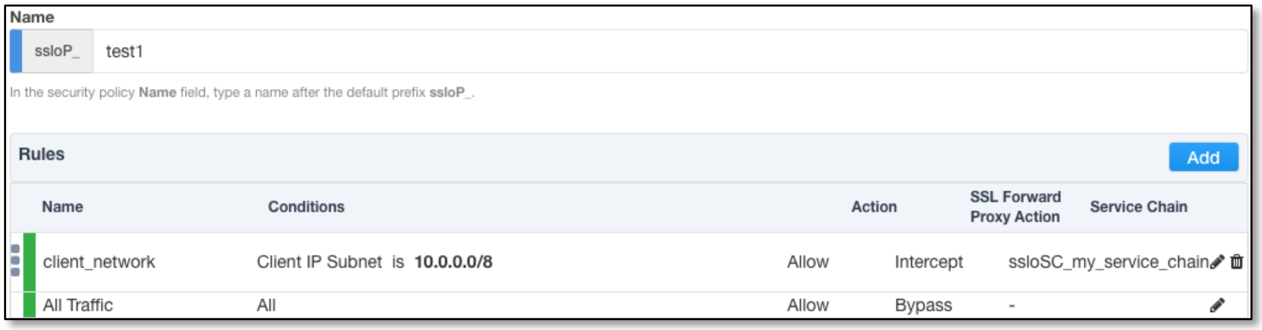
To view the underlying visual security policy, in the SSLO dashboard view, navigate to the Security Policies tab and click on a security policy (Per Request Policies). This will open a new tab with a view of the visual per-request policy. By default, the security policy is locked and prevents any changes to the visual per-request policy. To edit the visual policy, first unlock the policy in the SSLO dashboard, Security Policies tab.

Keep in mind, however, that the rules engine converts rules to visual elements in one direction only. It cannot convert visual elements back to rules, therefore once the visual per-request policy has been manipulated, the Guided Configuration security policies user interface will no longer be available.

Step 3: Practice creating Security Policies

The following are a few examples of security policy use cases,

- Create a new security policy that matches source addresses in the outbound desktop client's subnet, intercepts SSL, and sends to a service chain. All other traffic is bypassed with no service chain.


Name

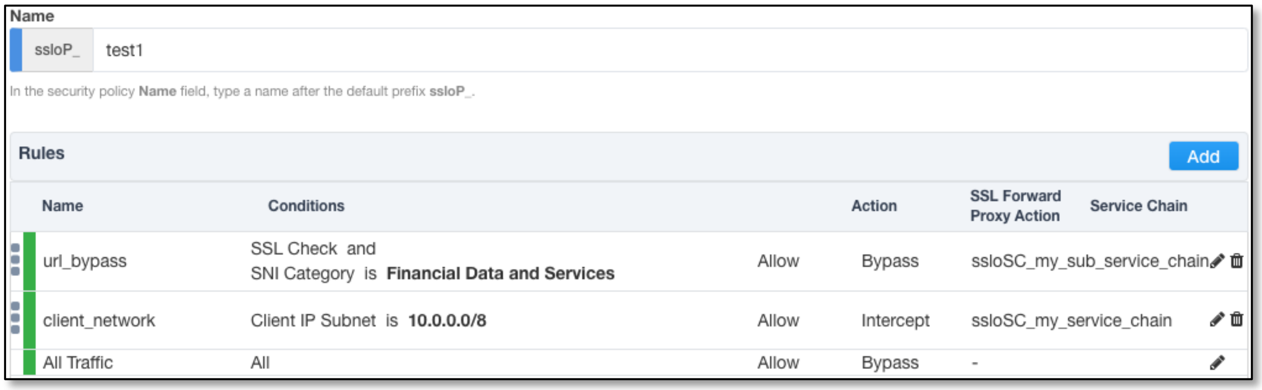
sslоP_ test1

In the security policy **Name** field, type a name after the default prefix **ssloP_**.

Rules

Name	Conditions	Action	SSL Forward Proxy Action	Service Chain
client_network	Client IP Subnet is **10.0.0.0/8**	Allow	Intercept	ssloSC_my_service_chain
All Traffic	All	Allow	Bypass	-

- Add a rule to the above security policy that matches a specific URL category, bypasses SSL and sends to a service chain. Move this rule to the top of the list.


Name

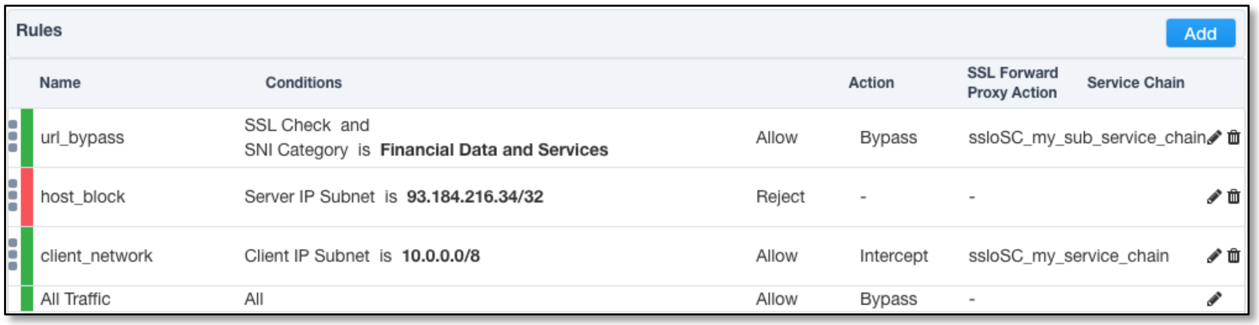
ssloP_ test1

In the security policy **Name** field, type a name after the default prefix **ssloP_**.

Rules

Name	Conditions	Action	SSL Forward Proxy Action	Service Chain
url_bypass	SSL Check and SNI Category is **Financial Data and Services**	Allow	Bypass	ssloSC_my_sub_service_chain
client_network	Client IP Subnet is **10.0.0.0/8**	Allow	Intercept	ssloSC_my_service_chain
All Traffic	All	Allow	Bypass	-

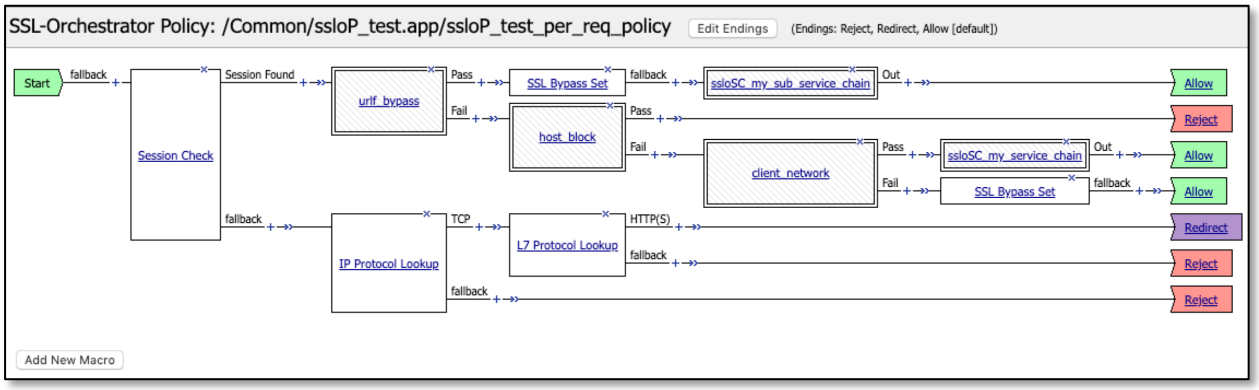
- Add a new rule to the above security policy that matches a specific destination IP and blocks this traffic. Move this rule below the URL category rule, but above the client network rule.



Rules

Name	Conditions	Action	SSL Forward Proxy Action	Service Chain
url_bypass	SSL Check and SNI Category is **Financial Data and Services**	Allow	Bypass	ssloSC_my_sub_service_chain
host_block	Server IP Subnet is **93.184.216.34/32**	Reject	-	-
client_network	Client IP Subnet is **10.0.0.0/8**	Allow	Intercept	ssloSC_my_service_chain
All Traffic	All	Allow	Bypass	-

- Click Deploy, then navigate to the **Security Policies** tab in the SSL Orchestrator UI. For the newly created security policy, click the link under the **Per Request Policies** header. This will open a new tab to the visual per-request policy.



Notice that the visual policy elements are nested in accordance with the ordered set of rules,

- If the URL category is "Financial Data and Services" (urlf_bypass), bypass SSL and send to a service chain.
- Otherwise, if the destination IP is 93.184.216.34/32 (host_block), reject the traffic.
- Otherwise, if the client IP matches 10.0.0.0/8 (client_network), send to a service chain (SSL interception implied).
- Otherwise, bypass SSL and do not send to a service chain.

Note that the **L7 Protocol Lookup** and **URL Match** options must assume that incoming traffic is either unencrypted or decrypted, therefore any rules that use these, and any rules after these cannot select to intercept or bypass the SSL.

Apply the new rule to an existing outbound topology and test that a) financial sites are bypassed, b) https://www.example.com is blocked, c) and all other client traffic flows through the defined service chain. View the APM log to follow the policy logic:

```
tail -f /var/log/apm |grep -E "Following|Executed"
```

LAB 6 – CREATE OUTBOUND CHANNELS FOR SERVICES

Security devices attached to SSL Orchestrator are opaque to the external environment. They are protected, isolated, and do not generally interact outside of internal connectivity with the F5 BIG-IP. While this design is intended to protect the security devices and the sensitive decrypted traffic flowing through them, it also presents a significant consideration.

Specifically, SSL Orchestrator dynamically steers traffic through security services by virtue of a signaling mechanism, such that only managed (signaled) traffic can flow through the service chain. Should a security device require its own connectivity to external resources, the device-initiated flows would be un-managed and thus, not allowed. There are scenarios, however, where a security device would still need to access external resources.

For example, an explicit proxy device would need access to DNS. Malware detection devices would need to be able to update signatures. And many security products require phone-home access to validate licensing. In these cases, it is necessary to create “service control channels”, pathways to allow device-initiated traffic to egress to the Internet.

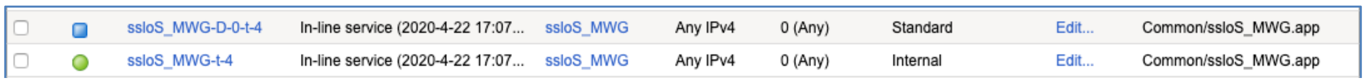
Note that service control channels are generally only required for inline L3 and HTTP-type services (however they could also be needed by inline L2 services).

Traffic originating from a service can technically pass through a separate SSLO inspection zone but ensure that this traffic does not flow through a service chain containing this service.

As of SSLO 6.0, SSLO-created VLANs are not selectable when creating new services. To create a service control channel for a security service, the original service definition must consume VLANs pre-created manually. If the service has already been created, it will be necessary to delete and re-create the service using the new manually created VLANs.

To better understand how service control channels work, it is necessary to first understand how traffic flows through an inline security service within the SSL Orchestrator service chain. Inline devices are inline because they represent a path through the device. Normally, that means traffic enters one interface and exits a second interface. A single interface can also be used if the layer 3/HTTP device supports 802.1Q VLAN tagging (separating a single interface into two logical VLANs and subnets) or, in rare cases can be deployed “one-arm” if the device source NATs on egress. When you create an inline service in the SSL Orchestrator UI, that service minimally creates two virtual servers:

- An “inbound” or “to-service” virtual server that sends traffic to the inline device
- An “outbound” or “from-service” virtual server that receives traffic back from the inline device



☐	■	ssloS_MWG-D-0-t-4	In-line service (2020-4-22 17:07...	ssloS_MWG	Any IPv4	0 (Any)	Standard	Edit...	Common/ssloS_MWG.app
☐	●	ssloS_MWG-t-4	In-line service (2020-4-22 17:07...	ssloS_MWG	Any IPv4	0 (Any)	Internal	Edit...	Common/ssloS_MWG.app

The above image represents an inline HTTP service from the perspective of LTM virtual servers:

- When instructed to do so by the active service chain, traffic flow attaches to the ssloS_MWG-t-4 virtual server which then load balances the traffic across the inline services. The -t-4 here represents a TCP IPv4 virtual server. There could also be TCP (-t) and UDP (-u), IPv4 (-4), and IPv6 (-6) combinations depending on configuration.

- An inline layer 3/HTTP service gateway routes back to the F5 BIG-IP, and its gateway is the VLAN attached to the ssloS_MWG-D-0-t-4 virtual server. Again, -t-4 represents a TCP IPv4 virtual server with -D meaning "destination". This virtual server is responsible for re-attaching the traffic to current flow context so that it can be attached to other devices in the service chain.

With the above flow in mind, it should be clear that the -D-0- virtual server consumes the service return data specifically because it listens on the VLAN attributed to "from-service" traffic. The -D-0- virtual server is a wildcard listener (0.0.0.0/0:0), meaning it listens for TCP IPv4 traffic on any source IP, any destination IP, and any destination port. So, to create a service control channel, you must create a new virtual server listening on this same "from-service" VLAN, but that is **more specific** than the wildcard -D-0- listener. Specificity in this case would be some combination of source IP, destination IP, destination port, and layer 4 protocol (TCP/UDP).

For example, to allow an inline proxy device to talk to an external DNS, you might create a service control channel virtual server with the following characteristics:

- Protocol: UDP
- Source IP: the service's from-service self IP
- Destination IP: 8.8.8.8
- Destination port: 53

It is almost always most useful, in this case, to a) define the service's IP in the source field, and b) disable source NAT on the device so that client-server traffic flowing through maintains the client's true address. In this case, anything on the device's IP would be device-initiated traffic. Some security products call this "SNAT", "source NAT", "secure NAT", or "IP spoofing". In any case, when possible, you should disable source NAT. Or if the term "IP spoofing" is used, you should enable that. Disabling source NAT also makes creating service control channels easier as you only need to define one to listen on the device's source IP, any destination IP, any port, and any protocol.

