Autor: Ivñan Matarrubias	Asunto: VIVAit Call - SSO Web Call
Revisado: Alfredo Rodríguez	Fecha: Junio 2026

VIVAit Call – SSO para Web Call

Documentación Técnica

Fecha : Junio 2026	**Número de revisión**: Versión 1
Objeto del documento : Inicio de sesión único (SSO) del portal Web Call de VIVAit Call mediante OAuth2/OIDC y SAML2, con HAProxy, pool Apache, Microsoft Entra ID y serCen.	
Actores (empresas): • **MDTEL**	

Contenido

1. Introducción

Este documento describe el **inicio de sesión único (SSO)** del portal **Web Call** (teléfono webfon, también referido como Webfon2) de **VIVAit Call**, incluso cuando el acceso se publica a través de un balanceador **HAProxy** que reparte la carga entre un pool de servidores **Apache**.

El objetivo del proceso es delegar la autenticación de los usuarios en **Microsoft Entra ID** (IdP corporativo) y, una vez autenticado el usuario, obtener los **tokens internos** (`token` y `token2`) que el portal necesita. Esos tokens internos los emite el proceso **serCen** (autenticación y doble factor de VIVAit Call), al que **Apache** llama localmente.

Se soportan **dos protocolos de federación**, mutuamente excluyentes por sitio Apache:

- **OAuth2 / OIDC** mediante el módulo `mod_auth_openidc`. Tiene un canal posterior (back-channel) en el que **Apache** canjea el `code` por tokens contra el endpoint de Microsoft.
- **SAML2** mediante el módulo `mod_auth_mellon`. La validación de la respuesta es 100% local: `mod_auth_mellon` verifica la firma del IdP con los certificados en disco, sin back-channel en esa fase.

HAProxy

Esta versión de documento tiene en cuenta la posible incusión de **HAProxy** (alta disponibilidad y balanceo): al existir más de un nodo **Apache**, aparecen dos requisitos críticos que no existen en un despliegue mononodo:

1. **Afinidad por IP de origen (`balance source`)**: el flujo de autenticación (state/nonce en OAuth2, o el inicio del AuthRequest en SAML2) queda ligado al **Apache** que atendió la primera petición. HAProxy garantiza esa afinidad balanceando por hash consistente de la IP de origen (`balance source` + `hash-type consistent`), sin necesidad de sticky session ni stick-table. Si el callback del IdP cayese en otro nodo, el flujo fallaría (`invalid_state` en OAuth2, error de validación en SAML2).
2. **IP real del cliente**: con HAProxy delante, **Apache** y **fail2ban** deben ver la IP de origen real (vía `X-Forwarded-For`) y no la del balanceador, para que el baneo de IPs y los logs sean correctos.

Actores y sistemas implicados

Navegador del usuario, HAProxy, pool de **Apache** (servidor de páginas / proxy inverso de VIVAit Call), **Microsoft Entra ID** (IdP), **serCen** (emisión de tokens internos) y los servicios protegidos publicados por Apache (**Web Call**, **Vivait-FonBO**, **Janus**). La seguridad perimetral de cada nodo la aportan **nftables** (firewall) y **fail2ban**.

Restricciones conocidas

- La afinidad se basa en la **IP de origen** del cliente (`balance source` con hash consistente). Si la IP del cliente cambia entre la petición inicial y el callback (por ejemplo, salida NAT con varias IPs públicas), el cliente podría aterrizar en otro Apache y la autenticación fallaría.
- El endpoint interno `/sercen/getautenticacionapache` **nunca** debe ser accesible desde el exterior (está explícitamente denegado en Apache).

- Las credenciales sensibles (`OIDCClientSecret`, `OIDCCryptoPassphrase`, claves SAML) aparecen en los ficheros de configuración y deben protegerse y rotarse según la política de la instalación.

2. Arquitectura

2.1 Diagrama general

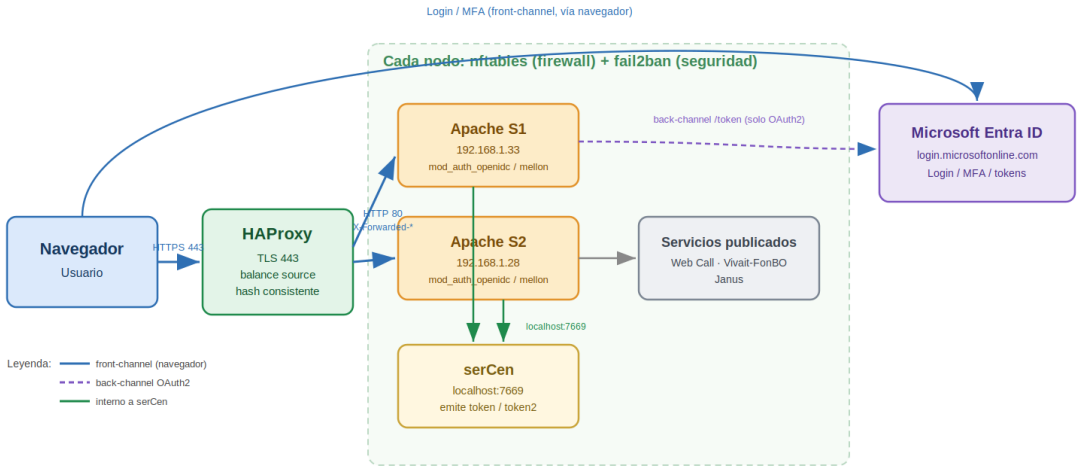

Ilustración 1 Arquitectura: navegador → HAProxy → pool Apache, con serCen, servicios y MS Entra ID.

Notas del diagrama:

- HAProxy (**opcional**) termina TLS en el puerto 443 y reenvía a los Apache por HTTP en el puerto 80, añadiendo las cabeceras `X-Forwarded-*`.
- El back-channel contra Microsoft Entra ID (línea discontinua) solo existe en **OAuth2** (canje del `code` por tokens). En **SAML2** la validación es local con certificados en disco.
- **serCen** se invoca siempre en local (`localhost:7669`) desde el mismo Apache que mantiene la sesión SSO.

