

SAML-Anwendung an ADFS anbinden – Praxisbeispiel SaaS

ADFS • Praxis mit SAML 2.0

SaaS-Anwendung an ADFS anbinden

Zwei XML-Dateien, ein Format-String, ein paar Claims – und der Anbieter sagt trotzdem „Invalid SAML Response“.

ADFS-Farm
sts.contoso.de

→ Assertion
← AuthnRequest

SaaS-App
„Zeitwerk“

Metadaten • NameID • Claims • Signatur

boddenberg.de

WISSEN

Grundlagen, Architektur und alle Praxisbeiträge rund um ADFS an einem Ort.

› [Active Directory Federation Services](#)

BERATUNG

SaaS-Anbindung prüfen, Claims sauber schneiden, hängende Anmeldungen gemeinsam auseinandernehmen.

› [Consulting zu ADFS](#)

SCHULUNG

SAML im Labor: Metadaten tauschen, Tokens lesen – und absichtlich eine NameID zerbrechen.

› [ADFS-Schulungen](#)

Die Mail kommt am Freitag um 15:47 Uhr. Der Fachbereich hat eine neue SaaS-Anwendung gekauft – im Beispiel eine Zeiterfassung, nennen wir sie „Zeitwerk“ –, der Vertrag ist unterschrieben, die Schulung der Belegschaft für Montag angesetzt. Im Anhang ein PDF des Anbieters mit dem schönen Satz: „Single Sign-on via SAML 2.0 wird unterstützt. Bitte konfigurieren Sie Ihren Identity Provider entsprechend.“ Das ist ungefähr so hilfreich wie ein Beipackzettel, auf dem nur „Wirkt“ steht. Aber gut: Dein Identity Provider ist eine ADFS-Farm, die seit Jahren zuverlässig läuft, und genau für so etwas wurde sie gebaut.

Dieser Beitrag zeigt dir an diesem anonymisierten Beispiel den kompletten Ablauf, um eine SAML-Anwendung an ADFS anzubinden: Metadaten tauschen, das NameID-Format festlegen, die typischen Claims E-Mail, UPN und Gruppen ausstellen, SP- und IdP-initiierte Anmeldung verstehen und das Ergebnis mit den Entwicklertools des Browsers oder einem SAML-Tracer prüfen. Am Ende steht eine Fehlerbild-Tabelle für die drei Klassiker, an denen gefühlt jede zweite Anbindung erst einmal scheitert: falsche NameID, Uhrzeitabweichung, Signatur. Die Grundlagen zu Farm, Protokollen und Vertrauensstellungen setzen wir voraus – die findest du auf der Übersichtsseite [Active Directory Federation Services](#).

FAKTEN – Das Beispiel in Zahlen und Namen

Identity Provider: ADFS-Farm unter sts.contoso.de, Active Directory contoso.local, Benutzer-UPN-Suffix contoso.de.

Service Provider: SaaS-Zeiterfassung, Mandanten-URL contoso.zeitwerk.example, Protokoll SAML 2.0, HTTP-POST-Binding.

Anforderung des Anbieters: NameID im Format E-Mail-Adresse, Attribute für E-Mail, Vorname, Nachname und eine Rolle für Administratoren.

Alle Namen sind Platzhalter. Die Abläufe gelten für praktisch jede SAML-fähige SaaS-Anwendung – ob Ticketsystem, Lernplattform oder Reisekostentool.

Vorbereitung: Was du vom Anbieter brauchst und was er von dir bekommt

Bevor du die ADFS-Verwaltung öffnest, klärst du mit dem Anbieter fünf Dinge. Wer das überspringt, klärt sie später trotzdem – nur dann unter Zeitdruck und mit einem Ticket beim Support, das „in der Reihenfolge des Eingangs bearbeitet“ wird.

Frage an den Anbieter	Warum sie zählt	Antwort im Beispiel
Gibt es SP-Metadaten als URL oder Datei?	Spart Tipparbeit und erlaubt ADFS, Zertifikatswechsel des Anbieters selbst nachzuziehen.	Ja, als URL im Admin-Portal
Welcher Entity-ID-Wert gilt?	Wird in ADFS zum Bezeichner des Relying Party Trust und muss in der Audience exakt so stehen.	https://contoso.zeitwerk.example
Welche ACS-URL und welches Binding?	Dorthin schickt ADFS die SAMLResponse. Eine falsche URL beendet die Anmeldung schon bei ADFS.	.../saml/acs, HTTP-POST
Welches NameID-Format und welcher Wert?	Der Schlüssel, unter dem die App ihre Benutzer wiederfindet.	emailAddress, Wert = mail
Welche Attribute unter welchen Namen?	Die App mappt Attributnamen zeichengenau. „email“ ist nicht „Email“.	E-Mail, Vorname, Nachname, Rolle

Tabelle 1: Die Checkliste vor der ersten Konfiguration

Deine Metadaten: die Visitenkarte der Farm

ADFS veröffentlicht seine eigenen Metadaten unter einer festen Adresse, im Beispiel <https://sts.contoso.de/FederationMetadata/2007-06/FederationMetadata.xml>. Darin stehen der Bezeichner der Farm, die Anmelde- und Abmeldeendpunkte und der öffentliche Teil des Token-Signing-Zertifikats. Die meisten Anbieter nehmen entweder diese URL oder die heruntergeladene XML-Datei entgegen. Manche wollen die Werte einzeln: dann trägst du die Login-URL <https://sts.contoso.de/adfs/ls/> ein, den Issuer und das Zertifikat als CER-Datei.