Step 1: Review the service's remote access requirements

For this lab, the inline proxy service simply needs external DNS access to 8.8.8.8 UDP.

Step 2: Create a service control channel virtual server

To create the service control channel from the above DNS example, navigate to **Local Traffic -> Virtual Servers** in the F5 BIG-IP UI. Any setting not specified below can be left as is.

- Virtual Server
    - **Name**: Provide some name (ex. "proxy_sc_dns")
    - **Type**: Select Performance (Layer 4)
    - **Source Address**: Enter the source IP of the inline service's "from-service" interface. If there are multiple devices, you can enter a subnet (ex. 198.19.96.128/25).

  - **Destination Address/Mask**: Enter 8.8.8.8, assuming the inline service is also using this IP for external DNS. Again, with SNAT disabled on the inline service, it may simply be possible to define a wildcard here (0.0.0.0/0) to allow traffic to any destination.

  - **Service Port**: Enter 53, assuming this is the service port the inline service will send to. As with the above SNAT recommendation, this could also define a wildcard (*) to allow traffic to any destination port.

  - **Protocol**: Enter UDP, assuming DNS requests will be on UDP.

  - **VLAN and Tunnel Traffic**: Select the service's from-service VLAN. This is most critical setting. Traffic leaves the service on this VLAN, and normally consumed by -D-0- virtual service bound to that VLAN. This service control channel must listen on this same VLAN and represent a more specific path for outbound traffic.

  - **Source Address Translation**: If the SSL Orchestrator topology requires source NAT to egress, the service control channels will too. Enable this setting as required for outbound traffic flow.

  - **Address Translation**: Select Disabled.

  - **Port Translation**: Select Disabled.

  - **Default Pool**: Select an existing gateway pool such as the one created by the SSL Orchestrator topology (or create a new one). Note that selecting a pool may also enable address and port translation. Check these again after selecting the pool.

  Click the Finished button to complete the service control channel virtual server configuration. Depending on your environment, you may need to create multiple service control channels. However, as previously stated, if source NAT can be disabled on the inline device, it will usually only be necessary to create one listening on the device's from-service interface IP address.

  Also note that the service control channel binds to a VLAN, and potentially a gateway pool, used by an SSL Orchestrator-managed configuration. If the VLAN or pool was created by SSL Orchestrator, this control channel virtual server will need to unbind from this VLAN or pool if the service is to be removed or networking settings changed (since it will cause dependency errors). If VLANs are created in the BIG-IP (manually) and then consumed by the SSL Orchestrator configuration, there will be no dependency issues.

- **Test** – To verify the service channel is working, SSH to the proxy service and attempt to perform a DNS query to 8.8.8.8,

  ```
  dig @8.8.8.8 www.example.com
  ```

  Assuming this works, the proxy service can be configured to use this DNS service. Additional service channels can be created to provide direct access to other applications.

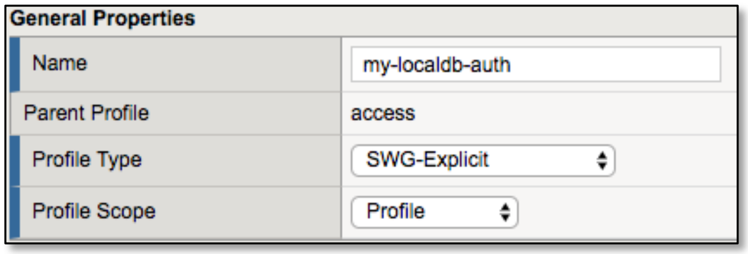
LAB 7 – CONFIGURE FORWARD PROXY AUTHENTICATION

When users must be authenticated to allow external access, an SSLO forward proxy topology (L3 and explicit proxy) can be configured to attach an Access policy. Forward proxy authentication therefore minimally requires the addition of the Access Policy Manager (APM) module. Authentication in a forward proxy environment can be generally classified as explicit or transparent.

Explicit proxy authentication

Explicit forward proxy (web) authentication employs a "407-based" mechanism, whereby the explicit proxy prompts the client for identity using an HTTP 407 challenge-response. Modern browsers only support Basic, NTLM and Kerberos for 407-based authentication, so explicit proxy is also limited to these. Enabling explicit proxy authentication in SSLO requires two steps,

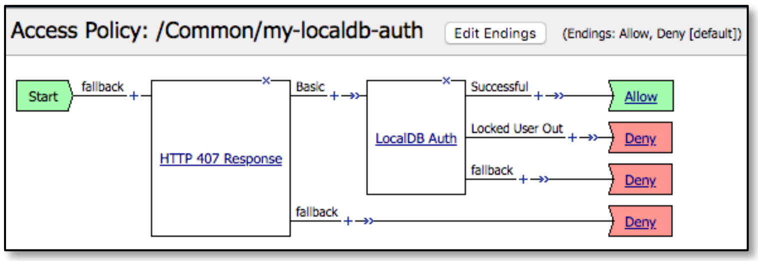
- **Create an SWG-Explicit access policy** – Explicit proxy authentication is defined as an access policy of type SWG-Explicit.



General Properties

Name	my-localdb-auth
Parent Profile	access
Profile Type	SWG-Explicit
Profile Scope	Profile

This policy will typically contain an HTTP 407 Response challenge, and then some form of authentication, which could HTTP Basic, NTLM or Kerberos.



- **Create or edit an Explicit Proxy SSLO topology and attach the SWG-Explicit access policy** – To attach the SWG-Explicit access policy to SSLO, create or edit an Explicit proxy SSLO topology. On the Interception Rules page, select this policy under the **Access Profile** option.

To test the above Basic authentication configuration, open a browser or cURL to an external resource. For cURL, include the explicit proxy address and client credentials.

```
curl --proxy 10.1.10.150:3128 --proxy-basic --proxy-user 'bob:pass' https://www.example.com
```

Transparent proxy (captive portal) authentication

In a traditional transparent forward proxy, the client/user is unaware of the proxy's existence. Client traffic is simply routed through the proxy. To authenticate users accessing the Internet through a transparent proxy, there are generally two options:

- **Direct** – Where the proxy inserts integrated authentication challenged (i.e. NTLM/Kerberos/Basic) directly in the path of outbound traffic. This implies that users would get challenged for local authentication when going to remote Internet sites, which is typically disabled by default in most browsers, and would need to be enabled. While many proxies support this authentication mode, it is not terribly practical or secure.

- **Captive portal** – Where the proxy redirects new user sessions to a separate authentication site. The site provides the necessary authentication challenge, then redirects the user back through the proxy. A session is then tracked by virtue of some unique client attribute. Captive portals are very often deployed in hotels, airports, airplanes and other public places to authenticate users for outbound Internet access. Captive portal authentication also has a distinct advantage over explicit proxy authentication in that it supports more than just integrated authentication (i.e. NTLM/Kerberos/Basic). A captive portal can, for example, support login page, client certificate, and federated authentication.

Note: it's also worth noting that any transparent proxy authentication mechanism requires cleartext access to the user traffic. The user's first request to the Internet must be over HTTP if not decrypting, so that the proxy can insert the direct challenge, or HTTP redirect to a captive portal. SSL Orchestrator generally solves this problem as a function of automatically decrypting outbound flows.

Transparent proxy captive portal authentication essentially flows like this,

- A new client opens a browser and attempts to connect to a remote Internet site (ex. www.google.com).

- The transparent proxy detects no session for this client, so inserts an HTTP redirect to its captive portal authentication service. As this is a separate site, the redirect must convey some information about the client, specifically the URL of the original request. This is usually provided in the query string of the redirect URL.

- The client accesses the captive portal, does some form of authentication, and is then redirected back to the origin URL. The origin URL is the one conveyed from the initial redirect in the query string.

- The client makes a second request to the origin URL through the transparent proxy and is permitted. Subsequent requests are tracked by whatever mechanism the proxy uses to identity the user.

As of SSLO 6.0, transparent proxy captive portal authentication is supported with the addition of Access Policy Manager (APM). The configuration requires two separate per-session access profiles – one attached to the SSLO transparent proxy configuration, and one attached to a separate virtual service configured to support the captive portal "login" process.

The following describes a minimal configuration using an APM Local DB user account and login page. As stated previously, this can also support integrated NTLM/Kerberos/Basic authentication, login pages, client certificates and federation, and can rely on Active Directory, LDAP and other identity management services.

- Optionally create the Local DB instance and any Local DB user accounts. This lab will use Local DB to authenticate users, but other identity management services may also be defined. Please see relevant APM documentation for the specific configuration steps needed for other authentication methods.

- Create an SWG-Transparent access profile. This profile will be attached to the separate captive portal "login" virtual server.

    - **Profile Type**: Select SWG-Transparent.

    - **Profile Scope**: SSLO introduces a new profile scope (named) for captive portal authentication that must match between it and the SWG-Transparent access profile. Select Named.

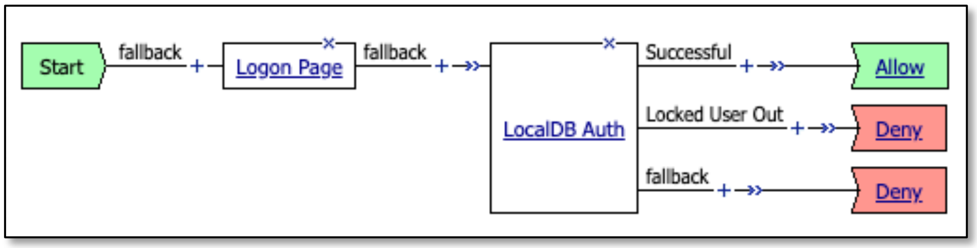
    - **Named Scope**: Enter a unique name here to represent the "authentication domain" shared between the two access profiles. For this lab, use something like "sslo".

    - **Customization Type**: This is a new option in 7.0, but for captive portal, must be set to "standard".

    - **Captive Portal**: Leave this as disabled. It is only enabled in the SSL Orchestrator access profile.

    - **Language**: Select the desired language.

    Once completed, access the profile's visual policy to create the desired authentication strategy. This lab demonstrates a simple Local DB authentication via logon page.



- Create a new SSL Orchestrator access profile. Captive portal authentication requires a separate access profile configuration not created directly within the SSLO workflow.

    - **Profile Type**: select SSL Orchestrator.

    - **Profile Scope**: select Named to match the SWG-Transparent access profile.

    - **Named Scope**: enter the same named scope used in the SWG-Transparent profile (ex. "sslo").

    - **Captive Portal**: set this to enabled.

- **Primary Authentication URI**: This is the URL that the SSLO transparent proxy will redirect new users to, represented by and resolving to the separate virtual server instance and SWG-Transparent access profile. This would be a full URL, example: https://login.f5labs.com.

- **Language**: Select the desired language.

Note: an SSL Orchestrator access profile does not have an editable visual policy.

- Create a client SSL profile. This will be needed by the captive portal login virtual server to handle incoming HTTPS communications.

- Create a virtual server. This virtual server will define the captive portal login instance that handles user authentication. Transparent proxy traffic will be redirected to this virtual server, so also ensure that the Captive Portal URL specified in the SSL Orchestrator access profile resolves to this VIP's IP address.

  - **Type**: Select Standard

  - **Destination Address/Mask, and Service Port**: Enter an IP address and port accessible to the client. This is the IP address that the specified captive portal URL should DNS resolve to.

  - **HTTP Profile (Client)**: The default http profile is adequate here.

  - **SSL Profile (Client)**: Select the previously created client SSL profile.

  - **VLANs and Tunnels**: Select the client facing VLAN here.

  - **Access Profile**: Select the SWG-Transparent access profile here.

- Attach the new SSL Orchestrator access profile to the SSLO transparent proxy topology. In the SSLO UI, click on the relevant transparent forward proxy topology, or create a new one as described in Lab 1. On the Interception Rule page, under **Access Profile**, select the new SSL Orchestrator access profile. Complete the topology workflow and (re)deploy.

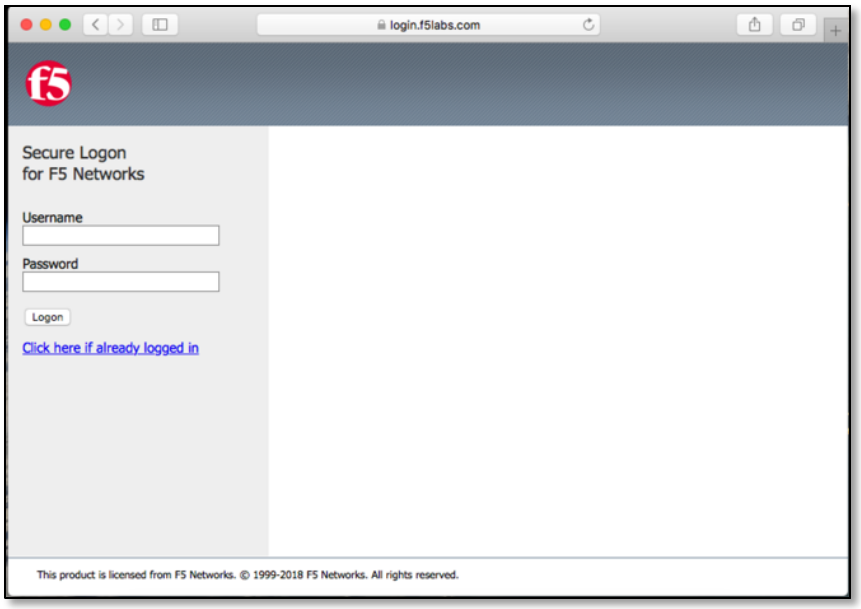
Note: when selecting a new access profile, a banner warning will be displayed above stating, "*If Access Profile is not topology specific the user cannot change the Log Settings in a topology deployment flow*". As log settings are now topology-specific, SSLO applies the selected options on the Log Settings page to the built-in SSLO access profile only. When a different access profile is selected, the Log Settings options are disabled for editing.

To re-acquire logging for the SSLO topology, create a new logging profile under Access – Overview – Event Logs – Settings

  - **General**: Enable both options (as required).
  - **Access System Logs**: Select the "/Common/sys-sslo-publisher" and select desired logging settings for each object.