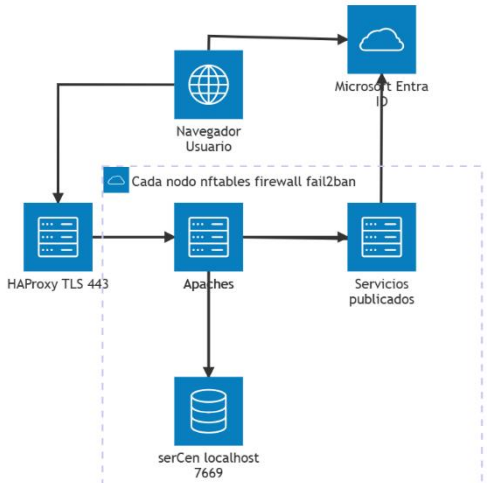

Ilustración 2.- Arquitectura general

2.2 Flujo OAuth2 / OIDC

El flujo OAuth2 se desarrolla en tres fases (**HA Proxy opcional**).

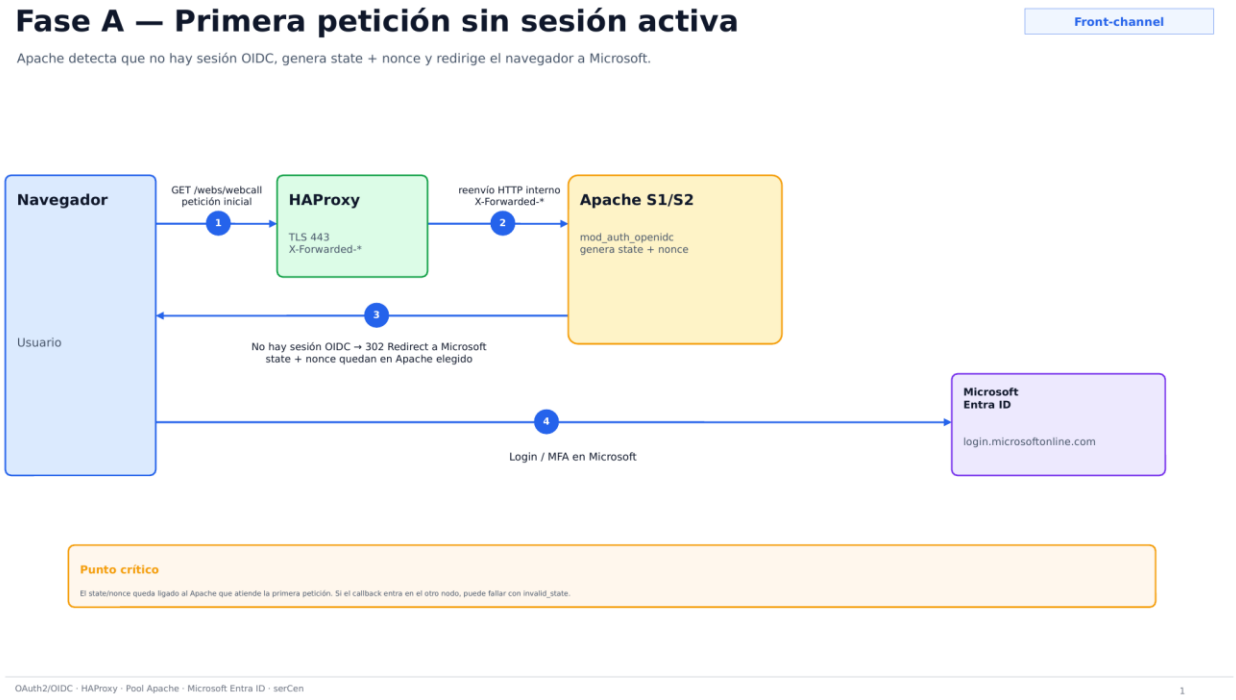

Ilustración 3. OAuth2 — Fase A: primera petición sin sesión activa.

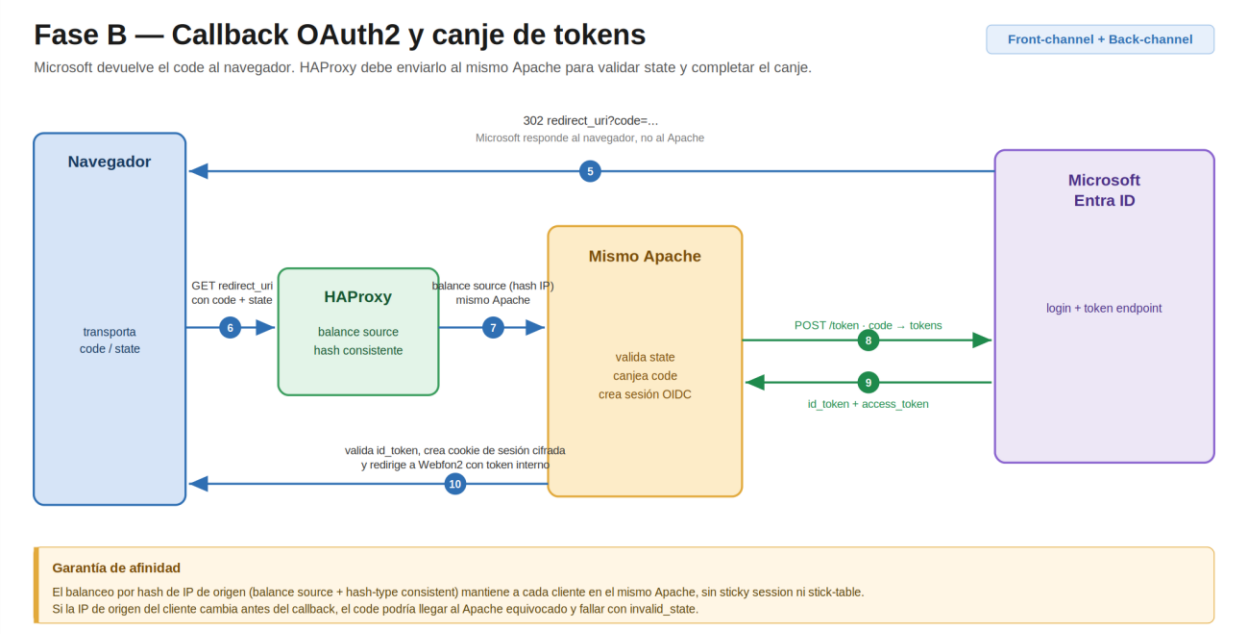

Ilustración 4. OAuth2 — Fase B: callback y canje de tokens (back-channel).