Ein Detail, das mehr Anbindungen aufhält, als man glauben möchte: Der Bezeichner einer ADFS-Farm beginnt in der Standardkonfiguration mit [http://](http://sts.contoso.de/adfs/services/trust) und nicht mit [https://](https://sts.contoso.de/adfs/services/trust), also etwa <http://sts.contoso.de/adfs/services/trust>. Das ist kein Sicherheitsproblem, sondern nur ein Name. Aber wenn jemand im Admin-Portal des Anbieters „aus Sicherheitsgründen“ ein s ergänzt, passt der Issuer nicht mehr, und die App lehnt jede Antwort ab.

```
# Bezeichner der Farm und Metadaten-Adresse prüfen
Get-AdfsProperties | Select-Object HostName, Identifier, FederationPassiveAddress

# Aktuelles Token-Signing-Zertifikat als CER-Datei für den Anbieter exportieren
$ts = (Get-AdfsCertificate -CertificateType Token-Signing | Where-Object IsPrimary).Certificate
[IO.File]::WriteAllBytes("C:\Temp\sts-contoso-token-signing.cer", $ts.Export("Cert"))
```

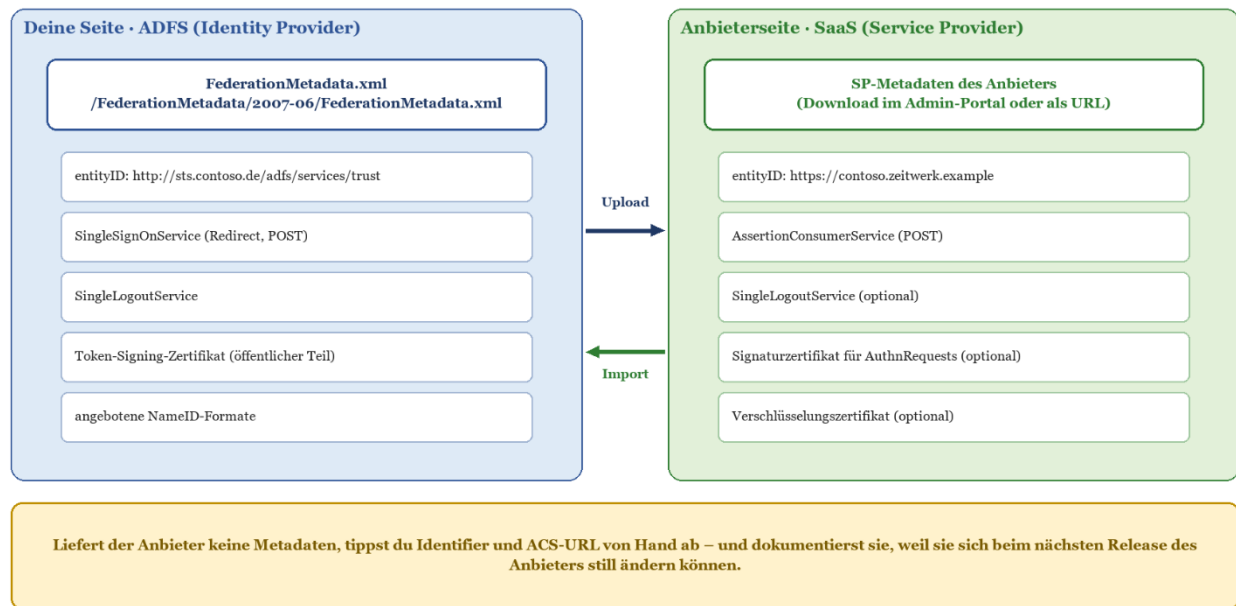
Listing 1: Was der Anbieter von dir braucht – Bezeichner, Login-Adresse und Signaturzertifikat

WARNUNG — Ein exportiertes Zertifikat hat ein Verfallsdatum – auch im Kopf des Anbieters

Liest der Anbieter deine Metadaten-URL regelmäßig neu ein, übernimmt er einen Zertifikatswechsel selbst. Hat er nur eine hochgeladene CER-Datei, scheitert die Anmeldung am Tag nach dem Rollover deines Token-Signing-Zertifikats. Notiere solche Anwendungen in deiner Dokumentation. Wie du den Wechsel planst, steht in [ADFS Token-Signing-Zertifikat erneuern – mit und ohne AutoCertificateRollover](#).

Skizze 1: Der Metadatentausch – wer wem was liefert

Zwei XML-Dokumente ersetzen ein halbes Dutzend Telefonate – wenn beide Seiten sie auch lesen



Skizze 1: Jede Seite liefert der anderen ihre Metadaten – Bezeichner, Endpunkte und Zertifikate.

Den Relying Party Trust anlegen

Auf der ADFS-Seite wird der Anbieter zum Relying Party Trust. In der Verwaltungskonsole geht das über den Assistenten „Vertrauensstellung der vertrauenden Seite hinzufügen“ mit der Option „Ansprüche unterstützend“; per PowerShell ist es reproduzierbarer und lässt sich in der Dokumentation ablegen. Die ausführliche Fassung mit allen Varianten – Metadaten-URL, Datei, Handarbeit – findest du in [Relying Party Trust anlegen – per Metadaten und von Hand](#). Für unser Beispiel reichen zwei Wege.

```
# Variante A: Der Anbieter stellt SP-Metadaten bereit
Add-AdfsRelyingPartyTrust -Name "Zeitwerk (SaaS)" `
  -MetadataUrl "https://contoso.zeitwerk.example/saml/metadata" `
  -MonitoringEnabled $true -AutoUpdateEnabled $true `
  -AccessControlPolicyName "Permit everyone" `
  -IssuanceTransformRulesFile "C:\ADFS\Regeln\zeitwerk.txt"

# Variante B: Keine Metadaten – Bezeichner und ACS-Endpunkt von Hand
$sacs = New-AdfsSamlEndpoint -Binding "POST" -Protocol "SAMLAssertionConsumer" `
  -Uri "https://contoso.zeitwerk.example/saml/acs" -IsDefault $true

Add-AdfsRelyingPartyTrust -Name "Zeitwerk (SaaS)" `
  -Identifier "https://contoso.zeitwerk.example" `
  -SamlEndpoint $sacs -ProtocolProfile "SAML" `
  -AccessControlPolicyName "Permit everyone" `
  -IssuanceTransformRulesFile "C:\ADFS\Regeln\zeitwerk.txt"
```

Listing 2: Relying Party Trust für die SaaS-Anwendung – mit und ohne Metadaten des Anbieters

TIPP – „Permit everyone“ ist ein Startwert, keine Dauerlösung

Für den ersten Test darf jeder Benutzer durch. Sobald die Anmeldung funktioniert, stellst du auf eine Richtlinie um, die nur die vorgesehene Gruppe zulässt – sonst landet jeder Praktikant per SSO in der Zeiterfassung des Vorstands, und die App legt ihm freundlich ein Konto an. Wie das ohne Regelakrobatik geht, zeigt [Access Control Policies in ADFS – Zugriffsregeln ohne Claim-Rule-Akrobatik](#).