- **URL Request Logs**: Select the "/Common/sys-sslo-publisher" and minimally enable "Blocked" and "Confirmed" events (as required).
- **Access Profiles**: Select the new SSL Orchestrator access profile. The SWG-Transparent access profile can also be selected (as required).

- Test the captive portal. On attempting to access an Internet site, a new user should be redirected to the login page defined in the SWG-Transparent access profile. Once authenticated, the user will be redirected back to the proxy and gain access to the origin URL content.



Secure Logon
for F5 Networks

This product is licensed from F5 Networks. © 1999-2018 F5 Networks. All rights reserved.

LAB 8 – DELETE AN SSL ORCHESTRATOR CONFIGURATION

One of the benefits of the new SSLO architecture is that configurations can be edited, deployed and re-deployed without affecting existing traffic flows. For this capability, the SSLO packaging is now broken into separate independent components. When deleting a defined topology, most of the attached components are also deleted. However, some objects, particularly those that can be consumed by multiple topologies, are not automatically deleted. This lab explores the different methods for deleting SSL Orchestrator objects.

Step 1: Deleting a topology

Deleting a topology will also delete any relying Interception Rules. The deletion process performs a complex set of REST-based tasks, therefore only one topology can be deleted at a time. In the SSLO UI, select a topology and click the Delete button. Confirm that both the topology and respective interception rules are removed.

Step 2: Deleting other objects

While deleting a topology also removes its respective interception rules, it does not remove the other objects - services, service chains, security policies and SSL settings. These can all be removed individually, however must be deleted in a hierarchical order. Once the topology and interception rules have been deleted,

- SSL Settings can be deleted any time
- Delete any unused Security Policies
- Delete any unused Service Chains
- Delete any unused Services

Step 3: Deleting everything

To completely remove the SSLO configuration and start from scratch,

- In the SSLO UI, click Delete Configurations and then click OK. This process will take some time as SSLO walks through all of the objects and dependencies to remove all configurations.
- Under the iApps menu, Application Services, Applications LX – Un-deploy any remaining SSL orchestrator objects. If using any other Guided Configuration engine (ex. Access GC), ensure that only SSLO objects are deleted here.
- Under the iApps menu, Templates, Templates LX – Delete all of the SSL Orchestrator templates.
- Under the iApps menu, Package management LX – Delete the SSL Orchestrator package.

The next time the SSL Orchestrator configuration menu is accessed, SSLO will automatically restore the on-box package.

Optional: Deleting everything…the hard way

In the unlikely event that the above steps do not work, and some SSLO objects remain and cannot be deleted, one of the following steps can be used,

- If the topology and interception rules are gone but other objects remain and will not uninstall in the SSL Orchestrator UI, in the BIG-IP UI navigate to iApps -> Application Services -> Applications LX. The remaining objects will all be here in states of deployed (green), undeployed (gray), and error (red). Delete any objects in an error state and toggle the other objects from deployed to undeployed and back until they enter an error state and can also be deleted.

- If the above fails, the following script can be used to automate destruction of SSLO objects.

  Note: this is an option of last resort if the SSLO configuration is beyond saving.

  - Copy the script to the BIG-IP (ex. sslo-nuclear-delete.sh)
  - Chmod the script to give it execute privileges: `chmod +x sslo-nuclear-delete.sh`
  - Execute the script: `./sslo-nuclear-delete.sh`
  - It will typically be necessary to execute the script several times to get through dependencies. It is completely done when the script returns quickly with no additional output. Validate that all SSLO objects are gone from the BIG-IP UI under the Local Traffic and Network sections.
  - Under the iApps menu, Application Services, Applications LX – Un-deploy any remaining SSL orchestrator objects. If using any other Guided Configuration engine (ex. Access GC), ensure that only SSLO objects are deleted here.
  - Under the iApps menu, Templates, Templates LX – Delete all of the SSL Orchestrator templates.
  - Under the iApps menu, Package management LX – Delete the SSL Orchestrator package.

  The nuke-delete script can be found here: https://github.com/f5devcentral/sslo-script-tools/tree/main/sslo-nuke-delete.

  Note: in this lab, the sslo-nuclear-delete.sh script already exists under the /config/dev directory.

- If the above fails, manually clear the REST database from the command line,

  - Break any HA configuration
  - Issue the '`clear-rest-storage [options]`' command, where the options are "-l" (lowercase L) to delete the restjavad log files as well as the stored state, and "-d" to reset the system configuration to default. This command will remove all SSL Orchestrator objects from the restnoded database. After issuing this command, follow with '`bigstart restart restnoded`' and '`bigstart restart restjavad`', clear the browser cache, log out and back in. The recommended method here is to simply issue the -l option.

    ```
    clear-rest-storage -l
    ```

- Issue the '`tmsh delete sys application service recursive`' command to also delete any remaining SSL Orchestrator application service objects.
- Once all SSLO objects have been removed, also uninstall the SSLO RPM package under the iApps menu, Package management LX – delete the SSL Orchestrator package.
- Rebuild HA and redeploy SSLO by navigating to the SSL Orchestrator configuration UI. On first visit it will automatically restore the on-box package.

Lab 9 – Test TLS 1.3

Formally defined in RFC 8446, TLS 1.3 is a significant enhancement to modern "secure" Internet communications.

SSLO now supports reverse proxy (SSLO inbound) and forward proxy (SSLO outbound) handling. To configure inbound TLS 1.3 support, perform the following steps.

- Create a new Cipher Rule to represent the more condensed set of ciphers supported in TLS 1.3. In the BIG-IP UI, under Local Traffic – Ciphers – Rules, click Create.
    - **Cipher Suites**: Enter the following string to enable the three TLS 1.3 ciphers currently supported in BIG-IP 15.0:

      ```
      TLS13-AES128-GCM-SHA256:TLS13-AES256-GCM-SHA384:TLS13-CHACHA20-POLY1305-SHA256
      ```

    - **DH Groups**: Enter the keyword DEFAULT.
    - **Signature Algorithms**: Enter the keyword DEFAULT.

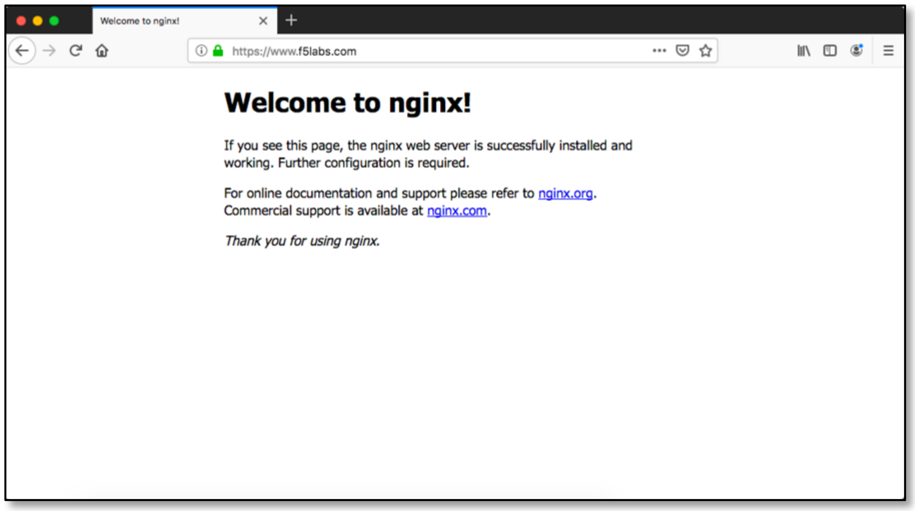
  Note: It is also acceptable to simply use the built-in "f5-default" cipher group, as that contains the two TLS13-AES-GCM ciphers.

- Create a new Cipher Group to contain the Cipher Rule. The SSLO SSL configuration will reference this Cipher Group. In the "Allow the following" column, add the previously created Cipher Rule.
- Create a new SSLO topology or edit an existing one. On the SSL Configuration page, in the top right corner click the "Show Advanced Setting" link.
    - **Client-side SSL (Processing Options)**: TLS 1.3 is disabled by default, by function of the disabled "TLSv1.3" option. To enable TLS 1.3, move this item from the Disabled Options column to the Enabled Options column.
    - **Client-side SSL (Cipher Type)**: TLS 1.3 requires a Cipher Group. Select this option then specify the previously created Cipher Group.
    - Optionally enable TLS 1.3 in the same manner on the server-side, if server-side resources support and require TLS 1.3.
- (Re)Deploy.

To test TLS 1.3 through an inbound SSLO topology, enable TLS 1.3 support in a browser and access the internal websites through SSLO. See https://geekflare.com/enable-tls-1-3-in-browsers/ for tips on enabling TLS 1.3 support in the various browsers.

Note: If performing these TLS 1.3 tests from the SSL Orchestrator UDF labs, the Ubuntu desktop clients are version 14.04, and too old to support TLS 1.3 (even for browsers). For best results, install an Ubuntu 18.04 client.

For example, to force Firefox to use TLS 1.3 (only), open the browser and navigate to about:config. Search for ‘security.tls.version.min’ and ‘security.tls.version.max’. To force TLS 1.3, set them both to a value of 4. The below screenshots show a successful TLS 1.3 HTTPS request through SSLO, and a client-side Wireshark capture of that transaction.

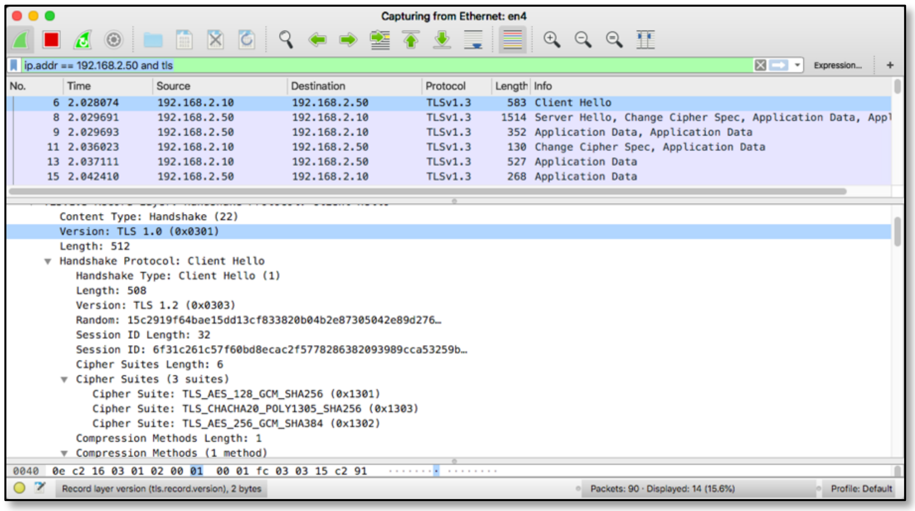


Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



```
ip.addr == 192.168.2.50 and tls

No.  Time      Source         Destination    Protocol  Length Info
  6 2.028074   192.168.2.10   192.168.2.50   TLSv1.3    583 Client Hello
  8 2.029691   192.168.2.50   192.168.2.10   TLSv1.3   1514 Server Hello, Change Cipher Spec, Application Data, Appl
  9 2.029693   192.168.2.50   192.168.2.10   TLSv1.3    352 Application Data, Application Data
 11 2.036023   192.168.2.10   192.168.2.50   TLSv1.3    130 Change Cipher Spec, Application Data
 13 2.037111   192.168.2.10   192.168.2.50   TLSv1.3    527 Application Data
 15 2.042410   192.168.2.50   192.168.2.10   TLSv1.3    268 Application Data

      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 512
    ▼ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 508
        Version: TLS 1.2 (0x0303)
        Random: 15c2919f64bae15dd13cf833820b04b2e87305042e89d276…
        Session ID Length: 32
        Session ID: 6f31c261c57f60bd8ecac2f5778286382093989cca53259b…
        Cipher Suites Length: 6
      ▼ Cipher Suites (3 suites)
          Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
          Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
          Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
        Compression Methods Length: 1
      ▼ Compression Methods (1 method)

0040  0e c2 16 03 01 02 00 01  00 01 fc 03 03 15 c2 91   ········ ·········

Record layer version (tls.record.version), 2 bytes    Packets: 90 · Displayed: 14 (15.6%)    Profile: Default
```

To test an outbound SSLO topology, navigate to https://www.cloudflare.com.

Note: Using the “f5-default” cipher group on the server SSL profile in an SSLO outbound topology, a server-side wire capture will show an initial TLS 1.2 attempt. To force a server-side TLS 1.3 handshake, create a new cipher rule and cipher group using the dedicated cipher string noted above.

Lab 10 – Manage SSL Orchestrator High Availability

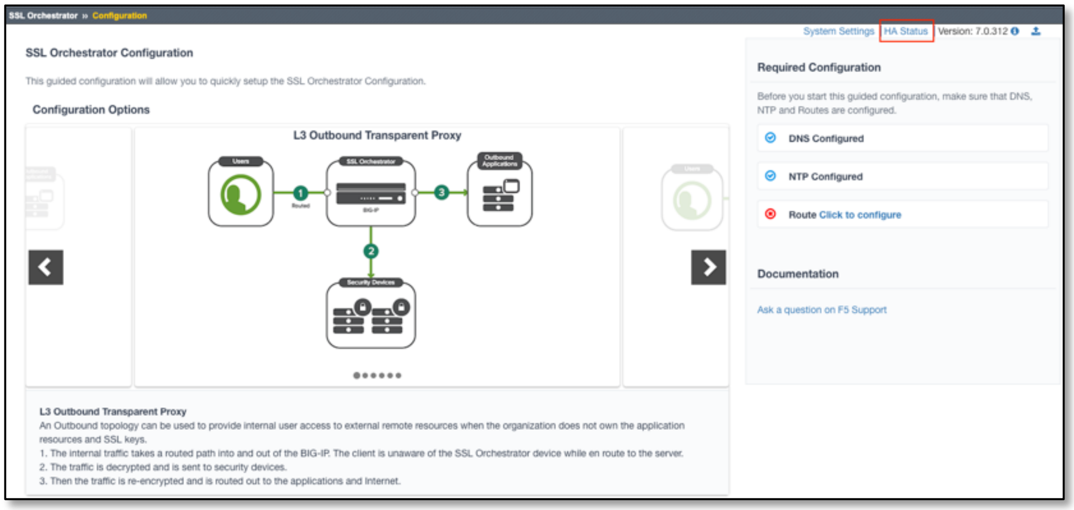
SSL Orchestrator high availability presents a different model for HA sync/failover than normal BIG-IP HA. Specifically, SSLO relies on a separate REST-based process to perform internal sync between the peers and does not specifically rely on BIG-IP mcpd. It is therefore important to understand the conditions and caveats of this separate process.