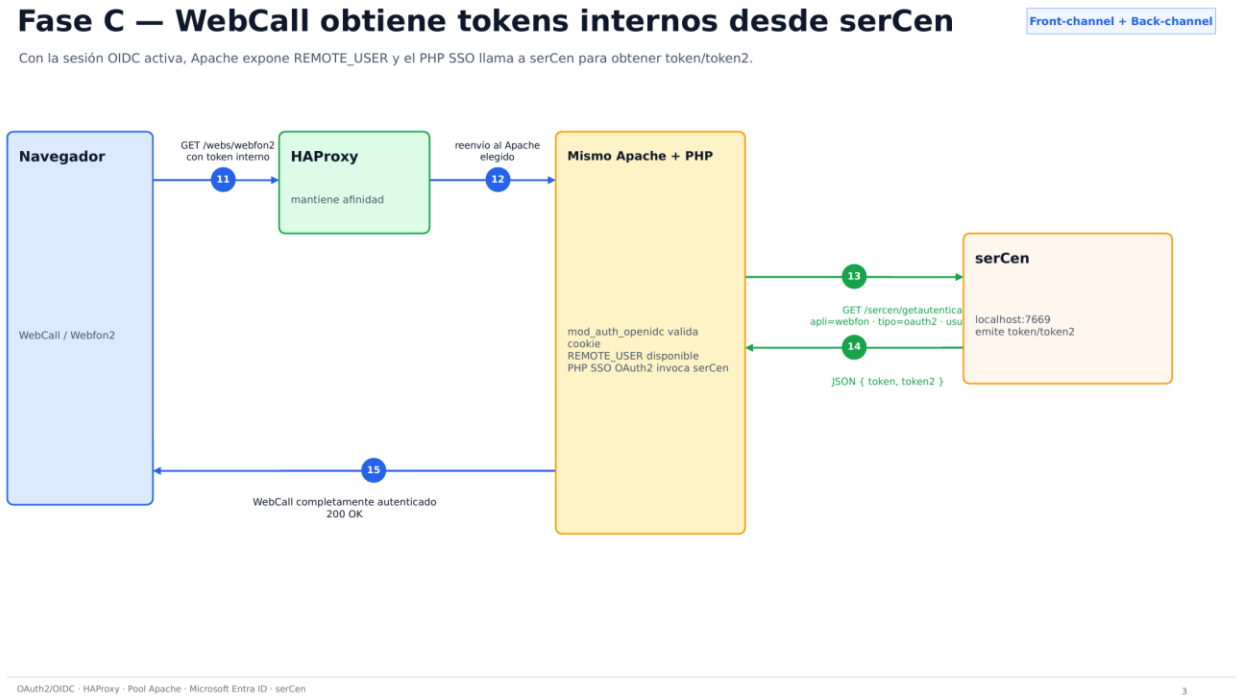

Ilustración 5. OAuth2 — Fase C: WebCall obtiene los tokens internos desde serCen.

> **Punto crítico (OAuth2)**
>
> El state/nonce queda ligado al Apache que atendió la primera petición. Si el callback entra en el otro nodo, falla con invalid_state. El balanceo por hash consistente de IP de origen (balance source + hash-type consistent) es lo que garantiza que el cliente vuelva siempre al mismo Apache.

2.3 Flujo SAML2

El flujo SAML2 se desarrolla también en tres fases (**HAProxy opcional**).

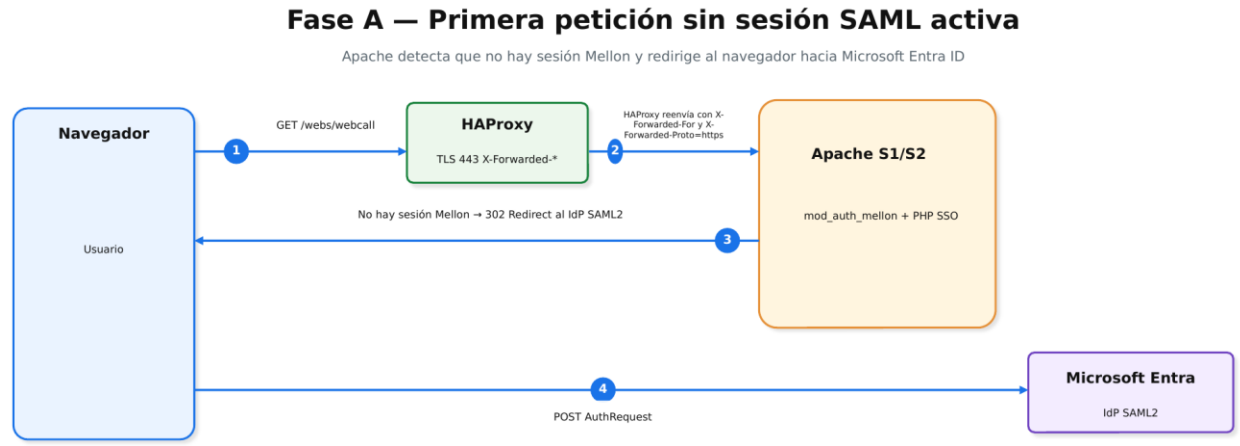

Ilustración 6. SAML2 — Fase A: primera petición sin sesión Mellon.