NameID und Claims: Was im Token stehen muss

Der Relying Party Trust steht, aber ohne Ausstellungsregeln schickt ADFS ein Token, in dem praktisch nichts steht. Die SaaS-Anwendung braucht mindestens eine NameID – das Feld im Subject der Assertion, an dem sie den Benutzer erkennt. Dazu kommen Attribute, mit denen sie Konten anlegt oder aktualisiert. Hier entscheidet sich, ob die Anbindung in drei Jahren noch ruhig läuft oder ob jede Namensänderung nach einer Heirat ein Ticket erzeugt.

Das NameID-Format festlegen

SAML kennt mehrere NameID-Formate. Welches du wählst, gibt in aller Regel der Anbieter vor – die Frage ist nur, ob seine Vorgabe zu deinem Verzeichnis passt. Die vier Formate, die dir in der Praxis begegnen:

Format	Format-URI (gekürzt)	Typischer Wert	Einschätzung
E-Mail-Adresse	...:SAML:1.1:nameid-format:emailAddress	anna.berg@contoso.de	Häufigste Vorgabe bei SaaS. Gut lesbar, aber Adressen ändern sich – bei Heirat, Umfirmierung, neuer Domäne.
Unspecified	...:SAML:1.1:nameid-format:unspecified	UPN oder Personalnummer	Der Anbieter nimmt, was kommt. Bequem, solange beide Seiten wissen, was gemeint ist.
Persistent	...:SAML:2.0:nameid-format:persistent	stabile, oft undurchsichtige ID	Fachlich die sauberste Wahl: bleibt bei Namensänderungen gleich. Erfordert, dass die App damit umgehen kann.
Transient	...:SAML:2.0:nameid-format:transient	wechselt bei jeder Anmeldung	Nur für Apps ohne eigenes Benutzerkonto. Für Zeiterfassung, CRM und Co. ungeeignet.

Tabelle 2: NameID-Formate und ihr Charakter

Im Beispiel verlangt der Anbieter das Format E-Mail-Adresse. Das ist in Ordnung, wenn du eine Regel für Adressänderungen hast: Entweder der Anbieter kann den Benutzer anhand eines zweiten, stabilen Attributs umschlüsseln, oder du änderst im Admin-Portal die Adresse mit, bevor die neue im AD steht. Fragt dich niemand, ist das Ergebnis vorhersehbar: Frau Berg heißt nach der Hochzeit Frau Hoffmann, meldet sich an, und die Zeiterfassung begrüßt sie als neue Kollegin mit null erfassten Stunden. Für die Buchhaltung ist das ein Wunder, für die Betroffene weniger.

WICHTIG – Die NameID ist ein Fremdschlüssel – behandle sie so

Wähle als Quelle ein Attribut, das sich so selten wie möglich ändert und eindeutig ist. mail und userPrincipalName sind praktisch, aber nicht unveränderlich.

Wechselst du später das Quellattribut oder das Format, verlieren alle bestehenden Benutzer die Zuordnung zu ihrem Konto in der App. Das ist ein Migrationsprojekt, keine Konfigurationsänderung.

Halte schriftlich fest, welches Attribut in welchem Format als NameID geht. Bei der späteren Migration nach Entra ID ist das die erste Frage.

Die typischen Claims: E-Mail, UPN, Gruppen

In ADFS entsteht die NameID nicht direkt aus einem AD-Attribut, sondern über einen Zwischenschritt: Zuerst holt eine LDAP-Regel die Attribute als Claims, dann wandelt eine Transformationsregel einen dieser Claims in den Name-ID-Claim mit der passenden Format-Eigenschaft um. Gruppen kommen über eine eigene Regel dazu – gefiltert, denn die SaaS-

Anwendung muss nicht wissen, dass Anna in 74 Verteilerlisten steht und Mitglied von „Kantine-Kuchenspender“ ist. Die Regelsprache im Detail erklärt der [ADFS Claim Rules Leitaden](#).

```
@RuleName = "Regel 1: AD-Attribute als Claims"
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname",
  Issuer == "AD AUTHORITY"]
=> issue(store = "Active Directory",
  types = ("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress",
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn",
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname",
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"),
  query = ";mail,userPrincipalName,givenName,sn;{0}", param = c.Value);

@RuleName = "Regel 2: E-Mail als NameID im Format emailAddress"
c:[Type == "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"]
=> issue(Type = "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier",
  Issuer = c.Issuer, OriginalIssuer = c.OriginalIssuer, Value = c.Value, ValueType =
c.ValueType,
  Properties["http://schemas.xmlsoap.org/ws/2005/05/identity/claimproperties/format"]
    = "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress");

@RuleName = "Regel 3a: Gruppen ermitteln (nur intern)"
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname",
  Issuer == "AD AUTHORITY"]
=> add(store = "Active Directory", types = ("http://schemas.xmlsoap.org/claims/Group"),
  query = ";tokenGroups;{0}", param = c.Value);

@RuleName = "Regel 3b: Nur App-Gruppen als Rolle ausstellen"
c:[Type == "http://schemas.xmlsoap.org/claims/Group", Value =~ "^SaaS-Zeitwerk-"]
=> issue(Type = "http://schemas.microsoft.com/ws/2008/06/identity/claims/role", Value =
c.Value);
```

Listing 3: Inhalt von zeitwerk.txt – Attribute, NameID und gefilterte Gruppen

Zwei Feinheiten an diesem Regelsatz. Erstens: Regel 3a verwendet add statt issue. Die Gruppen landen damit nur in der internen Verarbeitung und nicht im Token; erst Regel 3b stellt die gefilterten Werte aus. Zweitens: Der Anbieter mappt Attribute über ihren Namen, und ADFS schreibt als Attributnamen genau den Claim-Typ in die Assertion – also die lange URI. Erwartet die App kurze Namen wie „email“ oder „firstName“, stellst du die Claims in einer eigenen Regel mit genau diesem Typ aus, etwa Type = "firstName". Das sieht unorthodox aus, ist aber zulässig und bei vielen SaaS-Anbindungen der einzige Weg, ohne Mapping-Tabelle beim Anbieter auszukommen.