- An SSL Orchestrator system must be configured for **MANUAL with INCREMENTAL SYNC**. The REST operations take care of synchronizing the SSLO configurations. This will also, at times, trigger a warning on the BIG-IP that the boxes are out of sync, though they are not. It is acceptable to ignore these warnings, though it is also possible to manually sync the boxes **AFTER** the SSLO sync process.
- HA must be configured and stable **BEFORE** deploying SSLO.

Note: the extra BIG-IP in this lab environment is perfectly suited to demonstrate SSLO HA. Simply configure identical networking, floating self-IPs, resource provisioning, and set up as an HA peer. Use the ICAP/DLP or TAP service VLAN for the sync channel (whichever won't be needed).

To deploy SSLO in a High Availability pair, perform the following operations.

- Confirm that you have administrative access to the BIG-IP UI and to the shell (SSH).
- Verify that the BIG-IP .iso is the same on both devices.
- Check port lockdown settings on both HA devices and verify that the designated sync and failover VLANs use either "Allow All" or "Allow Default" in their Self-IP configurations.
- Verify that NTP and DNS settings match on both HA devices.
- Check HA sync failover group settings (Device Management – Device Groups). Verify that a device group exists, that the device group type is "Sync-Failover", that all HA devices are in the "includes" field, and that sync type is "Manual with Incremental Sync".
- Verify that overview status (Device Management – Overview Status) shows no warnings or errors.
- Perform the typical BIG-IP HA setup process and confirm that the boxes are in a good HA state. New in SSLO is an **HA Status** page that displays current state of requisite HA, including overall HA status REST communications (gossip).


SSL Orchestrator » Configuration

System Settings | HA Status | Version: 7.0.312

SSL Orchestrator Configuration

This guided configuration will allow you to quickly setup the SSL Orchestrator Configuration.

Configuration Options

L3 Outbound Transparent Proxy
An Outbound topology can be used to provide internal user access to external remote resources when the organization does not own the application resources and SSL keys.
1. The internal traffic takes a routed path into and out of the BIG-IP. The client is unaware of the SSL Orchestrator device while en route to the server.
2. The traffic is decrypted and is sent to security devices.
3. Then the traffic is re-encrypted and is routed out to the applications and Internet.

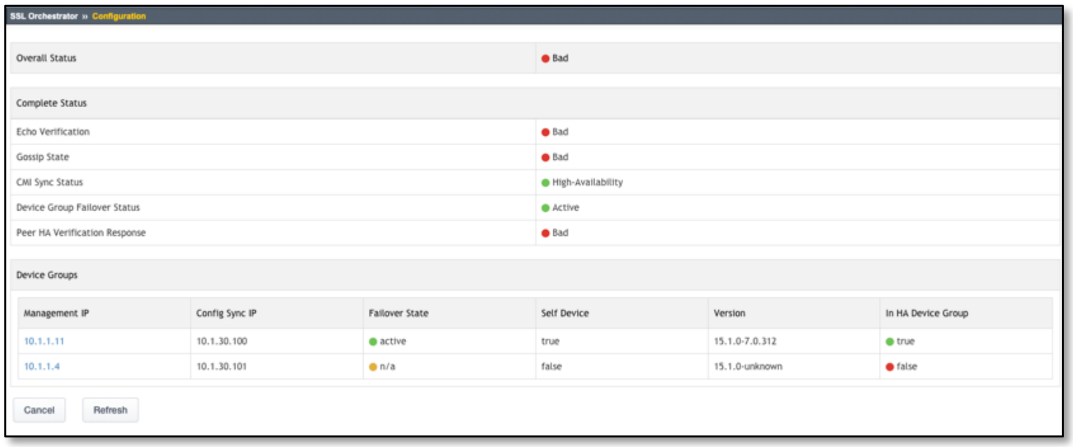
Required Configuration

Before you start this guided configuration, make sure that DNS, NTP and Routes are configured.

- DNS Configured
- NTP Configured
- Route Click to configure

Documentation

Ask a question on F5 Support

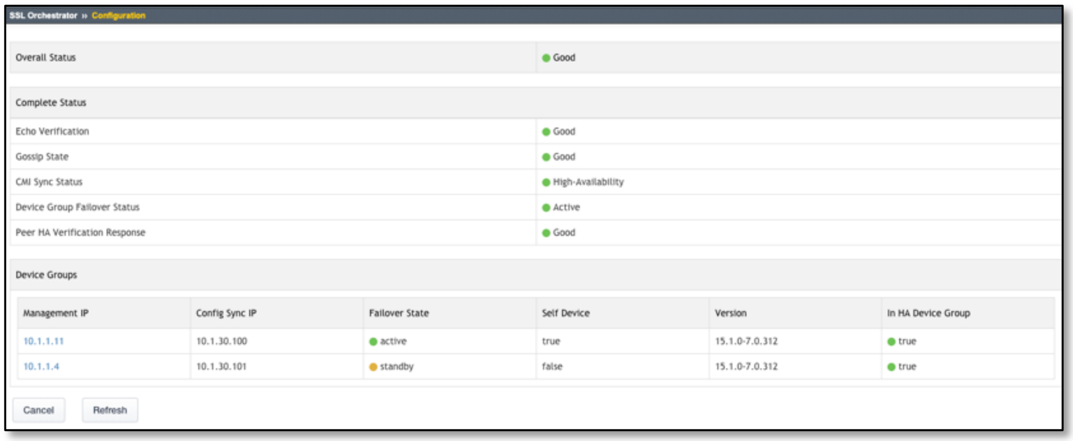

SSL Orchestrator » Configuration

Overall Status	● Bad
Complete Status	
Echo Verification	● Bad
Gossip State	● Bad
CMI Sync Status	● High-Availability
Device Group Failover Status	● Active
Peer HA Verification Response	● Bad

Device Groups

Management IP	Config Sync IP	Failover State	Self Device	Version	In HA Device Group
10.1.1.11	10.1.30.100	● active	true	15.1.0-7.0.312	● true
10.1.1.4	10.1.30.101	● n/a	false	15.1.0-unknown	● false

Cancel | Refresh


SSL Orchestrator » Configuration

Overall Status	● Good
Complete Status	
Echo Verification	● Good
Gossip State	● Good
CMI Sync Status	● High-Availability
Device Group Failover Status	● Active
Peer HA Verification Response	● Good

Device Groups

Management IP	Config Sync IP	Failover State	Self Device	Version	In HA Device Group
10.1.1.11	10.1.30.100	● active	true	15.1.0-7.0.312	● true
10.1.1.4	10.1.30.101	● standby	false	15.1.0-7.0.312	● true

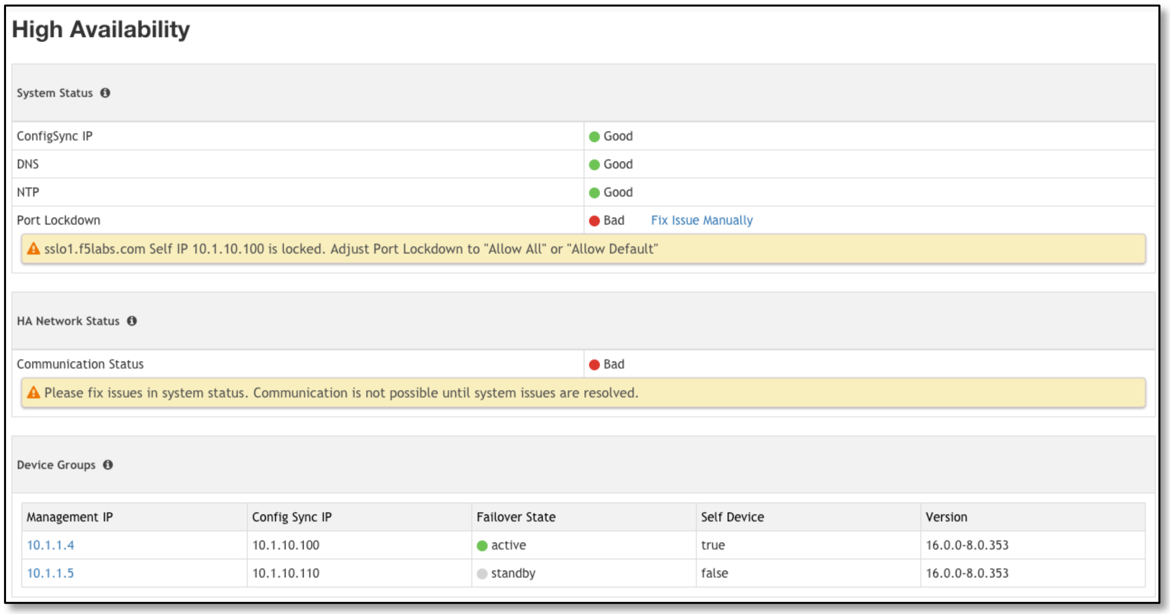
Cancel | Refresh

- Deploy SSL Orchestrator. The SSLO peers will sync their configs via REST calls between them. This may at times present a warning in the UI that the boxes are out of sync, though they are not. Review both devices to verify like settings. It is also acceptable to perform a manual sync AFTER the SSLO sync process.

If any of these values are not “Good”, the following steps may help to clear up errors.

- Navigate to the iApps Package Management menu and verify that the correct .rpm package is installed or replace if not.
- Confirm HA state by reviewing logs (restnoded, restjavad, and tmm) for any errors relating to HA/sync.
- Navigate to *https://<management-ip>/mgmt/shared/gossip* on both HA devices. Verify that the “status” value indicated “ACTIVE”.
- Navigate to https://<management-ip>/mgmt/tm/cm/device on both HA devices. Verify that the item count returned is the same on both. Verify that the “configsyncip” attribute exists and matches the HA VLAN IP on each corresponding device. And also verify that the “unicastAddress” exists, and that “configsyncip” is contained in this value. Usually the management IP should also be an attribute of “unicastAddress”.
- Navigate to https://management-ip/mgmt/shared/resolver/device-groups/tm-shared-all-big-ips/devices on both HA devices. Verify that the returned information is the same on both boxes. Specifically, validate the “address” attribute exists and matches the corresponding “configsyncip”.
- Navigate to https://management-ip/mgmt/tm/shared/bigip-failover-state on both HA devices. Verify that the “failoverState” value is “active” for the active device, and “standby” for the standby device.

New in SSLO 8, an HA remediation option is available when issues are detected. Depending on the issue, the utility will either be able to fix it directly or send you to the UI where the fix is required.


High Availability

System Status

ConfigSync IP	● Good
DNS	● Good
NTP	● Good
Port Lockdown	● Bad Fix Issue Manually

⚠ sslo1.f5labs.com Self IP 10.1.10.100 is locked. Adjust Port Lockdown to "Allow All" or "Allow Default"

HA Network Status

Communication Status	● Bad

⚠ Please fix issues in system status. Communication is not possible until system issues are resolved.

Device Groups

Management IP	Config Sync IP	Failover State	Self Device	Version
10.1.1.4	10.1.10.100	● active	true	16.0.0-8.0.353
10.1.1.5	10.1.10.110	● standby	false	16.0.0-8.0.353

LAB 11 – CREATE ICAP FILTERS

Among all of the security service types, ICAP processing typically drives the most latency, as SSLO as the ICAP client must pass the entire request and response payload to an ICAP server and wait for a response. It is often useful then to limit what gets sent to the ICAP server and when. This is accomplished with an SSL Orchestrator feature called **ICAP filters**. An ICAP filter is essentially an LTM policy object that enables and disables the ICAP "Adapt" profiles based on qualities of an HTTP request and/or response. The default action in an SSLO ICAP service configuration is to send all traffic to ICAP, so the practical approach to an ICAP filter is to disable ICAP when traffic does not match a required flow type.

There are many different scenarios for disabling ICAP processing, so this lab will only cover some of the more common use cases. There are also many different ways to accomplish each of the below use cases, so the following only covers one possible method for each use case.

Enable ICAP only for HTTP response traffic
In this use case, ICAP processing will be enabled only for HTTP response traffic.

- In the BIG-IP UI, under Local Traffic -> Policies, create a new policy.
- In the new policy configuration, create a new rule to disable all request Adapt processing:
  - Match all of the following conditions:
    - All traffic
  - Do the following when the traffic is matched:
    - Disable Request Adapt at request
- Create a second rule to enable Adapt processing if the HTTP Status exists in a response:
  - Match all of the following conditions:
    - HTTP Status (full string) exists at response
  - Do the following when the traffic is matched
    - Enable Response Adapt at response
- Save and publish the policy.
- In the SSLO UI, under Services, create an ICAP service, or edit an existing. At the bottom of the ICAP configuration, under ICAP Policy, select the LTM policy.
- Deploy and test. The ICAP server should now only see HTTP response traffic.

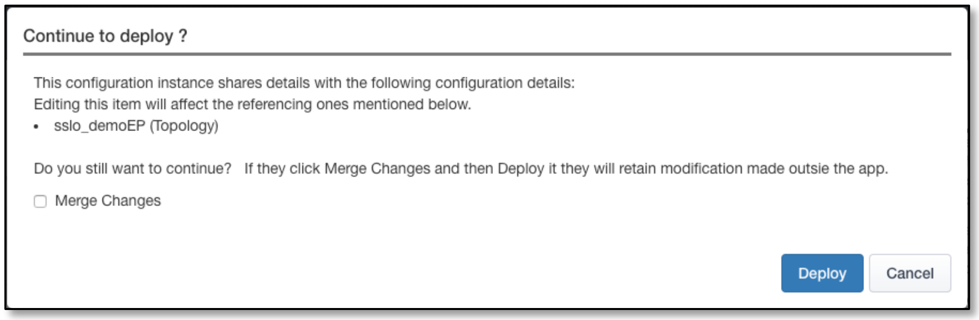
Enable ICAP only for HTTP POST requests
In this use case, ICAP processing will be enabled only for HTTP POST requests.