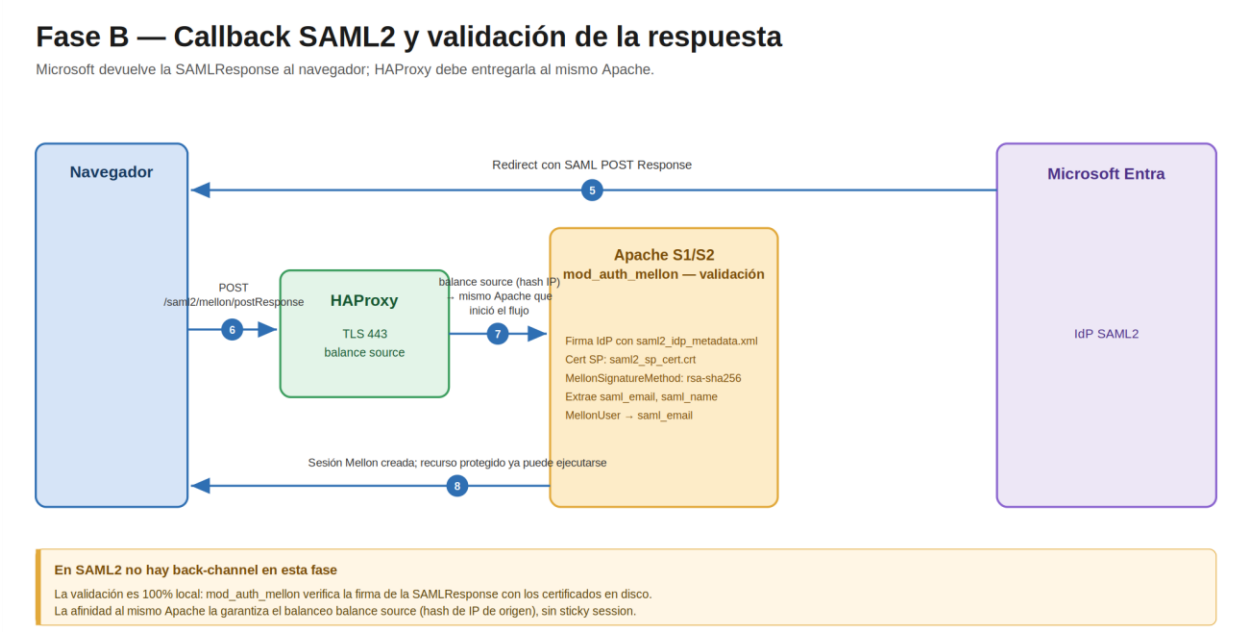

Ilustración 7. SAML2 — Fase B: callback y validación local de la SAMLResponse.

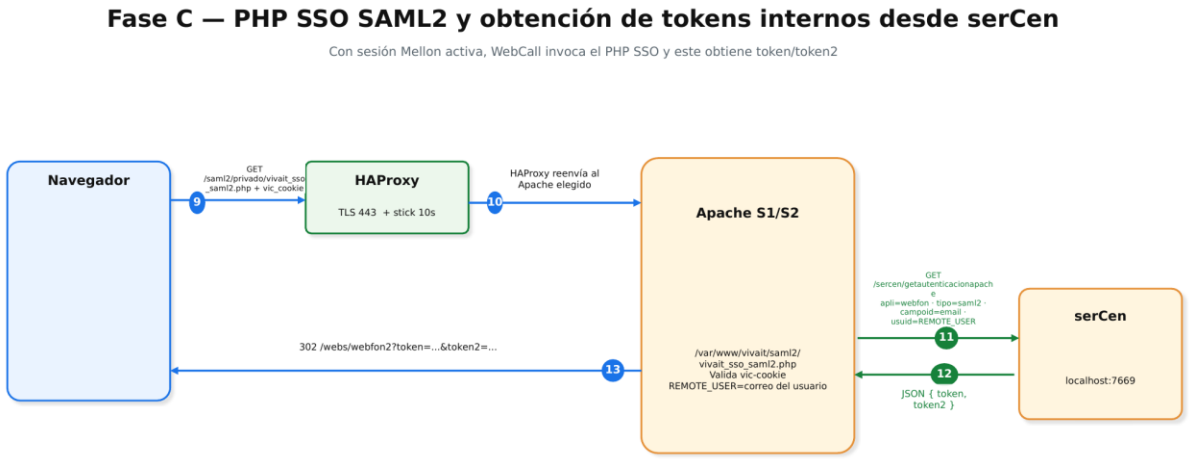

Ilustración 8. SAML2 — Fase C: PHP SSO obtiene los tokens internos desde serCen.

Diferencia clave frente a OAuth2

En SAML2 NO hay back-channel en la fase de callback. mod_auth_mellon valida la firma de la SAMLResponse localmente, usando saml2_idp_metadata.xml (firma del IdP) y los certificados del SP (saml2_sp_cert.crt / .key) en disco. El back-channel solo aparece en la Fase C, contra serCen en local.

2.4 Componentes principales

Componente	Función
HAProxy (**OPCIONAL**)	Termina TLS (443), balancea por IP de origen con hash consistente y mantiene afinidad hacia el mismo Apache durante el flujo de autenticación.
Apache (pool S1/S2)	Servidor de páginas y proxy inverso de VIVAit Call. Ejecuta el módulo SSO (mod_auth_openidc o mod_auth_mellon), publica los servicios y llama a serCen.
mod_auth_openidc	Cliente OAuth2/OIDC: genera state/nonce, canjea el code, valida id_token y crea la cookie de sesión cifrada.
mod_auth_mellon	SP SAML2: emite el AuthRequest, valida la firma de la SAMLResponse y crea la sesión Mellon.
Microsoft Entra ID	Proveedor de identidad (IdP) corporativo. Realiza login y MFA.
serCen	Proceso de autenticación de VIVAit Call (localhost:7669). Emite los tokens internos token/token2 a partir del usuario autenticado (REMOTE_USER).
Web Call	Destino final del flujo de autenticación.
Vivait-FonBO	Proporciona información de agenda e histórico de llamadas (servicio protegido publicado por Apache).
Janus	Servidor WebRTC, publicado por Apache como proxy.
nftables	Firewall del nodo. Controla el acceso a 443, 80, 3306, RTP, etc.
fail2ban	Seguridad del nodo. Monitoriza logs y banea IPs de origen ante intentos abusivos.