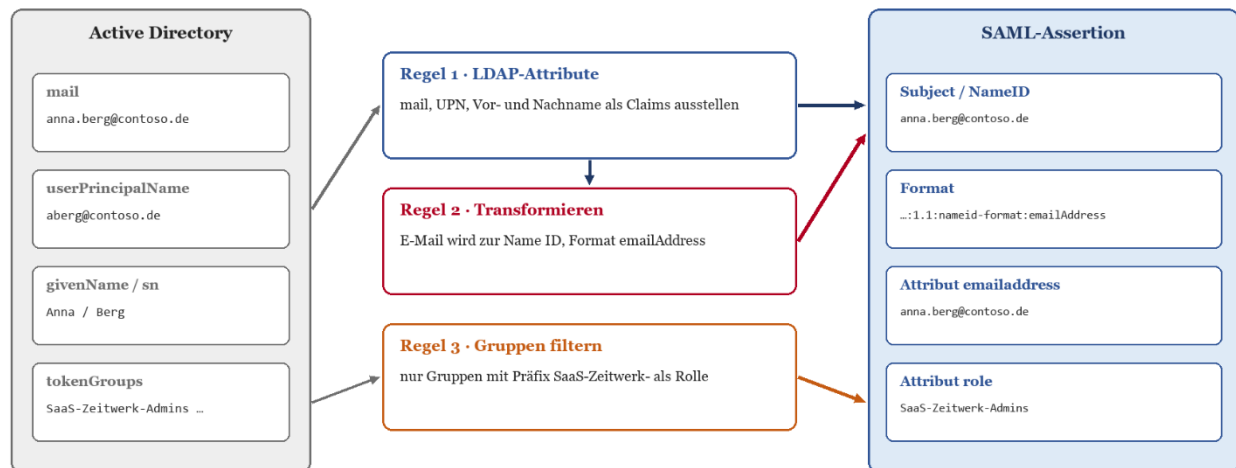
```
Set-AdfsRelyingPartyTrust -TargetName "Zeitwerk (SaaS)" `
  -IssuanceTransformRulesFile "C:\ADFS\Regeln\zeitwerk.txt"

# Kontrolle: Stehen die Regeln wirklich am Trust?
(Get-AdfsRelyingPartyTrust -Name "Zeitwerk (SaaS)").IssuanceTransformRules
```

Listing 4: Regeln zuweisen und prüfen

Skizze 2: Vom AD-Attribut zur Assertion

Drei Regeln, eine NameID, ein paar Attribute – mehr braucht eine typische SaaS-App nicht



Die NameID ist der Schlüssel, unter dem die App den Benutzer wiederfindet. Ändert sich dieser Wert, ist Anna für die App eine Fremde – mit leerem Konto.

Skizze 2: Aus vier AD-Attributen werden über drei Regeln eine NameID und eine Handvoll Attribute.

Claim	Quelle im AD	Wofür die App ihn braucht	Stolperstein
E-Mail	mail	NameID, Kontaktadresse, Kontoabgleich	Leeres mail-Attribut bei Funktions- und Dienstkonten – dann fehlt die NameID ganz.
UPN	userPrincipalName	Alternative NameID, Abgleich mit anderen Systemen	Interner UPN-Suffix wie contoso.local ist für SaaS meist wertlos.
Vor- und Nachname	givenName, sn	Anzeige, Just-in-Time-Anlage des Kontos	Wird bei manchen Apps nur bei der ersten Anmeldung übernommen.
Rolle / Gruppe	tokenGroups, gefiltert	Berechtigungen in der App, Admin-Rechte	Ungefiltert entstehen riesige Tokens – und ein Informationsleck.

Tabelle 3: Die typischen Claims einer SaaS-Anbindung

WARNUNG – Gruppenclaims sind Berechtigungen – auch außerhalb deines Hauses

Viele SaaS-Anwendungen vergeben Administratorrechte anhand eines Rollen-Claims. Wer in AD die Gruppe SaaS-Zeitwerk-Admins pflegen darf, verwaltet damit faktisch die App. Prüfe also, wer diese Gruppen ändern kann, und stelle nie ungefiltert alle Gruppen aus. Warum ADFS-Tokens als Ganzes ein lohnendes Angriffsziel sind, beschreibt [ADFS-Sicherheit: Angriffe, Härtung, Golden SAML und MFA](#).

Signatur, Algorithmus und Verschlüsselung

ADFS signiert in der Standardeinstellung die Assertion, nicht die gesamte Response, und verwendet SHA-256. Beides lässt sich pro Relying Party Trust ändern, und beides ist eine häufige Quelle für Anbindungsfehler. Manche Anbieter erwarten eine signierte Response, manche beides. Ältere Anwendungen bestehen gelegentlich noch auf SHA-1 – das stellst du nur um, wenn es wirklich nicht anders geht, und mit einer Notiz im Feld Notes des Trusts.

Einstellung	Parameter	Werte	Wann du sie änderst
Was wird signiert	-SamlResponseSignature	AssertionOnly, MessageOnly,	Der Anbieter meldet „Response not signed“ oder

		MessageAndAssertion	Ähnliches.
Signaturalgorithmus	-SignatureAlgorithm	rsa-sha256 oder rsa-sha1	Nur wenn der Anbieter SHA-256 nachweislich nicht kann.
Verschlüsselung	-EncryptClaims	\$true / \$false	Wirkt nur mit Verschlüsselungszertifikat aus den SP-Metadaten.
Signierte Requests	- SignedSamlRequestsRequired	\$true / \$false	Wenn der Anbieter AuthnRequests signiert und du das erzwingen willst.
Zeitpuffer	-NotBeforeSkew	ganze Zahl, Standard 0	Wenn die App meldet, das Token sei noch nicht gültig.