- In the BIG-IP UI, under Local Traffic -> Policies, create a new policy.
- In the new policy configuration, create a new rule to enable Adapt processing for HTTP POST requests:
  - Match all of the following conditions:
    - HTTP Method (full string) is 'POST' at request time
  - Do the following when the traffic is matched
    - Enable Request Adapt at request time
    - Disable Response Adapt at response time (optional)
- Create a second rule to disable everything else:
  - Match all of the following conditions:
    - All traffic

    - Do the following when the traffic is matched
        - Disable Request Adapt at request time
        - Disable Response Adapt at response time
- Save and publish the policy.
- In the SSLO UI, under Services, create an ICAP service, or edit an existing. At the bottom of the ICAP configuration, under ICAP Policy, select the LTM policy.
- Deploy and test. The ICAP server should now only see HTTP POST request traffic. If the optional action is not created in the first rule (*disable response adapt at response time*), the corresponding response to this HTTP POST request will also flow to the ICAP server.

LAB 12 – MANAGE STRICT UPDATES MODIFICATIONS (PREVIEW)

In previous versions of SSL Orchestrator (5.0+), if any configuration object is taken out of strictness mode, that object's UI configuration becomes read-only. The SSL Orchestrator configuration naturally attempts to solve for the "majority" of visibility use cases but cannot solve for them all. The ability to "tweak" data flow configurations and properties on an F5 BIG-IP is an important characteristic and competitive differentiator, thus the concept of configuration "strictness" can sometimes be too restrictive for many environments. New in SSLO is the concept of **strict updates modification**. Essentially, the SSLO configuration allows non-strict changes to the SSLO configuration – changes made outside of the SSLO UI – to persist through SSLO object deployments. When an object is re-deployed after non-strict updates, a new UI provides the option to keep ("Merge Changes") any non-strict changes to the object. Leaving the Merge Changes option unchecked overwrites any non-strict updates.

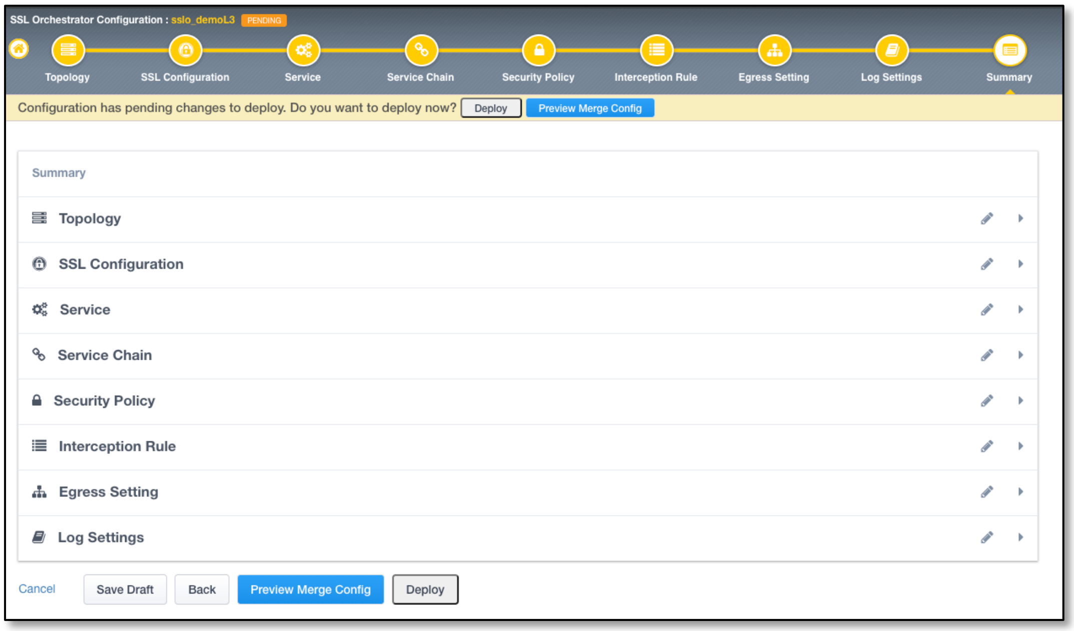

Continue to deploy ?

This configuration instance shares details with the following configuration details:
Editing this item will affect the referencing ones mentioned below.
- sslo_demoEP (Topology)

Do you still want to continue?   If they click Merge Changes and then Deploy it they will retain modification made outsie the app.

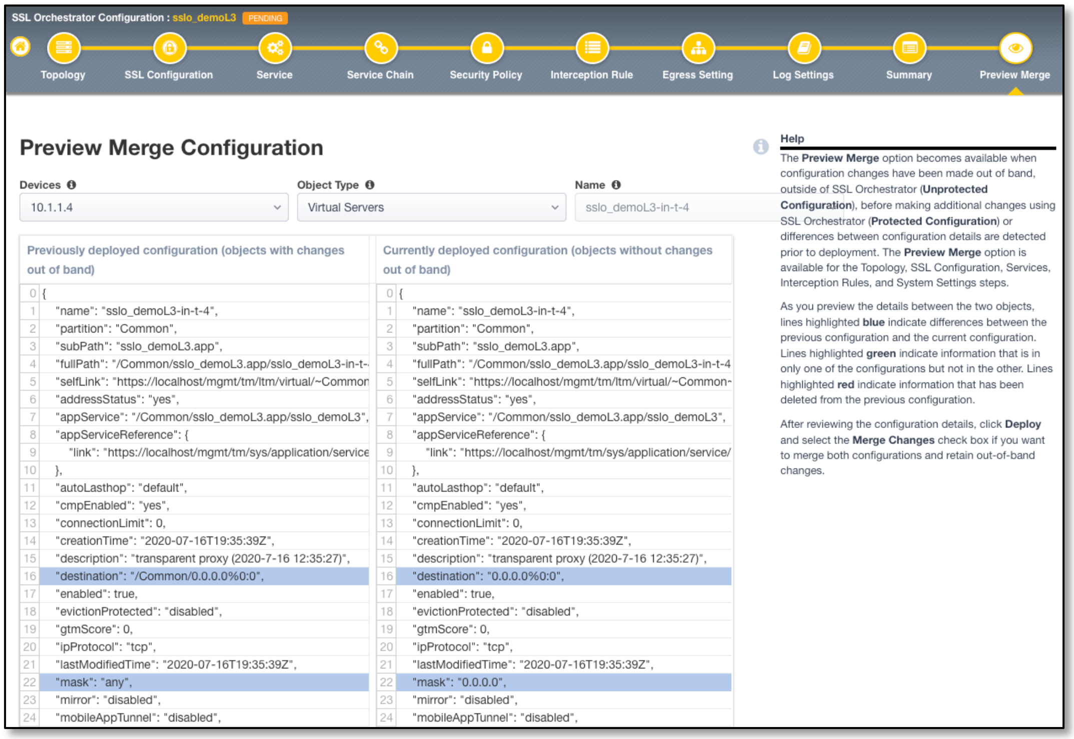
☐ Merge Changes

SSLO 8 now includes a **Preview** option when you attempt to deploy with non-strict changes applied.


SSL Orchestrator Configuration : sslo_demoL3 PENDING

Configuration has pending changes to deploy. Do you want to deploy now?

Click the **Preview Merge Config** button to see the before and after configurations side-by-side.



Preview Merge Configuration

Devices: 10.1.1.4

Object Type: Virtual Servers

Name: sslo_demoL3-in-t-4

Previously deployed configuration (objects with changes out of band)

```
{
    "name": "sslo_demoL3-in-t-4",
    "partition": "Common",
    "subPath": "sslo_demoL3.app",
    "fullPath": "/Common/sslo_demoL3.app/sslo_demoL3-in-t-
    "selfLink": "https://localhost/mgmt/tm/ltm/virtual/~Common
    "addressStatus": "yes",
    "appService": "/Common/sslo_demoL3.app/sslo_demoL3",
    "appServiceReference": {
        "link": "https://localhost/mgmt/tm/sys/application/service
    },
    "autoLasthop": "default",
    "cmpEnabled": "yes",
    "connectionLimit": 0,
    "creationTime": "2020-07-16T19:35:39Z",
    "description": "transparent proxy (2020-7-16 12:35:27)",
    "destination": "/Common/0.0.0.0%0:0",
    "enabled": true,
    "evictionProtected": "disabled",
    "gtmScore": 0,
    "ipProtocol": "tcp",
    "lastModifiedTime": "2020-07-16T19:35:39Z",
    "mask": "any",
    "mirror": "disabled",
    "mobileAppTunnel": "disabled",
```

Currently deployed configuration (objects without changes out of band)

```
{
    "name": "sslo_demoL3-in-t-4",
    "partition": "Common",
    "subPath": "sslo_demoL3.app",
    "fullPath": "/Common/sslo_demoL3.app/sslo_demoL3-in-t-4
    "selfLink": "https://localhost/mgmt/tm/ltm/virtual/~Common-
    "addressStatus": "yes",
    "appService": "/Common/sslo_demoL3.app/sslo_demoL3",
    "appServiceReference": {
        "link": "https://localhost/mgmt/tm/sys/application/service/
    },
    "autoLasthop": "default",
    "cmpEnabled": "yes",
    "connectionLimit": 0,
    "creationTime": "2020-07-16T19:35:39Z",
    "description": "transparent proxy (2020-7-16 12:35:27)",
    "destination": "0.0.0.0%0:0",
    "enabled": true,
    "evictionProtected": "disabled",
    "gtmScore": 0,
    "ipProtocol": "tcp",
    "lastModifiedTime": "2020-07-16T19:35:39Z",
    "mask": "0.0.0.0",
    "mirror": "disabled",
    "mobileAppTunnel": "disabled",
```

Help

The **Preview Merge** option becomes available when configuration changes have been made out of band, outside of SSL Orchestrator (**Unprotected Configuration**), before making additional changes using SSL Orchestrator (**Protected Configuration**) or differences between configuration details are detected prior to deployment. The **Preview Merge** option is available for the Topology, SSL Configuration, Services, Interception Rules, and System Settings steps.

As you preview the details between the two objects, lines highlighted **blue** indicate differences between the previous configuration and the current configuration. Lines highlighted **green** indicate information that is in only one of the configurations but not in the other. Lines highlighted **red** indicate information that has been deleted from the previous configuration.

After reviewing the configuration details, click **Deploy** and select the **Merge Changes** check box if you want to merge both configurations and retain out-of-band changes.

The strict updates modification option, however, presents a set of caveats:

- When the Merge Changes option is not selected, any non-strict changes are overwritten, and the object is re-locked (strictness enabled)

- Security policies do not benefit from the new strict updates modification feature. There is no reasonable expectation that any non-strict updates made to the security policy visual policy can be retained when a security policy is re-deployed. All other "strict" object types (topologies, services, and SSL configurations) can employ the feature.

- Any non-strict updates that are in direct conflict with SSLO object changes (ex. SSL ciphers) are reset to the SSLO-configured setting.

To test this new feature:

- Deploy a typical SSLO topology, then take one of the object types of strictness mode by unlocking its respective object (ex. SSL Configuration).

- Make an out-of-band change to that object (ex. disable Generic Alerts in the respective client SSL profile).

- Go back to the object in the SSLO UI, make a change, and re-deploy. When the above message appears, click the Preview **Merge Changes** button to review the differences between the running and proposed config, and then click **Deploy** to commit the changes. Notice in the LTM object that the non-strict change persists.

LAB 13 – SOLVE HA ISSUES WITH THE HA-SYNC SCRIPT

SSL Orchestrator users with an HA setup may use the ha-sync tool and script to troubleshoot and fix HA setup issues (such as when gossip has gone out of sync, when some REST blocks are missing/out of sync, or even when MCP data is out of sync between devices).

The ha-sync script includes the diagnostic capability to identify potential issues and can print out all of the issues found with the HA setup. The ha-sync script can then perform a sync-up, which should fix those issues, and ensure that both devices are fully in sync (both in MCP and REST). See the following for additional details:

https://techdocs.f5.com/en-us/bigip-16-0-0/ssl-orchestrator-ha-sync-repair-tool.html

In SSL Orchestrator deployments, where a service has been created, you may need to manually create non-syncable network objects if they are missing on the peer device before using the ha-sync script. These network objects include VLANs, non-floating IPs, and route domains created for the service.