2.5 Integraciones y dependencias

- **Externa:** Microsoft Entra ID (`login.microsoftonline.com`). Requiere salida HTTPS desde los nodos Apache (solo OAuth2) y desde el navegador del usuario.
- **Interna: serCen** debe estar operativo en `localhost:7669` en cada nodo Apache. Sin él, la Fase C no entrega `token`/`token2`.
- **Infraestructura:** HAProxy debe poder alcanzar a los Apache del pool por HTTP/80; los Apache deben publicar sus servicios internos (Janus 8088, serCen 7669, Tomcat/Vivait-FonBO 8180).

3. Instalación

3.1 Requisitos previos

- **Sistema operativo:** Debian.
- **HAProxy** instalado en el nodo balanceador, con el certificado TLS del FQDN público.
- **Apache** en cada nodo del pool, con `remoteip` habilitado y `ssl` deshabilitado (el TLS lo termina HAProxy).
- OAuth2: módulo `mod_auth_openidc`. SAML2: paquete `libapache2-mod-auth-mellon`.
- **serCen** operativo y escuchando en `localhost:7669` en cada nodo Apache.
- Certificado TLS **común** entre HAProxy y los nodos (el FQDN publicado es el mismo).

3.2 Instalación de HAProxy

```
apt update
apt install haproxy

# Colocar el certificado combinado (cert + clave) del FQDN público
cp <fqdn>.pem /etc/haproxy/certs/<fqdn>.pem
chmod 600 /etc/haproxy/certs/<fqdn>.pem

# Editar /etc/haproxy/haproxy.cfg (ver sección 4.1)
systemctl enable --now haproxy
```

3.3 Instalación de Apache para OAuth2

```
# Módulos necesarios
a2dismod ssl
a2enmod remoteip
a2enmod auth_openidc   # requiere el paquete de mod_auth_openidc

# Habilitar el sitio OAuth2
a2ensite 901-CON-HAproxy-oauth2.conf
systemctl reload apache2
```

3.4 Instalación de Apache para SAML2

```
a2dismod ssl
a2enmod remoteip
apt install libapache2-mod-auth-mellon

# Certificados y metadatos SAML en /etc/apache2/saml2/
#   saml2_sp_metadata.xml, saml2_sp_cert.crt, saml2_sp_cert.key, saml2_idp_metadata.xml

a2ensite 900-CON-HAproxy-saml2.conf
```

```
systemctl reload apache2
```

> **Importante**
> OAuth2 y SAML2 son configuraciones de sitio alternativas. Se debe habilitar únicamente el sitio correspondiente al protocolo elegido para la instalación; nunca ambos sobre el mismo VirtualHost.

3.5 Verificación

```
# HAProxy: comprobar configuración y estado de los backends
haproxy -c -f /etc/haproxy/haproxy.cfg
echo "show servers state" | socat /run/haproxy/admin.sock stdio

# Apache: comprobar módulos activos
apache2ctl -M | grep -E "remoteip|auth_openidc|mellon"

# Comprobar que serCen responde en local
curl -s http://localhost:7669/sercen | head

# Prueba funcional: acceder al FQDN y verificar el 302 al IdP
curl -skI https://<fqdn>/webs/webcall
```

Una autenticación correcta debe terminar en Fase C con un `200 OK` (OAuth2) o un `302 /webs/webfon2?token=...&token2=...` (SAML2).

4. Configuración

4.1 HAProxy — /etc/haproxy/haproxy.cfg

(OPCIONAL)

El elemento más importante para el SSO es el **balanceo por IP de origen con hash consistente**, que garantiza que el callback del IdP vuelva al mismo Apache que inició el flujo. La afinidad la proporciona el propio algoritmo `balance source`; **no se emplea sticky session ni stick-table** con ventana temporal.

```
frontend fe_vivait_https
    bind *:443 ssl crt /etc/haproxy/certs/<fqdn>.pem
    mode http
    default_backend be_vivait_apache

backend be_vivait_apache
    mode http
    balance source                # afinidad por hash de la IP de origen
    hash-type consistent          # minimiza la redistribución al caer/añadir un nodo
    http-request set-header X-Forwarded-Proto https
    option forwardfor             # añade X-Forwarded-For con la IP real del cliente
    cookie SRV insert indirect nocache
    server apache33 192.168.1.33:80 check
    server apache28 192.168.1.28:80 check
```

Parámetro	Efecto
balance source	Algoritmo de balanceo: selecciona el backend a partir del hash de la IP de origen. La misma IP va siempre al mismo Apache, lo que aporta la afinidad necesaria para state/nonce (OAuth2) y la SAMLResponse (SAML2) sin sticky session.
hash-type consistent	Hash consistente: al caer o añadirse un nodo solo se redistribuye una fracción de los clientes, en lugar de recalcular todo el reparto.
http-request set-header X-Forwarded-Proto https	Informa a Apache de que el cliente llegó por HTTPS (Apache escucha en HTTP/80).
option forwardfor	Inserta X-Forwarded-For con la IP real del cliente; imprescindible para logs y para fail2ban.
cookie SRV insert indirect nocache	Inserta una cookie de backend. Con balance source la afinidad ya está garantizada por el hash de IP; esta línea queda como mecanismo complementario.

Cambio respecto a versiones anteriores

La afinidad ya NO depende de una stick-table por IP con expiración (p. ej. expire 10s). Toda la persistencia hacia el mismo nodo se obtiene del balanceo balance source + hash-type consistent, que es estable mientras la IP de origen del cliente no cambie.