Tabelle 4: Die Stellschrauben am Relying Party Trust, die SaaS-Anbieter am häufigsten interessieren

TIPP – Verschlüsselung zum Testen kurz abschalten – und wieder einschalten

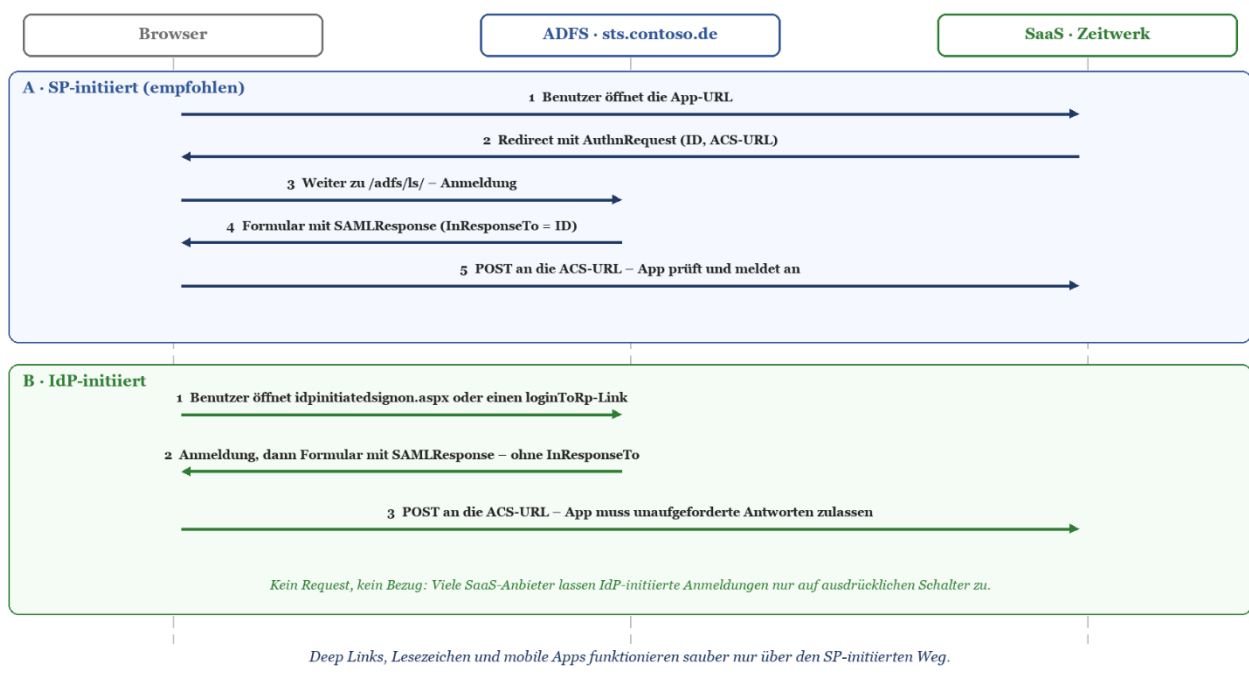
Bringen die SP-Metadaten ein Verschlüsselungszertifikat mit, verschlüsselt ADFS die Assertion. Im SAML-Tracer siehst du dann nur ein EncryptedAssertion-Element und keinen einzigen Claim. Für die Fehlersuche in einer Testumgebung kannst du mit Set-AdfsRelyingPartyTrust -EncryptClaims \$false kurz in die Assertion schauen. Danach schaltest du die Verschlüsselung wieder ein. In der Produktion lässt du das bleiben – dort hilft das Fehlerprotokoll des Anbieters.

SP-initiiert oder IdP-initiiert: Wo die Anmeldung beginnt

Eine SAML-Anmeldung kann von zwei Seiten starten. Beim SP-initiierten Weg öffnet der Benutzer die Anwendung, die App schickt ihn mit einem AuthnRequest zu ADFS, und ADFS antwortet mit einer SAMLResponse, die sich per InResponseTo auf genau diesen Request bezieht. Beim IdP-initiierten Weg beginnt alles bei ADFS: Der Benutzer wählt die App auf einer ADFS-Seite oder klickt einen vorbereiteten Link, und ADFS schickt eine Antwort, um die niemand gebeten hat.

Skizze 3: SP-initiiert und IdP-initiiert im Vergleich

Gleiches Ziel, anderer Startpunkt – und nur einer der beiden Wege kennt einen AuthnRequest



Skizze 3: Der SP-initiierte Weg kennt einen AuthnRequest, der IdP-initiierte nicht – mit Folgen für Sicherheit und Deep Links.

Kriterium	SP-initiiert	IdP-initiiert
Startpunkt	URL der Anwendung, Lesezeichen, Deep Link	ADFS-Anmeldeseite oder vorbereiteter Link
AuthnRequest	Ja, mit ID, ACS-URL, gewünschtem NameID-Format	Nein
Bezug der Antwort	InResponseTo verweist auf den Request	Kein Bezug – unaufgeforderte Antwort
Deep Links	Funktionieren über RelayState	Nur, wenn App und Link RelayState sauber auswerten
Voraussetzung in ADFS	Keine besondere	IdP-initiierte Anmeldeseite aktiviert oder loginToRp-Link
Voraussetzung beim Anbieter	Standard	Muss unaufgeforderte Antworten ausdrücklich zulassen
Empfehlung	Standardweg für Benutzer	Testwerkzeug und Portal-Link, wenn der Anbieter es will

Tabelle 5: Die beiden Anmeldewege im direkten Vergleich

Die IdP-initiierte Anmeldeseite als Testwerkzeug

Seit ADFS unter Windows Server 2016 ist die Seite `idpinitiatedsignon.aspx` in der Standardkonfiguration abgeschaltet. Für Tests ist sie trotzdem Gold wert, weil sie den Anbieter zunächst aus dem Spiel lässt: Du meldest dich direkt bei ADFS an und siehst, ob die Farm überhaupt ein Token für die App ausstellt. Ob du sie dauerhaft eingeschaltet lässt, ist eine bewusste Entscheidung – sie zeigt in der Auswahlliste die Namen deiner SAML-Anwendungen an.

```
# Seite einschalten (wirkt farmweit)
Set-AdfsProperties -EnableIdpInitiatedSignonPage $true
Get-AdfsProperties | Select-Object EnableIdpInitiatedSignonPage

# Aufruf im Browser:
# https://sts.contoso.de/adfs/ls/idpinitiatedsignon.aspx
# Direkt zur App, ohne Auswahlliste – Bezeichner des Relying Party Trust als Parameter:
#
https://sts.contoso.de/adfs/ls/idpinitiatedsignon.aspx?loginToRp=https://contoso.zeitwerk.example
```

Listing 5: IdP-initiierte Anmeldung für den Test aktivieren

FAKTEN – Warum SP-initiiert der bessere Standard ist

Beim SP-initiierten Weg kann die App prüfen, ob die Antwort zu einem Request gehört, den sie selbst verschickt hat. Eine unaufgeforderte Antwort hat diesen Bezug nicht – deshalb lassen viele Anbieter IdP-initiierte Anmeldungen nur auf ausdrücklichen Schalter zu.