Below is the usage for the ha-sync script:

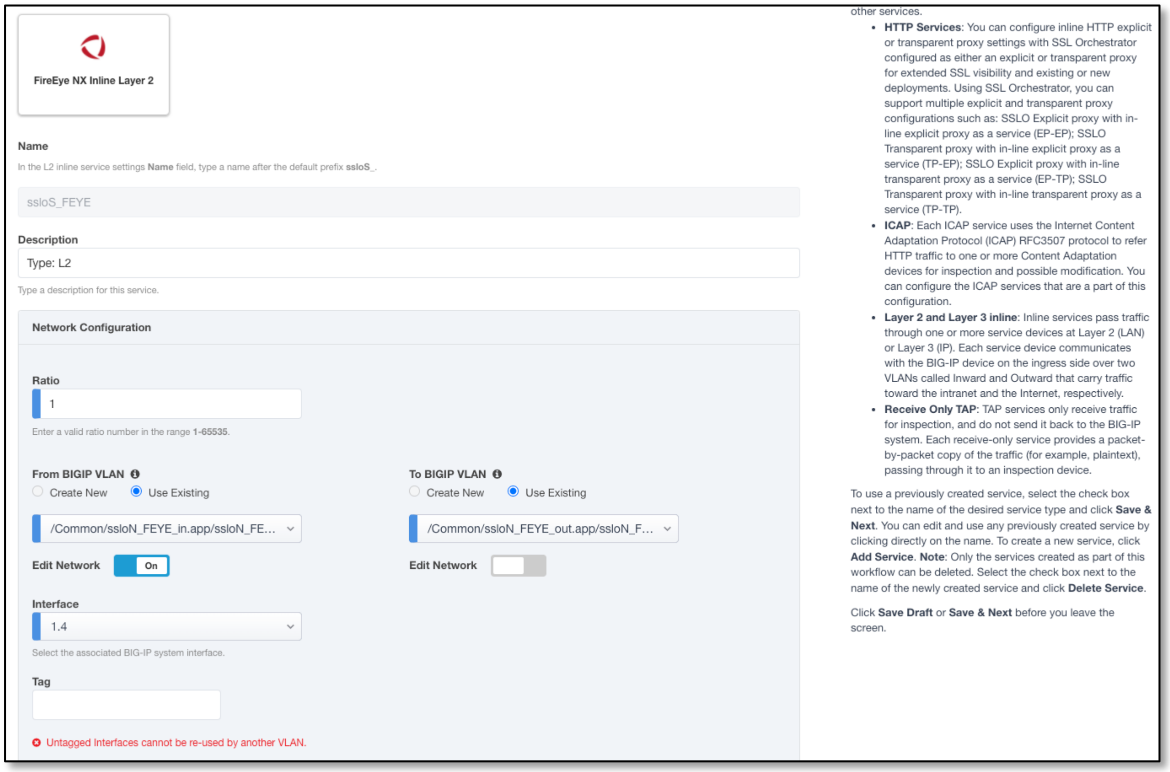
```
# ./ha-sync -h
BIG-IP HA sync repair helper
Usage: ha-sync [OPTIONS]...

-d, --dryrun              Dry-run (simulation) mode
-D, --devicegroup NAME    Specifies the HA device group name. Default: HA_GROUP
--diagnostic              Runs a diagnostic and attempts to detect      possible HA sync problems
-f, --force               Enforces a more coercive HA sync (see README for details)
-h, --help                Displays help text
-H, --host HA_PEER        Specifies the HA sync peer
-l, --localonly           Attempts a local repair only, without touching the remote HA peer
-m, --manual              Manual (step-by-step) mode
-t, --target [NAMES]...   Specifies the HA sync target(s) [ALL MCP REST]. Default: ALL
-v, --verbose             Provides additional (debug) information
-V, --version             Displays the current version of this script

Examples:
         ha-sync -H 10.192.228.78
         ha-sync -H 10.192.228.78 -d -m
         ha-sync -H 10.192.228.78 -d -m -f -t mcp
```

LAB 14 – MODIFY SERVICE NETWORK OBJECTS

In previous versions, in order to change network properties of an inline service (ex. VLANs, IPs, interfaces) you would have to delete and recreate the service. Now network properties can be edited directly inside the inline service configuration. To do this, in the SSLO UI navigate to **SSL Orchestrator -> Configuration**, click the **Services** tab and click again on the service to be edited.

For inline L2 services

In the Network Configuration section click on the pencil icon next to the device to be edited. Under the **From BIG-IP VLAN** and **To BIGIP VLAN** sections find a new "Edit Network" control. Enable this option for the network setting to be edited. For inline L2 services you will be able to adjust the interface and VLAN tag on an existing VLAN or create a new VLAN.



Name

In the L2 inline service settings **Name** field, type a name after the default prefix **ssloS_**.

ssloS_FEYE

Description

Type: L2

Type a description for this service.

Network Configuration

Ratio

1

Enter a valid ratio number in the range **1-65535**.

From BIGIP VLAN

Create New [✓ Use Existing]

/Common/ssloN_FEYE_in.app/ssloN_FE...

Edit Network On

Interface

1.4

Select the associated BIG-IP system interface.

Tag

Untagged Interfaces cannot be re-used by another VLAN.

To BIGIP VLAN

Create New [✓ Use Existing]

/Common/ssloN_FEYE_out.app/ssloN_F...

Edit Network

other services.

- **HTTP Services**: You can configure inline HTTP explicit or transparent proxy settings with SSL Orchestrator configured as either an explicit or transparent proxy for extended SSL visibility and existing or new deployments. Using SSL Orchestrator, you can support multiple explicit and transparent proxy configurations such as: SSLO Explicit proxy with in-line explicit proxy as a service (EP-EP); SSLO Transparent proxy with in-line explicit proxy as a service (TP-EP); SSLO Explicit proxy with in-line transparent proxy as a service (EP-TP); SSLO Transparent proxy with in-line transparent proxy as a service (TP-TP).
- **ICAP**: Each ICAP service uses the Internet Content Adaptation Protocol (ICAP) RFC3507 protocol to refer HTTP traffic to one or more Content Adaptation devices for inspection and possible modification. You can configure the ICAP services that are a part of this configuration.
- **Layer 2 and Layer 3 inline**: Inline services pass traffic through one or more service devices at Layer 2 (LAN) or Layer 3 (IP). Each service device communicates with the BIG-IP device on the ingress side over two VLANs called Inward and Outward that carry traffic toward the intranet and the Internet, respectively.
- **Receive Only TAP**: TAP services only receive traffic for inspection, and do not send it back to the BIG-IP system. Each receive-only service provides a packet-by-packet copy of the traffic (for example, plaintext), passing through it to an inspection device.

To use a previously created service, select the check box next to the name of the desired service type and click **Save & Next**. You can edit and use any previously created service by clicking directly on the name. To create a new service, click **Add Service**. **Note**: Only the services created as part of this workflow can be deleted. Select the check box next to the name of the newly created service and click **Delete Service**.

Click **Save Draft** or **Save & Next** before you leave the screen.

For inline L3/HTTP services

In the Service Definition section, under either the **To Service Configuration** or **From Service Configuration** section, find a new "Edit Network" control. For the network to be edited, enable this option. For inline L3 services you will be able to adjust the interface and VLAN tag on an existing VLAN or create a new VLAN.

LAB 15 – DEFINE SWG AS AN INLINE SERVICE

As of SSL Orchestrator 8.2 you are now able to insert F5's Secure Web Gateway (SWG) as an inline security service in a service chain for decrypted traffic. SWG allows you to perform transactional web gateway functions, like URL filtering, block/continues operations, and request/response (malware) analytics. This new feature presents an exciting new opportunity to displace third party web gateway products and reduce complexity in the security stack.

Note: SWG as an inline service was introduced in SSL Orchestrator 8.2 as an early access (hidden) feature. It becomes fully accessible in version 9.2. To unhide the SWG service icon in versions between 8.2 and 9.2, issue the following REST call from the BIG-IP console:

```
curl -d '{"F5_SWG_SERVICE_VISIBILITY": true}' -H "Content-Type: application/json" -X POST
http://localhost:8105/shared/iapp/f5-iappslx-ssl-orchestrator/globalSettings
```

- As noted above, SWG currently maintains a per-session concurrency and is limited to a maximum concurrency count that may be lower than the available access sessions per platform. Please see: https://support.f5.com/csp/article/K01151630 for a listing of per-platform SWG maximums. In some cases, a new max SWG count may be available (at higher price) to match the access session count. Please consult Sales Operations for more information.

- This release is intended as an "on-box" solution, where SWG is provisioned on the SSL Orchestrator appliance. An off-box option will be available in a later release.

- An SWG per-request policy may consume considerable CPU resources on the BIG-IP, thus must be accounted for in sizing an SSL Orchestrator opportunity.

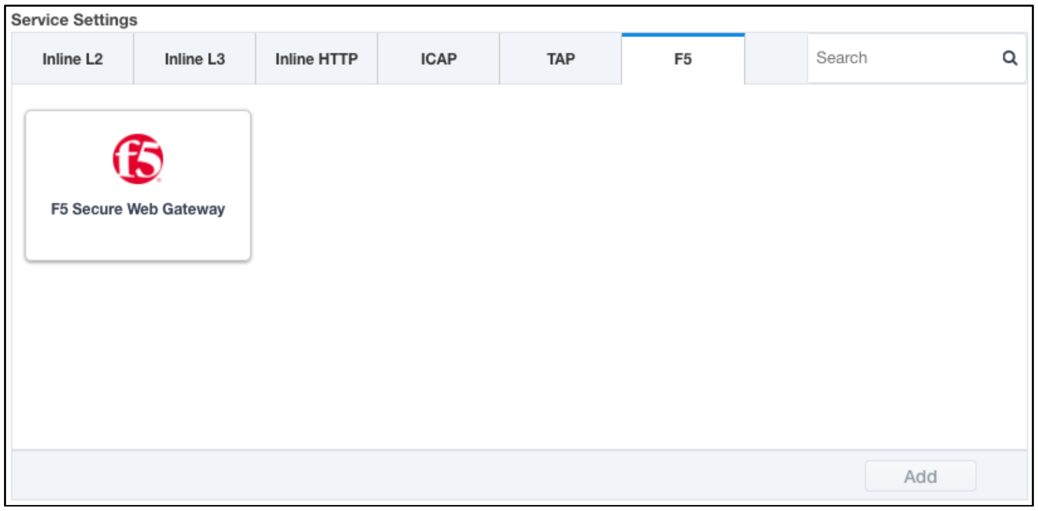
- SWG, like ICAP and HTTP proxy services, is intended for decrypted HTTP traffic. SSL Orchestrator will automatically skip any of these services for traffic that is not decrypted HTTP.

Given the above, the requirements for an SWG service are straightforward:

- SWG must be licensed and provisioned.

- To get SWG reporting, enable URL Request logging.

The F5 Secure Web Gateway tile will now be visible in the SSL Orchestrator service catalog.



To configure an SWG inline service:

- **Name**: enter a unique name for the SWG service.

- **Access Profile**: an access (per-session) profile will be generated automatically, or an existing profile can be selected here. Normally this will be a simple "start -> allow" policy, as the bulk of the SWG configuration is handled in the SWG per-request.

- **Access Profile Scope**: in the event that an SWG inline service is used in a topology that also combines explicit or transparent forward proxy authentication, the user session information collected in the authentication policy can be shared with the SWG policy by way of "named" scoping. Ensure that both the authentication policy and SWG policy have the same named scope. Select Named here to make the user session information available to the SWG policy.

- **Named Scope**: if the above setting is Named, enter a name string here that matches the named scope string in the associated topology authentication policy. Note also that an SWG service can be used with or without frontend authentication. With a frontend authenticating access policy, the SWG service will re-use the existing access session. Without a frontend authenticating access policy, the SWG service will create its own "per flow" session.

- **Per Request Policy**: the per-request policy is the heart of the SWG service. You may select an existing SWG policy here or Create New. The latter will open a new tab to the Access Per-Request Policies UI.

- **Service Action Down**: this option is controlled by the enabled status on the SWG virtual server. If the virtual server status is disabled, traffic arriving at this SWG service in the service chain can either be dropped, reset, or ignored, in which case the SWG service is skipped.

- **Log Settings**: to enable special logging setting for the SWG service, select an appropriate logging profile here.

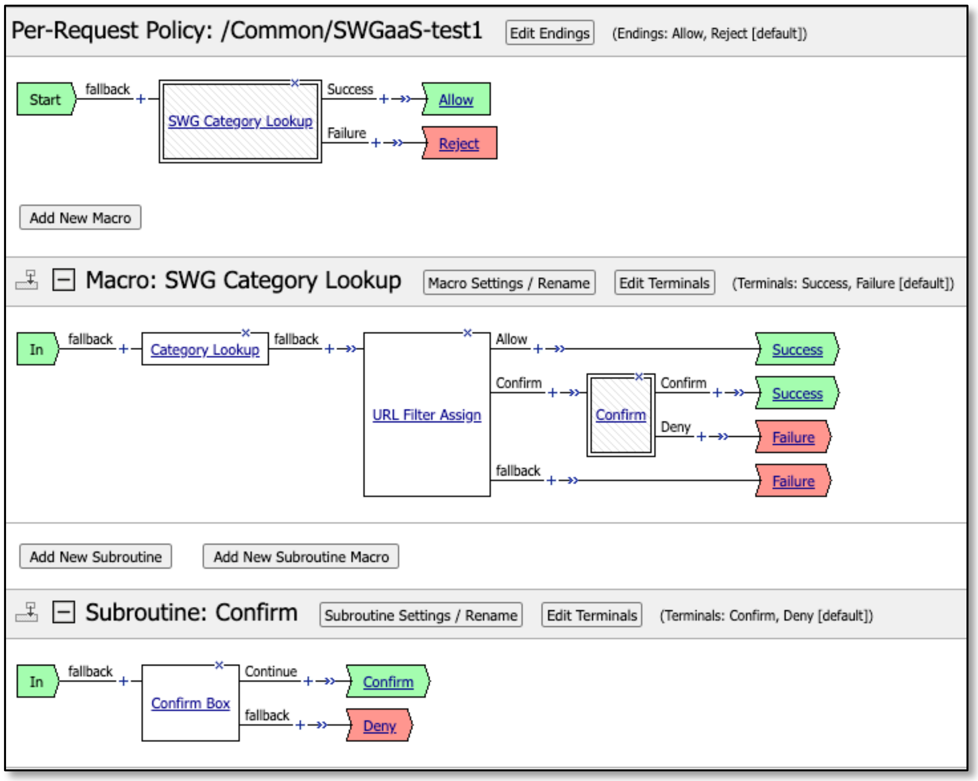
- **iRules**: as with other service types, you can inject additional iRule logic at the ingress of the SWG service traffic flow.

Creating an SWG per-request policy

SWG as an inline security service is intended to perform typical transactional web gateway functions. While it is possible to create SSL intercept/bypass and traffic flow actions in an SWG policy, these are not appropriate inside the SSL Orchestrator service chain. In the BIG-IP UI, under **Access -> Profiles / Policies -> Per Request Policies**, click the **Create** button.

- **Name**: enter a unique name.
- **Policy Type**: select All.
- **Incomplete Action**: select Allow.
- **Customization Type**: either type will work.
- **Languages**: select the appropriate language(s).