4.2 Apache OAuth2 — 901-CON-HAproxy-oauth2.conf

IP real del cliente (tras HAProxy si procede):

```
RemoteIPInternalProxy 192.168.1.33     # IP del HAProxy/proxy interno de confianza
RemoteIPHeader X-Forwarded-For
```

Parámetros OIDC principales:

```
OIDCProviderMetadataURL  https://login.microsoftonline.com/<TENANT-GUID>/.well-known/openid-configuration
OIDCProviderAuthRequestMethod POST
OIDCScope "openid email profile"
OIDCRemoteUserClaim upn               # upn -> email | usar 'oid' si campoid=identidad
OIDCXForwardedHeaders X-Forwarded-Proto
OIDCRedirectURI  https://<fqdn>/oauth2/privado/redirect_uri     #  invocada  por  Azure;  sin contenido propio
OIDCAuthNHeader X-USUID
OIDCAuthNHeader X-Forwarded-User
OIDCCookieDomain <fqdn>
OIDCStateMaxNumberOfCookies 7
OIDCClientID <CLIENT-ID-de-la-app-en-Azure>
OIDCClientSecret <CLIENT-SECRET>
OIDCCryptoPassphrase <CLAVE-CIFRADO-COOKIES>   # única por instalación
OIDCSSLValidateServer On
OIDCOAuthSSLValidateServer On

<Location /oauth2/privado>
    AuthType openid-connect
    Require valid-user
</Location>
```

Parámetro	Descripción
OIDCProviderMetadataURL	URL de descubrimiento OIDC del tenant de Microsoft Entra ID. El <TENANT-GUID> se obtiene en Azure (Id. del inquilino).
OIDCRemoteUserClaim	Claim que se expone como REMOTE_USER. upn para campoid=email; oid para campoid=identidad.
OIDCRedirectURI	URI de callback registrada en Azure. No debe servir contenido propio; la consume Azure.
OIDCCookieDomain	Dominio de la cookie de sesión cifrada. Debe ser el FQDN público.
OIDCClientID / OIDCClientSecret	Identificador y secreto de la aplicación registrada en Azure.
OIDCCryptoPassphrase	Clave de cifrado de cookies; valor distinto por instalación.

Servicios publicados y endpoint protegido de serCen:

```
<Location /sercen>
    ProxyPass http://localhost:7669/sercen retry=0 disablereuse=on
</Location>
<Location /sercen/getautenticacionapache>
    Require all denied              # NUNCA accesible desde el exterior
</Location>
<Location /janus>
    ProxyPass http://localhost:8088/janus retry=0 disablereuse=on
</Location>
<Location /Vivait-FonBO>
    ProxyPass http://localhost:8180/Vivait-FonBO retry=0 disablereuse=on
</Location>
```

4.3 Apache SAML2 — 900-CON-HAproxy-saml2.conf

```
ServerName https://<fqdn>
UseCanonicalName On
UseCanonicalPhysicalPort Off
RemoteIPInternalProxy 192.168.1.33
RemoteIPHeader X-Forwarded-For

<Location /saml2>
    MellonEnable info
    MellonEndpointPath /saml2/mellon/
    MellonSPMetadataFile /etc/apache2/saml2/saml2_sp_metadata.xml
    MellonSPPrivateKeyFile /etc/apache2/saml2/saml2_sp_cert.key
    MellonSPCertFile /etc/apache2/saml2/saml2_sp_cert.crt
    MellonIdPMetadataFile /etc/apache2/saml2/saml2_idp_metadata.xml
    MellonSecureCookie On
    MellonVariable vic-cookie
    MellonSessionLength 60
    MellonSignatureMethod rsa-sha256
    MellonSetEnv                                                              "saml_email"
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
    MellonSetEnv "saml_name"  "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"
    MellonUser "saml_email"        # REMOTE_USER = correo del usuario
</Location>

<Location /saml2/privado>
    AuthType Mellon
    MellonEnable auth
    Require valid-user
</Location>
```

Parámetro	Descripción
MellonEndpointPath	Ruta de los endpoints de Mellon (incluye /postResponse, destino del callback).
MellonSPMetadataFile / MellonSPCertFile / MellonSPPrivateKeyFile	Metadatos y certificados del SP (este servidor).
MellonIdPMetadataFile	Metadatos del IdP (Microsoft Entra ID); contiene la clave con la que se valida la firma.
MellonSignatureMethod	Algoritmo de firma esperado (rsa-sha256).
MellonVariable	Nombre de la cookie de sesión Mellon (vic-cookie).
MellonUser	Variable que se expone como REMOTE_USER (aquí, saml_email).

4.4 Firewall (nftables) — /etc/firewall/

La configuración de firewall es **simétrica entre los servidores VIVAit** y debe complementarse con reglas **en el firewall del cliente**. Se distinguen ambos planos a continuación.

4.4.1 Variables — /etc/firewall/vars.sh

```
export COUNTERS="counter packets 0 bytes 0"
export ETH=ens3
export IP_LOC=<IP-local-del-nodo>          # ej. 172.25.128.253
export IP_FLO=<IP-flotante>                 # = IP_LOC si no hay heartbeat
export RED_VOZ=<red-de-voz>                 # ej. 172.25.0.0/16
export RED_GESTION=<red-de-gestion-ssh>     # ej. 172.25.0.0/16
export ZABBIX_SRV=<IP-servidor-zabbix>      # ej. 172.25.128.92
```

> **Nota de despliegue**
>
> En la configuración de ejemplo del firewall las redes usan el rango 172.25.0.0/16, mientras que el pool de Apache de HAProxy está en 192.168.1.0/24 (192.168.1.33, 192.168.1.28). Se deben ajustar IP_LOC/IP_FLO y las reglas de pool a las direcciones reales de la instalación.