Lesezeichen, Links in Mails und Deep Links in die App funktionieren nur sauber, wenn die App die Anmeldung selbst anstößt und sich die Zieladresse merkt.

Für Benutzer ist SP-initiiert ohnehin der natürliche Weg: Sie öffnen die App und werden einmal kurz zu `sts.contoso.de` umgeleitet. Bei internen Clients mit Windows-Anmeldung merken sie davon nicht einmal das.

Im Admin-Portal des Anbieters findest du meist einen Schalter in der Art „SP-initiated login only“ oder „Allow IdP-initiated login“. Für die Zeiterfassung im Beispiel lassen wir beides zu: Der Normalfall ist die App-URL, die IdP-initiierte Variante bleibt als Prüfweg für den Betrieb. Wer die Anmeldung zusätzlich mit MFA absichern will, hängt das an den Relying Party Trust und nicht an die App – wie, steht in [ADFS mit Entra-MFA koppeln – Adapter, Zertifikat und Fallstricke](#).

Testen mit Entwicklertools und SAML-Tracer

Der Moment der Wahrheit: Du öffnest die App-URL, wirst zu ADFS umgeleitet, meldest dich an – und landest entweder in der Zeiterfassung oder auf einer Fehlerseite des Anbieters, die so

aussagekräftig ist wie ein Horoskop. „Invalid SAML response“, „User not found“, „Authentication failed“. Damit du nicht raten musst, schaust du dir an, was tatsächlich über die Leitung ging. Das Schöne an SAML im Browser: Alles, was zwischen ADFS und App passiert, läuft durch deinen Browser. Du musst es nur sichtbar machen.

Variante 1: Die Entwicklertools des Browsers

Jeder aktuelle Browser bringt Entwicklertools mit, die du mit F12 öffnest. Im Reiter Netzwerk aktivierst du „Protokoll beibehalten“ – sonst verschwinden die Einträge bei jeder Umleitung, und SAML besteht praktisch nur aus Umleitungen. Dann startest du die Anmeldung. Du suchst zwei Einträge: die Anfrage an /adfs/ls/ mit dem Parameter SAMLRequest, und den POST an die ACS-URL des Anbieters mit dem Formularfeld SAMLResponse.

- Die SAMLResponse ist Base64-kodiert. Den Inhalt dekodierst du lokal, zum Beispiel in PowerShell mit [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String(\$wert)).
- Der SAMLRequest im Redirect-Binding ist zusätzlich komprimiert und URL-kodiert. Hier ist ein SAML-Tracer bequemer.
- Kopiere Tokens aus der Produktion nicht in Online-Dekodierer. Eine signierte Assertion ist bis zu ihrem Ablauf eine gültige Eintrittskarte – und enthält personenbezogene Daten.

Variante 2: SAML-Tracer als Browser-Erweiterung

Ein SAML-Tracer ist eine Browser-Erweiterung, die SAML-Nachrichten im Datenverkehr erkennt, dekodiert und als lesbares XML anzeigt. Du startest ihn, meldest dich an und klickst den markierten Eintrag an. Im Reiter SAML siehst du die komplette Response, mit Issuer, NameID, Zeitstempeln, Audience und allen Attributen. Für die Fehlersuche ist das die mit Abstand schnellste Methode – und sie funktioniert, ohne dass du am ADFS-Server irgendetwas umstellen musst. Wie bei jeder Erweiterung gilt: Installiere sie auf einem Admin-Arbeitsplatz oder in einem eigenen Browserprofil, nicht im Alltagsbrowser.

Skizze 4: Was du im SAML-Tracer prüfst

Eine dekodierte SAMLResponse, von oben nach unten gelesen – und an jeder Stelle eine typische Fehlerquelle

```
<samlp:Response Destination="https://contoso.zeitwerk.example/saml/acs"
  InResponseTo="_8f3c..." IssueInstant="2026-10-01T07:42:10Z">
  <Issuer>http://sts.contoso.de/adfs/services/trust</Issuer>
  <samlp:Status> ... Success ... </samlp:Status>
  <Assertion>
    <ds:Signature> ... rsa-sha256 ... </ds:Signature>
    <Subject>
      <NameID Format="...:nameid-format:emailAddress">
        anna.berg@contoso.de</NameID>
      <Conditions NotBefore="07:42:10Z"
        NotOnOrAfter="08:42:10Z">
      </Conditions>
      <Audience>https://contoso.zeitwerk.example</Audience>
      <AttributeStatement> ... emailaddress, role ...
    </Subject>
  </Assertion>
</samlp:Response>
```

Destination = ACS-URL aus den SP-Metadaten?

Issuer = Bezeichner, den der Anbieter erwartet?

Status nicht Success? Dann liegt der Fehler bei ADFS – Admin-Log

Signatur: Zertifikat und Algorithmus wie beim Anbieter hinterlegt?

NameID: Format und Wert wie in der App gespeichert?

Zeitfenster: Uhren beider Seiten synchron?

Audience = entityID des Anbieters, Zeichen für Zeichen?

Attributnamen exakt so, wie die App sie mappt?

Skizze 4: Die dekodierte SAMLResponse – acht Stellen, an denen eine Anbindung typischerweise hakt.