Once complete, edit the new SWG per-request policy VPE. You can begin adding functionality directly, or you can start with one of the pre-built macro templates. Click the **Add New Macro** button and select from one of the three templates and click **Save**. Now add that macro to the primary policy flow to enable it. Make any additional modifications as required.



For detailed instructions on creating SWG policies, please see the following resources:

https://techdocs.f5.com/en-us/bigip-15-1-0/big-ip-access-policy-manager-secure-web-gateway/per-request-policy-configuration-for-swg.html

https://techdocs.f5.com/kb/en-us/products/big-ip_apm/manuals/product/apm-secure-web-gateway-implementations-12-1-0/10.html

As stated previously, while most options are possible, some of these are not appropriate for an inline SWG service. The below is a list of SWG agents that are inappropriate for use in an SWGaaS per-request policy:

- Assignment
    - Pool Assignment
- General Purpose
    - IP Based SSL Bypass Set
    - Server Cert Response Control
    - Server Cert Status
    - SSL Check
    - SSL Intercept Set
    - SSL Configuration Select
- Traffic Management
    - Proxy Select
    - Service Connect
    - Session Check

Lab 16 – Implement an OCSP Responder

RFC 6960 describes the implementation of the Online Certificate Status Protocol (OCSP) as a method of providing certificate revocation status. Prior to or in lieu of OCSP, certificate revocation lists (CRLs) are generally the only other option to perform this task and are at times challenging to deploy given that CRLs are binary objects files that must be downloaded and parsed by the client and can sometimes be large. An OCSP transaction is represented as a small and discrete HTTP-wrapped request, minimally containing information about the issuer and a single certificate, and an equally small response containing the revocation status (good, revoked, or unknown) of that certificate. An OCSP server, referred to as a "responder", locally caches the binary CRLs and responds to client requests with that discrete status information. SSL Orchestrator can work with OCSP in multiple capacities. Most specifically, OCSP can be configured on the server side of an outbound topology to perform revocation checking of the real server certificates. When configured this way, SSL Orchestrator will relay OCSP "stapled" responses back to the client side. Starting in 9.0, SSL Orchestrator now supports a full client side OCSP service, in lieu of simple stapling in the TLS handshake. In this new method, SSLO can write an authorityInfoAccess (AIA) field in the forged server certificate. That AIA field contains a URL to an OCSP responder service listener (VIP) on the F5 that the client can query, and that will relay the real revocation status of the server certificate. Implementing this feature requires three steps:

Configure a DNS resolver

Server-side OCSP will require DNS if it needs to make a direct OCSP request. In the BIG-IP UI, under Network -> DNS Resolvers -> DNS Resolver List, click **Create**.

- **Name**: Provide a unique name.

Click **Finished**. Now edit this new DNS resolver and click on the Forward Zones tab. Click **Add**.

- **Name**: Enter "." (without the quotation marks)
- **Address**: Enter your local DNS resolver
- **Service Port**: Enter the local DNS resolver's listening port (almost always 53)

Click **Add** to add this resolver. Repeat for any additional resolvers.

Configure and enable server-side OCSP

Server-side OCSP is required here, as this is the information the client-side responder queries to generate an OCSP response. In the BIG-IP UI, under System -> Certificate Management -> Traffic Certificate Management -> OCSP, click **Create**. The below lists the minimum requirements.

- **Connection :: DNS Resolver**: Select the previously created DNS resolver.
- **Response Validation** :: Trusted Responders: This setting represents the CA bundle used to validate the signed OCSP response. In most cases the built-in ca-bundle.crt bundle will suffice.
- **Request Signing** :: Hash Algorithm: Select SHA1 to accommodate the majority of public responders.

On the SSL Configurations page of an outbound SSLO topology, select this new server-side OCSP profile under the **OCSP Certificate Validator** option.

Define and attach an OCSP Authentication profile

The OCSP Authentication profile creates three new objects:

- A new iRule attached to the topology interception rule virtual server that injects the AIA URL into the forged certificate. Example:

  ```
  when CLIENTSSL_CLIENTHELLO {
      SSL::forward_proxy extension AIA "ocsp,http://ocsp.f5labs.com:80"
  }
  ```

- An OCSP profile that defines the Max Age and Nonce settings.

- A virtual server that listens on the defined IP:port and includes the OCSP profile. Client OCSP requests to this VIP will query the assigned SSL profiles for the server-side revocation status, and relay that back in the client side OCSP response.

On the Authentication page in an outbound SSLO topology, click **Add**, select OCSP Responder and click the **Add** button.

- **FQDN**: Enter the fully qualified domain name of the local responder service. This will be the AIA URL value injected into the forged certificate and should resolve locally to the destination address defined below.

- **Source**: Enter 0.0.0.0%0/0 here to allow all clients to connect to the OCSP service, or filter as required.

- **Destination Address/Mask**: Enter an IP address here that will be locally accessible to internal clients.

- **Port**: Enter the OCSP service port number. OCSP is almost unencrypted, so port 80 is recommended.

- **VLANs**: Select the client facing VLAN(s).

- **SSL Configurations**: Select the same SSL configuration used by the outbound topology. This is the set of SSL profiles that the OCSP responder will query for server side OCSP revocation status.

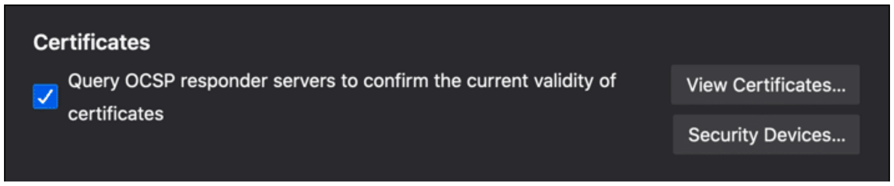
- **OCSP Profile**: Select Create New.

- **Max Age in Seconds**: This value represents the max age of the OCSP revocation status. The default 604800 seconds is 7 days.

- **Nonce**: A nonce is a random unique value that cryptographically binds a request and a response to prevent replay attacks. It is recommended to leave this enabled.

Testing client-side OCSP

Testing client-side OCSP requires a client that will perform this validation. Most modern browsers can do this, however because OCSP failures are still so common, most will "soft fail". Perhaps the easiest way to test is with a Firefox browser. Firefox's default settings query OCSP responders to confirm the validity of SSL/TLS certificates. (You can change this setting in Firefox's security preferences.)



Certificates

✓ Query OCSP responder servers to confirm the current validity of certificates

However, as Firefox (like other browsers) implements a "soft-fail" policy, If you wish to require strict OCSP checking, navigate to **about:config** and toggle **security.ocsp.require** to true.

For a more verbose check you can use the following script on a Linux-based client:

- Copy the script to your client.
- Copy the local issuer CA certificate to the same directory
- Make the script executable: `chmod +x [script]`
- Execute the script and provide the destination URL and issuer certificate

Example: `./test-ocsp.sh www.example.com subrsa.f5labs.com.cer`

```
#!/bin/bash

site=$1
issuer=$2

## get certificate
echo | openssl s_client -connect ${site}:443 2>&1 |sed -n '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > temp.crt

## get ocsp
ocsp=$(openssl x509 -noout -text -in temp.crt |grep OCSP |awk -F":" '{print $2,$3,$4}'| while read var1 var2 var3; do echo
"$var1:$var2:$var3"; done)

## issue local ocsp request
openssl ocsp -issuer ${issuer} -cert temp.crt -text -url ${ocsp} -noverify

## cleanup
rm -f temp.crt
```

You should see something like the following output:

```
…
temp.crt: good
    This Update: Apr  5 10:27:01 2021 GMT
    Next Update: Apr 12 09:42:01 2021 GMT
```

Lab 17 – Implement Multi-SNI Switching

Introduced in SSL Orchestrator 9.1, multi-SNI switching allows a single topology to contain multiple client SSL profiles, where the correct profile is selected based on incoming Client Hello Server Name Indication (SNI) extension. In a forward proxy scenario, this would often be used to select specific TLS behaviors (i.e., ciphers), or perhaps a different CA issuer based on the SNI. In a reverse proxy scenario, multi-SNI is used to select a different server certificate to present to the client. This would most often be used in an inbound gateway mode architecture.

Implementing multi-SNI simply requires attaching multiple SSL configurations to a topology:

Configure a new SSL configuration

In the SSL Orchestrator UI, navigate to the SSL Configurations tab and click **Create**. Notice that in SSL Orchestrator 9.1, a new **SNI Server Name (FQDN)** option exists. This setting mirrors the Server Name field in an LTM client SSL profile. When using this feature, there should always be one SSL configuration that is "default" – that is, the SSL configuration that gets used when there is no SNI match. For this default configuration, also select the "Default SNI" option. For all other SSL configurations, just enter the SNI value to match on. Make any other changes to the SSL configuration as required and deploy.

Attach multiple SSL configurations to a topology

To attach the additional SSL configurations to a topology, you can either edit the complete topology itself, or directly edit the respective interception rule from the Interception Rules tab. When editing from the topology workflow, Interception Rules step, move the SSL configurations into the Selected box under Protocol Settings: SSL Configurations. When editing directly from the Interception Rules workflow, you can independently select the client and server SSL profiles.

Appendix – Additional 9.0 Updates

Feature: TCP Keepalive pass-through support

This feature is implemented as a new iRule command to enable variable TCP keepalive times for dynamic traffic flows. To enable TCP keepalive, insert an iRule on the Interception Rule using the new `TCP::keepalive` command. The following example enables a specific keepalive time value for traffic that matches a source IP address data group:

```
when CLIENT_ACCEPTED {
    set ka 0
    if [class match [IP::client_addr] equals "keepalive_clients_dg"] } {
        TCP::keepalive 5
        set ka 1
    }
}
when SERVER_CONNECTED {
    if { ${ka} } {
        TCP::keepalive 5
    }
}
```

APPENDIX – CCMODE STIP UPDATES

The "SSL/TLS Intercept Proxy" (STIP) evaluation and is part of a larger US Federal NIAP (National Information Assurance Partnership) certification. F5 intends to achieve NIAP certification through a collection of enhancements per the standard to the SSL Orchestrator solution. As a result, F5 customers will achieve a higher level of assurance that the product meets the industry best practices in the SSL visibility solution space. STIP is an extension to the existing Common Criteria Mode (ccmode) function and adds a number of capabilities to SSL Orchestrator.

Common Criteria (ccmode) is an irreversible option for securing the BIG-IP per Common Criteria certification standards. Please refer to official documentation for more information on ccmode. To enable it:

```
tmsh ccmode
```

To then enable Common Criteria STIP mode:

```
tmsh modify sys db security.commoncriteria { value "true" }
tmsh modify sys db security.commoncriteria.stip { value "true" }
tmsh modify sys db ssl.forwardproxy.inspectionconsent { value "yes" }
```

If forged certificate logging is required, also enable the following:

```
tmsh modify sys db tmm.ssl.loggingcreatedcerts { value "enable" }
```

SSL Session Logging

(STIP ID# FAU_SAR.3) With STIP mode enabled, the SSL Configurations page now includes a "Server Log Publisher" option that defaults to using the same client log publisher. To change this, uncheck the option, then select a different log publisher as required. This translates to a new Log Publisher option in the respective SSL profiles. With the logging above enabled, the ltm log (`/var/log/ltm`) will generate a new set of SSL log messages. For example:

```
[Server-side handshake details]
SSL Handshake details for TCP 93.184.216.34:443 -> 10.1.20.100:59044 entity: client SID:
df7083a19fbec0dd5ab3f5693e0c0909e942f53782ec07810dbf66a0f0e8d4b2 version: TLSv1.2 cipher-suite: ECDHE-RSA-AES128-GCM-SHA256
key-exchange: 70 bytes client-cert-sha1: N/A server-cert-sha1: 0a:28:a6:eb:17:6e:a9:cc:59:6f:4c:73:fd:89:7e:fb:d3:2d:ca:2a
mutual-authentication: false

[Forged certificate information]
SSL certificate forgery succeeded from server cert for TCP 10.1.10.50:59044 -> 93.184.216.34:443 entity: server SID:
32e83402862ca8e61ab05c0bee75506d4a15230710d15fe39db7497bbe05a07f original-cert-sha1:
0a:28:a6:eb:17:6e:a9:cc:59:6f:4c:73:fd:89:7e:fb:d3:2d:ca:2a original-cert-dn: /C=US/ST=California/L=Los Angeles/O=Internet
Corporation for Assigned Names and Numbers/CN=www.example.org

[Client-side handshake details]
SSL Handshake details for TCP 10.1.10.50:59044 -> 93.184.216.34:443 entity: server SID:
32e83402862ca8e61ab05c0bee75506d4a15230710d15fe39db7497bbe05a07f version: TLSv1.2 cipher-suite: ECDHE-RSA-AES128-GCM-SHA256
key-exchange: 333 bytes client-cert-sha1: N/A server-cert-sha1: 69:46:e8:85:a1:ad:a7:5e:e7:6c:0b:46:a4:60:07:55:64:db:b9:3d
mutual-authentication: false
```

APPENDIX – TROUBLESHOOT SSL ORCHESTRATOR

While the SSL Orchestrator product has certainly evolved, as with anything in the computing world, problems are usually inevitable and poorly timed. In the event that an SSL Orchestrator configuration has failed, or that it has succeeded but not behaving as expected, the following troubleshooting tools should be useful.

Step 1: Test the configuration

It is important to first define "normal" behavior. If the SSL Orchestrator deployment process was successful, it will be possible to access remote Internet sites from the client workstation without issue, and HTTPS sites appear to have a locally trusted, re-issued server certificate. This would be considered normal behavior. If any of these do not happen, use the tools below to troubleshoot.

Step 2: Troubleshoot

Below is a reasonably-ordered list of troubleshooting steps.

- If the SSL Orchestrator deployment process fails, review the ensuing error message. It would be impossible to list here all of the possible error messages and their meanings, but often enough the messages will reveal the issue.

- Re-review the lab steps for any missing or misconfigured settings.