4.4.2 Reglas en los servidores VIVAit — qué añadir para SSO + HAProxy

```
# 1) HTTPS de clientes hacia HAProxy (IP local y flotante) - YA PRESENTE
nft add rule inet filter input iifname ${ETH} ip daddr { ${IP_LOC}, ${IP_FLO} } tcp dport 443
${COUNTERS} accept

# 2) HAProxy -> pool Apache por HTTP/80 (AÑADIR en los nodos Apache)
#    Permitir 80 SOLO desde la IP del HAProxy hacia cada Apache del pool.
nft add rule inet filter input iifname ${ETH} ip saddr <IP-HAProxy> tcp dport 80 ${COUNTERS}
accept

# 3) Salida HTTPS hacia Microsoft Entra ID (SOLO OAuth2: back-channel /token y metadata)
#    Necesaria desde los nodos Apache hacia login.microsoftonline.com.
nft add rule inet filter output oifname ${ETH} tcp dport 443 ct state new,established
${COUNTERS} accept

# 4) Respuestas HTTP/HTTPS de navegación/actualización - YA PRESENTE
nft add rule inet filter input iifname ${ETH} ip daddr ${IP_LOC} tcp sport 443 ct state
related,established ${COUNTERS} accept

# 5) DNS saliente (resolver login.microsoftonline.com) - asegurar salida 53
nft add rule inet filter output oifname ${ETH} udp dport 53 ct state new,established
${COUNTERS} accept

# 6) serCen: NO requiere regla de firewall (escucha en localhost:7669).
#    El acceso externo a /sercen/getautenticacionapache se bloquea en Apache, no en nftables.
```

Recordatorio

serCen se invoca siempre por localhost, por lo que NO debe abrirse el 7669 al exterior. MySQL (3306), RTP de Janus (20000-20999/udp) y el resto de servicios mantienen sus reglas existentes.

Resumen de puertos relevantes en los servidores VIVAit:

Sentido	Origen → Destino	Puerto	Motivo
Entrada	Cliente → HAProxy (IP_LOC/IP_FLO)	443/tcp	Acceso público al portal
Entrada	HAProxy → Apache pool	80/tcp	Reenvío balanceado
Salida	Apache → login.microsoftonline.com	443/tcp	Solo OAuth2: canje de code/metadata
Salida	Nodo → DNS	53/udp	Resolución del IdP
Local	Apache → serCen	7669/tcp (localhost)	Tokens internos (sin firewall)

4.4.3 Reglas en el firewall del cliente

El firewall del lado del cliente debe permitir, **para los navegadores de los usuarios** y la salida del back-channel OAuth2 desde los Apache:

Sentido	Origen → Destino	Puerto	¿Entrante en el perímetro?
Salida	Apache → login.microsoftonline.com (canje code, metadata, JWKS)	443/tcp	No
Salida	Navegador → <fqdn> (HAProxy, acceso al portal y callback)	443/tcp	Solo si el portal se publica hacia el exterior
Salida	Navegador → login.microsoftonline.com (login/MFA)	443/tcp	No
Salida	Nodo / Navegador → DNS	53	No

Sobre el back-channel OAuth2

El cliente debe habilitar conexiones SALIENTES desde los Apache hacia Microsoft (TCP 443 + DNS). El callback del IdP (OAuth2 redirect_uri, SAML2 /saml2/mellon/postResponse) siempre vuelve al navegador y de ahí a HAProxy (443). No es necesario permitir tráfico directo del cliente hacia los Apache internos (192.168.1.x): solo son alcanzables por HAProxy.

4.5 fail2ban

La introducción de HAProxy también afecta a **fail2ban**: como la carga se reparte entre varios nodos, el baneo de una IP debe **propagarse a todos los servidores** del pool. El mecanismo "truco" cubre esa necesidad escribiendo una marca en el syslog de todos los nodos.

4.5.1 Ficheros implicados

Fichero	Función
/etc/fail2ban/jail.d/vivait.local	Define qué se monitoriza (jails): Vivait-FonBO, Apache, FlexiSip, serCen, Janus…
/etc/fail2ban/jail.local	Lista de IPs que se ignoran (whitelist; no se les aplican las reglas).
/etc/fail2ban/filter.d/vivait-apache.local	Filtro: cadena a buscar en los logs de Apache.
/etc/fail2ban/filter.d/vivait-truco.local	Filtro del mecanismo "truco": busca la cadena TRUCO_KO seguida de la IP en los logs.
/etc/fail2ban/action.d/vivait-ban.local	Acción a ejecutar cuando se cumple el filtro: lanza fail2ban-vivait-ban.sh "<name>" "<ip>".

Estructura conceptual (terminología fail2ban): **Jail** → qué monitorizo · **Filter** → qué busco dentro del log · **Action** → qué hago si se cumple el filtro.

4.5.2 Flujo de baneo con HAProxy

Cambios en fail2ban

Provocado por la introducción del HAProxy (alta disponibilidad y balanceo)

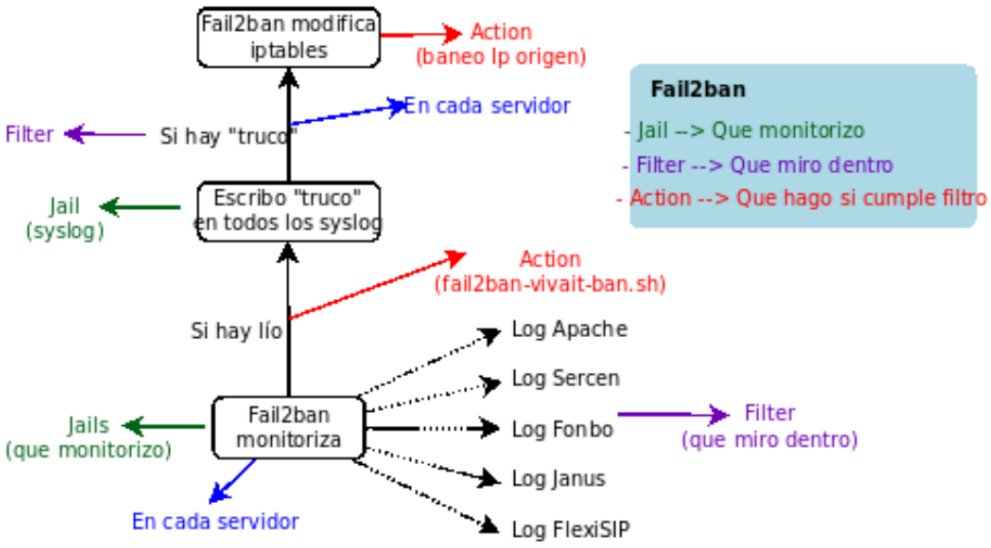

Ilustración 9. Cambios en fail2ban provocados por la introducción de HAProxy (mecanismo "truco").