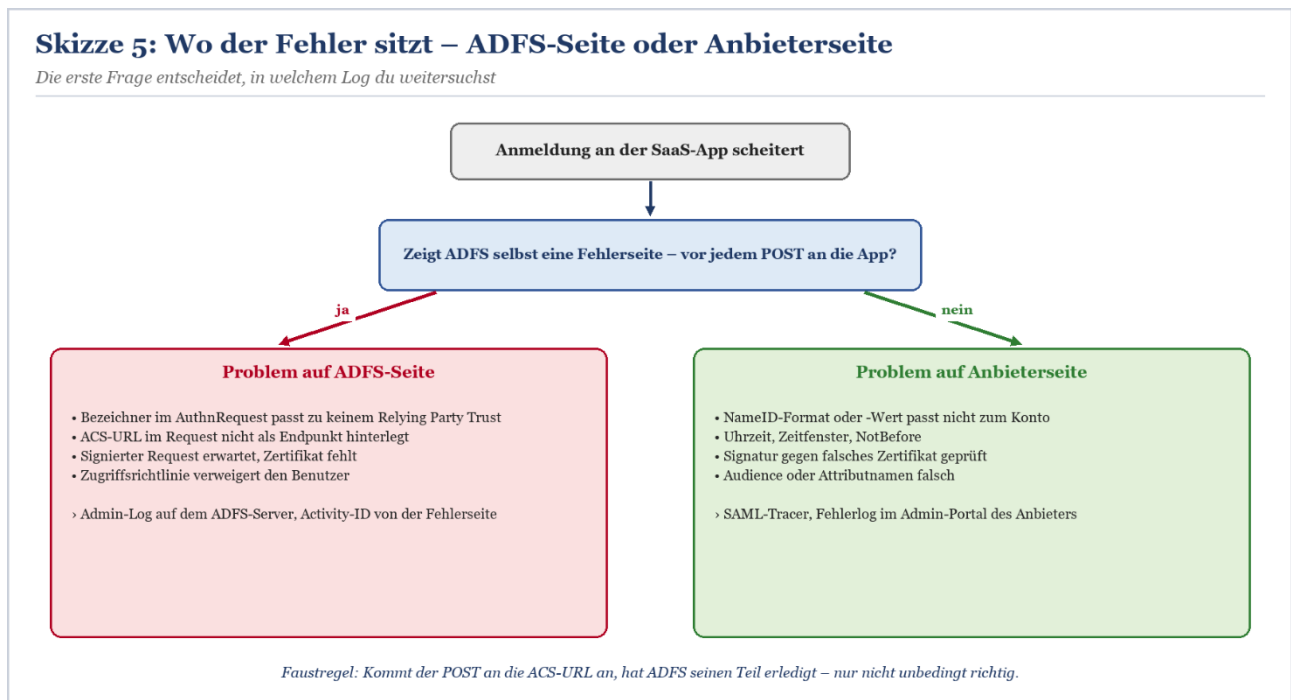
Die Prüfreihefolge

Bei der dekodierten Response gehst du am besten immer in derselben Reihenfolge vor, von außen nach innen. Das klingt pedantisch, spart aber Zeit, weil die Fehler in den äußeren Feldern alle inneren verdecken.

Schritt	Was du prüfst	Erwarteter Wert im Beispiel
1	Status der Response	urn:oasis:names:tc:SAML:2.0:status:Success
2	Destination und Ziel des POST	https://contoso.zeitwerk.example/saml/acs
3	Issuer	http://sts.contoso.de/adfs/services/trust
4	Audience in den Conditions	https://contoso.zeitwerk.example
5	NotBefore und NotOnOrAfter gegen die aktuelle Uhrzeit	Fenster schließt die Gegenwart ein, Standard 60 Minuten
6	NameID: Format und Wert	emailAddress, anna.berg@contoso.de
7	Attributnamen und -werte	exakt so benannt, wie die App sie mappt
8	Signatur: wo und mit welchem Algorithmus	an der Assertion, rsa-sha256

Tabelle 6: Die Prüfreihefolge für eine dekodierte SAMLResponse

Kommt gar kein POST an die ACS-URL zustande, weil ADFS selbst eine Fehlerseite zeigt, liegt das Problem auf deiner Seite. Dann steht auf der Fehlerseite eine Aktivitäts-ID, und im Admin-Log auf dem ADFS-Server findest du zu dieser ID den eigentlichen Grund – häufig als Ereignis 364 mit einer Ausnahme, die ausnahmsweise wirklich sagt, was los ist, etwa ein unbekannter Bezeichner oder eine nicht registrierte ACS-URL. Wie du die Protokolle liest und das Debug-Tracing einschaltest, beschreibt [ADFS-Ereignisprotokolle lesen – Admin-Log, Debug-Tracing und Auditing](#).



Skizze 5: Die erste Diagnosefrage – Fehlerseite von ADFS oder Fehlerseite des Anbieters?

Die Fehlerbild-Tabelle

Die folgenden Fehlerbilder decken nach unserer Erfahrung den Großteil der hängenden SaaS-Anbindungen ab. Die drei Klassiker stehen oben: NameID, Uhrzeit, Signatur.

Fehlerbild	Was du im Tracer siehst	Ursache	Lösung
App meldet „User	NameID vorhanden, aber	Falsches Quellattribut	Regel 2 an die Vorgabe

not found“ oder legt ein neues, leeres Konto an	Wert oder Format anders als im App-Konto	(UPN statt mail), falsches Format, geänderte Adresse	anpassen; bei Adressänderungen Konto in der App mitziehen
ADFS-Antwort mit Fehlerstatus statt Success	Status ungleich Success, oft mit Hinweis auf die NameID-Policy	App fordert im AuthnRequest ein NameID-Format, das keine Regel ausstellt	Format im AuthnRequest prüfen und Regel 2 auf genau dieses Format setzen
App meldet „Assertion not yet valid“ oder „expired“	NotBefore liegt aus Sicht der App in der Zukunft oder das Fenster ist abgelaufen	Uhrzeitabweichung zwischen ADFS-Servern und Anbieter, selten zu kurze Token-Lebensdauer	Zeitsynchronisation der Farm prüfen; als Notbehelf - NotBeforeSkew mit kleinem Wert
App meldet „Signature validation failed“	Signatur vorhanden, Zertifikat in der Response weicht vom hinterlegten ab	Token-Signing-Zertifikat erneuert, Anbieter hat noch das alte	Neues Zertifikat oder Metadaten-URL beim Anbieter hinterlegen
App meldet „Response must be signed“	Signatur nur in der Assertion	Anbieter erwartet signierte Response	-SamlResponseSignature MessageAndAssertion setzen
App meldet falschen Algorithmus	SignatureMethod rsa-sha256	Ältere App kann nur SHA-1	Bevorzugt Update beim Anbieter; notfalls - SignatureAlgorithm auf rsa-sha1, dokumentiert
App meldet „Invalid audience“ oder „Invalid issuer“	Audience oder Issuer weicht um ein Zeichen ab	Schrägstrich am Ende, http statt https, Groß- und Kleinschreibung	Bezeichner des Trusts bzw. Eintrag beim Anbieter zeichengenau angleichen
ADFS-Fehlerseite, kein POST an die App	Nur der Request an /adfs/ls/ ist sichtbar	Unbekannter Bezeichner, ACS-URL nicht registriert, Zugriffsrichtlinie verweigert	Admin-Log zur Aktivitäts-ID lesen, Endpunkte und Richtlinie am Trust prüfen
Anmeldung klappt, Rollen fehlen	Kein role-Attribut oder falscher Attributname	Gruppenfilter greift nicht, App erwartet anderen Attributnamen	Filter in Regel 3b und Attributnamen mit dem Mapping der App abgleichen
Im Tracer sind keine Claims lesbar	Nur EncryptedAssertion	Verschlüsselung über das Zertifikat aus den SP-Metadaten aktiv	Kein Fehler – Fehlerlog des Anbieters nutzen oder im Test kurz -EncryptClaims \$false