- Enable debug logging in the SSL Orchestrator configuration. Tail the APM log from a BIG-IP command line or from the logs page in the management UI. Debug logging will very often reveal important issues. Specifically, it will indicate traffic classification matches, mismatches or deployment issues.

  ```
  tail –f /var/log/apm
  tail -f /var/log/restnoded/restnoded.log
  tail -f /var/log/restjavad.0.log
  ```

- If the SSL Orchestrator deployment process succeeds, but traffic isn't flowing through the environment made evident by lack of access to remote sites from the client:

  - Ensure that the client is properly configured to either default route to the ingress VLAN and self-IP of the BIG-IP for transparent proxy access or has the correct browser proxy settings defined for explicit proxy access.

  - Ensure that traffic is flowing to the BIG-IP from the client with a tcpdump capture at the ingress interface.

  - Review the LTM configuration created by the SSL Orchestrator. Specifically, look at the pools and respective monitors for any failures.

  - Isolate service chain services. If at least one service chain has been created, and debug logging indicates that traffic is matching this chain, remove all but one service from that chain and test. Add one service back at a time until traffic flow stops. If a single added service breaks traffic flow, this service will typically be the culprit.

- If a broken service is identified, insert probes to verify inbound and outbound traffic flow. Inline services will have a source (S) VLAN and destination (D) VLAN, and ICAP and receive only services will each have a single source VLAN. Insert a tcpdump capture at each VLAN in order to determine if traffic is getting to the device, and if traffic is leaving the device through its outbound interface.

- If no service chains are defined, it may be necessary to remove all of the defined services and re-create them one-by-one to validate flow through the built-in All chain. If a broken service is identified, insert tcpdump probes as described above.

- If traffic is flowing through all of the security devices, insert a tcpdump probe at the egress point to verify traffic is leaving the BIG-IP to the gateway router.

- If traffic is flowing to the gateway router, perform a more extensive packet analysis to determine if SSL is failing between the BIG-IP egress point and the remote server.

  ```
  tcpdump –i 0.0:nnn –nn –Xs0 –vv –w <file.pcap> <any additional filters>
  ```

  Then either export this capture to WireShark or send to ssldump:

  ```
  ssldump –nr <file.pcap> -H –S crypto > text-file.txt
  ```

- If the WireShark or ssldump analysis verifies an SSL issue:

  - Plug the site's address into the SSLLabs.com server test site at:
    https://www.ssllabs.com/ssltest/
    This report will indicate any specific SSL requirements that this site has.

  - Verify that the SSL Orchestrator server SSL profiles (two of them) have the correct cipher string to match the requirements of this site. To do that, perform the following command at the BIG-IP command line:

    ```
    tmm --clientciphers 'CIPHER STRING AS DISPLAYED IN SERVER SSL PROFILES'
    ```

  - Further SSL/TLS issues are beyond the depth of this lab guide. Seek assistance.

- If all else fails, seek assistance.

APPENDIX – COMMON TESTING COMMANDS

The following are some simple, but powerful commands that are useful in developing and troubleshooting SSL visibility projects.

Packet capture

Second only to debug logging, packet captures are crucial to troubleshooting (and simply validating) network and service-related issues. Each security service is connected to separate "ingress" and "egress" VLANs, traffic going into the service from the F5, and traffic leaving the service back to the F5, respectively. To verify that traffic is entering, or leaving a security device, insert a tcpdump "tap" on the appropriate VLAN.

```
tcpdump –lnni [VLAN]:nnn -Xs0
```

Note that the service VLANs reside with application service containers, so must be referenced accordingly. The easiest way to derive this path is from the BIG-IP UI. Navigate to the Network – VLANs menu. In the "Partition / Path" column for the desired VLAN, copy the path beyond the "Common/" string. For example, if the path is "*Common/ssloN_IPS_in.app*", copy "*ssloN_IPS_in.app*". The full path will be this value, plus the same string again without the "*.app*" extension. The VLAN path will therefore look like this:

ssloN_IPS_in.app/ssloN_IPS_in

From the BIG-IP command line, insert the tcpdump tap on the ingress side of this IPS service like this:

```
tcpdump -lnni ssloN_IPS_in.app/ssloN_IPS_in
```

Pass traffic through SSLO to verify that data is flowing to the inline service. Switch the VLAN value to the egress VLAN to also verify data is flowing out of the inline service. To view decrypted traffic, add "-Xs0" (zero) to the tcpdump command.

```
tcpdump -lnni ssloN_IPS_in.app/ssloN_IPS_in -Xs0
```

And to filter out ICMP traffic and other unneeded flows, add filters to the end of the capture.

```
tcpdump -lnni ssloN_IPS_in.app/ssloN_IPS_in not icmp
```

Control the SSLFWD certificate cache

The behavior of the SSL Forward Proxy changes after a certificate is cached, which will make it difficult to troubleshoot some issues. The following allows you to both list and delete the certificates in the cache.

```
tmsh show ltm clientssl-proxy cached-certs clientssl-profile [CLIENTSSL PROFILE] virtual [INGRESS TCP VIP]
tmsh delete ltm clientssl-proxy cached-certs clientssl-profile [CLIENTSSL PROFILE] virtual [INGRESS TCP VIP]
```

Isolate SSLO traffic

Any given website will be full of images, scripts, style sheets, and very often references to document objects on other sites (like a CDN). This can make troubleshooting very complex. The following cURL commands allow you to isolate traffic to a single request and response.

```
curl –vk https://www.bing.com

curl –vk --proxy 10.30.0.150:3128 https://www.bing.com

curl –vk --proxy 10.30.0.150:3128 --location https://www.bing.com
```

Optionally, between each cURL test, delete the certificate cache and start logging:

```
tmsh delete ltm clientssl-proxy cached-certs clientssl-profile [CLIENTSSL PROFILE] virtual
[INGRESS TCP VIP] && tail –f /var/log/apm
```

Debugging

There is simply nothing better than debug logging for troubleshooting SSL intercept issues. The SSL Orchestrator in debug mode pumps out an enormous set of logs, describing every step along a connection’s path. Remember to never leave debug logging enabled.

```
tail –f /var/log/apm
```

SSL inspection

TLS is rarely the issue, but a sight or configuration error may render some sites inaccessible.

```
ssldump –AdNd –i [VLAN] port 443 <and additional filters>
tcpdump –i 0.0:nnn –nn –Xs0 –vv –w <file.pcap> <and additional filters>
ssldump –nr <file.pcap> -H –S crypto > text-file.txt
```

Control the URL Filtering database

To show the current status of the database:

```
tmsh list sys url-db download-result
```

To initiate (force) the URL DB to update:

```
tmsh modify sys url-db download-schedule all status true download-now true
```

To verify that the URL DB is actively updating:

```
tcpdump  -lnni 0.0 port 80 and host 204.15.67.80
```

External testing

Plug the site’s address into SSLLabs.com server test site at **https://www.ssllabs.com/ssltest/** to see if the site has any unusual SSL/TLS requirements.

APPENDIX – DEMO SCRIPTS

Lab 1 demo script

Configuration review and prerequisites
1. Optionally define DNS, NTP and gateway route
2. Click Next

Topology Properties
1. Name - some name
2. Protocol: Any
3. IP Family: IPv4
4. Topology: L3 Outbound
5. Click Save & Next

SSL Configuration
1. Create a New SSL Profile
2. Client-side SSL (Cipher Type): Cipher String
3. Client-side SSL (Cipher String): DEFAULT
4. Client-side SSL (Certificate Key Chain): default.crt and default.key
5. Client-side SSL (CA Certificate Key Chain): subrsa.f5labs.com
6. Server-side SSL (Cipher Type): Cipher String
7. Server-side SSL (Cipher String): DEFAULT
8. Server-side SSL (Trusted Certificate Authority): ca-bundle.crt
9. Click Save & Next

Service List
1. **Inline Layer 2 service**
   a. Name: some name (ex. FireEye)
   b. Network Configuration
      - Ratio: 1
      - From BIGIP VLAN: Create New, name (ex. FireEye_in), int 1.4
      - To BIGIP VLAN: Create New, name (ex. FireEye_out), int 1.5
      - Click Done
   c. Service Action Down: Ignore
   d. Enable Port Remap: Enable, 8080
   3. Click Save
2. **Inline layer 3 service**
   a. Name: some name (ex. IPS)
   b. IP Family: IPv4
   c. Auto Manage: Enabled
   d. To Service Configuration
      - To Service: 198.19.64.7/25
      - VLAN: Create New, name (ex. IPS_in), interface 1.3, tag 60
   e. Service Action Down: Ignore
   f. L3 Devices: 198.19.64.30
   g. From Service Configuration
      - From Service: 198.19.64.245/25
      - VLAN: Create New, name (ex. IP_out), interface 1.3, tag 70
   h. Enable Port Remap: Enabled, 8181
   i. Manage SNAT Settings: None
   j. Click Save
3. **Inline HTTP service**
   a. Name: some name (ex. Proxy)
   b. IP Family: IPv4
   c. Auto Manage: Enabled
   d. Proxy Type: Explicit
   e. To Service Configuration
      - To Service: 198.19.96.7/25
      - VLAN: Create New, name (ex. Proxy_in), interface 1.3, tag 30
   f. Service Action Down: Ignore
   g. HTTP Proxy Devices: 198.19.96.30, Port 3128
   h. From Service Configuration
      - From Service: 198.19.96.245/25
      - VLAN: Create New, name (ex. Proxy_out), interface 1.3, tag 40
   i. Manage SNAT Settings: None
   j . Authentication Offload: Disabled
   k. Click Save
4. **ICAP Service**
   a. name: some name (ex. DLP)
   b. IP Family: IPv4
   c. ICAP Devices: 198.19.97.50, Port 1344
   d. Request URI Path: /avscan
   e. Response URI Path: /avscan
   f. Preview Max Length(bytes): 524288
   g. Service Action Down: Ignore
   h. Click Save
5. **TAP Service**
   a. Some Name (ex. TAP)
   b. Mac Address: 12:12:12:12:12:12
   c. VLAN: Create New, name (ex. TAP_in)
   d. Interface: 1.6
   e. Service Action Down: Ignore
   f. Click Save
6. Click Save & Next

Service Chain List
1. Add
   a. Name: some name (ex. all_services)
   b. Services: all of the services
   c. Click Save
2. Add
   a. name: some name (ex. layer2_tap_services)
   b. Services: L2 and TAP services
   c. Click Save
3. Click Save & Next

Security Policy
1. Add a new rule
   a. Name: some name (ex. urlf_bypass)
   b. Conditions
      - Category Lookup (All)
      - SNI Category: Financial Data and Services, Health and Medicine
   c. Action: Allow
   d. SSL Forward Proxy Action: bypass
   e. Service Chain: L2/TAP service chain
   f. Click OK
2. Modify the All rule
   a. Service Chain: all services chain
   b. Click OK
3. Click Save & Next

Interception Rule
   a. Select Outbound Rule Type: Default
   b. Ingress Network (VLANs): client-side
   c. L7 Interception Rules: apply FTP and email protocols as required
   d. Click Save & Next

Egress Setting
   a. Manage SNAT Settings: Auto Map
   b. Gateways: New, ratio 1, 10.1.20.1

Summary
   a. Review configuration
   b. Click Deploy

Lab 2 demo script

Configuration review and prerequisites

1. Optionally define DNS, NTP and gateway route
2. Click Next

Topology Properties

1. Name: some name (ex. sslo-inbound-1)
2. Protocol: TCP
3. IP Family: IPv4
4. Topology: L3 Inbound
5. Click Save & Next

SSL Configuration

1. Show Advanced Setting
2. Client-side SSL (Cipher Type): Cipher String
3. Client-side SSL (Cipher String): DEFAULT
4. Client-side SSL (Certificate Key Chain): wildcard.f5labs.crt and wildcard.f5labs.com.key
5. Server-side SSL (Cipher Type): Cipher String
6. Server-side SSL (Cipher String): DEFAULT
7. Server-side SSL (Trusted Certificate Authority): ca-bundle.crt
8. Advanced (Expire Certificate Control): Ignore
9. Advanced (Untrusted Certificate Authority): Ignore
10. Click Save & Next

Services List

1. Click Save & Next

Service Chain List

1. Click Save & Next

Security Policy

1. Remove Pinners_Rule
2. Edit All Traffic rule and add L2/TAP service chain
3. Click Save & Next

Interception Rule

1. Gateway mode
   a. Hide Advanced Setting
   b. Source Address: 0.0.0.0/0
   c. Destination Address/Mask: 0.0.0.0/0
   d. Port: 443
   e. VLANs: outbound
2. Application mode
   a. Show Advanced Setting
   b. Source Address: 0.0.0.0/0
   c. Destination Address: 10.1.20.120
   d. Port: 443
   e. VLANs: outbound
   f. Pool: webserver-pool
3. Click Save & Next

Egress Settings

1. Manage SNAT Settings: Auto Map
2. Gateways: Default Route

Summary

1. Review configuration
2. Click Deploy

Lab 3 demo script

Configuration review and prerequisites

1. Optionally define DNS, NTP and gateway route
2. Click Next

Topology Properties

1. Name: some name (ex. sslo-explicit)
2. Protocol: TCP
3. IP Family: IPv4
4. Topology: L3 Explicit Proxy
5. Click Save & Next

SSL Configuration

1. SSL Profile: Use Existing, existing outbound SSL settings
2. Click Save & Next

Services List

1. Click Save & Next

Service Chain List

1. Click Save & Next

Security Policy

1. Type: Use Existing, existing outbound security policy
2. Click Save & Next

Interception Rule

1. IPV4 Address: 10.1.10.150
2. Port: 3128
3. VLANs: client-net
4. Click Save & Next

Egress Settings

1. Manage SNAT Settings: Auto Map
2. Gateways: Existing Gateway Pool, -ex-pool-4 pool

Summary

1. Review configuration
2. Click Deploy

System Settings

1. DNS Query Resolution: Local Forwarding Nameserver
2. Local Forwarding Nameserver(s): 10.1.20.1
3. Click Deploy