El script de acción `/usr/local/sbin/fail2ban-vivait-ban.sh`:

```
#!/bin/bash
JAIL="$1"
IP="$2"
HORA="$(/usr/bin/date +%H:%M:%S.%N)"

function ban_local {
  /usr/bin/logger -d -t vivait "TRUCO_KO [$IP] fail2ban $JAIL $HORA"
}
function ban_remoto {
  /usr/bin/logger -n $1 -d -t vivait "TRUCO_KO [$IP] fail2ban $JAIL $HORA"
}

ban_local
#ban_remoto <IP-nodo-remoto>     # descomentar para propagar el baneo a otros nodos
```

Función	Efecto
ban_local	Escribe la marca TRUCO_KO [IP] en el syslog local. El filtro vivait-truco la detecta y nftables banea la IP en ese nodo.
ban_remoto <host>	Envía la marca al syslog de otro nodo (logger -n), de forma que el baneo se replique en todo el pool. Para activar la propagación se deben revisar y descomentar las líneas ban_* del final del script.

Compatibilidad con HAProxy

Para que fail2ban banee la IP real del atacante y no la del balanceador, Apache debe exponer la IP de cliente correcta vía RemoteIPHeader X-Forwarded-For y HAProxy debe enviar option forwardfor. Sin ello, todas las peticiones aparecerían con la IP de HAProxy y el baneo sería inservible.

5. Diagnósticos

5.1 Comandos de diagnóstico

```
# --- HAProxy ---
haproxy -c -f /etc/haproxy/haproxy.cfg                    # validar configuración
echo "show stat" | socat /run/haproxy/admin.sock stdio    # estado backends, sesiones y reparto por IP

# --- Apache OAuth2 (mod_auth_openidc en debug) ---
tail -f /var/log/apache2/error.log | grep auth_openidc    # state/nonce, canje, invalid_state
apache2ctl -M | grep auth_openidc

# --- Apache SAML2 (mod_auth_mellon) ---
tail -f /var/log/apache2/error.log | grep -i mellon       # validación de firma, SAMLResponse
apache2ctl -M | grep mellon

# --- serCen ---
curl -s "http://localhost:7669/sercen" | head             # disponibilidad local

# --- firewall (nftables) ---
nft list ruleset                                          # reglas activas y contadores

# --- fail2ban ---
fail2ban-client status                                    # jails activos
fail2ban-client status vivait-apache                      # IPs baneadas por un jail
fail2ban-client status vivait-truco
```

5.2 Ficheros de log

Fichero	Contenido
/var/log/apache2/error.log	Errores de Apache y trazas de mod_auth_openidc (LogLevel auth_openidc:debug) / mod_auth_mellon.
/var/log/apache2/access.log	Accesos al portal (con la IP real si RemoteIP está bien configurado).
/var/log/haproxy.log (vía syslog local0)	Tráfico, selección de backend y errores de HAProxy.
/var/log/syslog	Marcas TRUCO_KO [IP] del mecanismo fail2ban.
/var/log/fail2ban.log	Detecciones y baneos de fail2ban.

5.3 Errores comunes y soluciones

Síntoma / Mensaje	Causa probable	Solución
invalid_state (OAuth2)	El callback cayó en un Apache	Verificar balance source + hash-type

Síntoma / Mensaje	Causa probable	Solución
	distinto al que generó el state/nonce (la IP de origen cambió o el balanceo no es por IP).	consistent; confirmar que la IP del cliente no cambia (cuidado con NAT de salida con varias IPs).
Error de validación SAML / firma	mod_auth_mellon no puede validar la SAMLResponse; metadatos/certificados incorrectos.	Revisar saml2_idp_metadata.xml, saml2_sp_cert.crt/.key y que MellonSignatureMethod coincida con el del IdP (rsa-sha256).
Bucle de redirección al IdP	La sesión (cookie OIDC o vic-cookie) no se conserva entre peticiones.	Revisar OIDCCookieDomain/MellonVariable, el FQDN y la afinidad de HAProxy.
REMOTE_USER vacío en Fase C	El claim configurado no llega o no se mapea.	OAuth2: revisar OIDCRemoteUserClaim (upn/oid). SAML2: revisar MellonUser y MellonSetEnv del email.
serCen no entrega token/token2	serCen caído o /sercen no proxificado.	Comprobar curl localhost:7669/sercen; revisar el bloque ProxyPass /sercen.
/sercen/getautenticacionapache accesible desde fuera	Falta el bloqueo en Apache.	Confirmar <Location /sercen/getautenticacionapache> Require all denied.
fail2ban banea la IP de HAProxy	Apache ve la IP del proxy, no la del cliente.	Configurar RemoteIPHeader X-Forwarded-For + RemoteIPInternalProxy y option forwardfor en HAProxy.
Backend DOWN en HAProxy	Apache caído o 80 bloqueado por nftables.	Revisar show stat; permitir 80 desde la IP de HAProxy hacia el Apache (sección 4.4.2).
OAuth2 falla solo en el canje de tokens	Salida 443 hacia login.microsoftonline.com bloqueada en el firewall del nodo.	Añadir la regla de salida 443 (sección 4.4.2, regla 3) y verificar DNS.

5.4 Comprobaciones de salud rápidas

```
# El portal redirige al IdP en la primera petición (302)
curl -skI https://<fqdn>/webs/webcall | grep -i location

# Afinidad por IP: desde la misma IP, las peticiones deben caer en el mismo backend
echo "show stat" | socat /run/haproxy/admin.sock stdio | cut -d, -f1,2,8,34

# Salida hacia Microsoft Entra ID (solo OAuth2)
curl -sI https://login.microsoftonline.com/<TENANT-GUID>/.well-known/openid-configuration | head -1
```

Nota de seguridad final

Los ficheros de configuración contienen secretos en claro (OIDCClientSecret, OIDCCryptoPassphrase, claves SAML, certificado .pem de HAProxy). Se deben restringir los permisos (chmod 600, propietario root/haproxy según el servicio), evitar versionarlos en claro y rotarlos según la política de la instalación.