Tabelle 7: Fehlerbilder bei SaaS-Anbindungen und ihre Lösung

WARNUNG — NotBeforeSkew ist ein Pflaster, kein Uhrwerk

Mit -NotBeforeSkew verschiebt ADFS den Beginn des Gültigkeitsfensters etwas in die Vergangenheit. Das hilft, wenn die Uhr des Anbieters ein wenig nachgeht. Gehen aber deine eigenen ADFS-Server falsch, weil die Zeitsynchronisation in der Domäne klemmt, haben bald auch Kerberos und jede andere Anwendung ein Problem. Erst die Uhr richten, dann über den Puffer nachdenken.

TIPP — Ein Testbenutzer, der alles darf, testet nichts

Teste mit mindestens drei Konten: einem normalen Benutzer, einem App-Administrator mit der Rollengruppe und einem Konto ohne Berechtigung, das abgewiesen werden muss. Dazu ein Konto mit Umlaut im Namen – „Jürgen Weiß“ hat schon manche Just-in-Time-Anlage zum Stolpern gebracht.

Wenn du das Fehlerbild nicht in der Tabelle findest, hilft der Blick in die allgemeine Fehlersuche: Viele Ursachen haben gar nichts mit der einzelnen App zu tun, sondern mit Zertifikaten, Kerberos oder dem Web Application Proxy. Die häufigsten Ursachen sind in [ADFS-Anmeldung schlägt fehl – die häufigsten Ursachen und ihre Lösung](#) zusammengetragen.

FAKTEN — Und die Zukunft dieser App?

Diese Anbindung läuft auf ADFS stabil, solange du sie pflegst. Planst du mittelfristig den Umzug der Anwendungen nach Entra ID, ist eine sauber dokumentierte SaaS-Anbindung der dankbarste Kandidat: NameID-Quelle, Format, Attributnamen und Signaturvorgaben lassen sich eins zu eins übertragen. Wie das konkret geht, zeigt [SAML-Anwendungen von ADFS nach Entra ID migrieren](#). Die Bestandsaufnahme aller Relying Party Trusts davor beschreibt [Relying Parties inventarisieren – die Bestandsaufnahme vor der Entra-Migration](#).

Fazit: Zwei XML-Dateien und eine Handvoll Entscheidungen

Eine SaaS-Anwendung per SAML an ADFS anzubinden, ist technisch keine Raketenwissenschaft. Die Metadaten sind in Minuten getauscht, der Relying Party Trust ist mit einem Befehl angelegt. Die eigentliche Arbeit steckt in den Entscheidungen, die im PDF des Anbieters nicht stehen: Welches Attribut wird zur NameID, und was passiert, wenn es sich ändert? Welche Gruppen verlassen dein Haus? Wer bekommt überhaupt Zugriff? Und wie merkst du, dass der Anbieter nach deinem nächsten Zertifikatswechsel noch das alte Zertifikat hat?

Wer diese Fragen am Anfang beantwortet, mit SP-initiierteter Anmeldung als Standard arbeitet und beim Testen einen SAML-Tracer statt eines Horoskops benutzt, hat eine Anbindung, die jahrelang unauffällig läuft – und sich später ohne Drama nach Entra ID umziehen lässt. Wenn du eine Reihe solcher Anbindungen aufräumen, standardisieren oder für den Umzug vorbereiten willst, unterstützen wir dich im [Consulting zu ADFS \(Active Directory Federation Services\)](#). Wer SAML lieber im Labor zerlegt, bevor die Produktion es tut, ist in den [ADFS-Schulungen](#) richtig. Und die Arbeit mit Relying Party Trusts, Claims und Tokens vertieft das Buch – mehr dazu unter [ADFS in der Praxis – das Buch im Detail](#).

WEITER — Passend zum Thema

- › [Relying Party Trust anlegen – per Metadaten und von Hand](#) – der Trust im Detail, mit allen Varianten
- › [ADFS Claim Rules Leitfaden](#) – die Regelsprache hinter NameID und Attributen
- › [Access Control Policies in ADFS – Zugriffsregeln ohne Claim-Rule-Akrobatik](#) – den Zugriff auf die App sauber einschränken
- › [ADFS Token-Signing-Zertifikat erneuern – mit und ohne AutoCertificateRollover](#) – damit der Anbieter beim Zertifikatswechsel nicht aussteigt
- › [OAuth 2.0 und OpenID Connect mit ADFS Application Groups](#) – wenn die App statt SAML lieber OpenID Connect spricht
- › [SAML-Anwendungen von ADFS nach Entra ID migrieren](#) – der Weg der App nach Entra ID

FAQ: SAML-Anwendung an ADFS anbinden

Wie binde ich eine SAML-Anwendung an ADFS an?

Du tauschst Metadaten mit dem Anbieter, legst einen Relying Party Trust mit Bezeichner und ACS-URL an, stellst über Ausstellungsregeln NameID und Attribute aus, weist eine Zugriffsrichtlinie zu und testest die Anmeldung mit einem SAML-Tracer.

Wo finde ich die Metadaten meiner ADFS-Farm?

Unter /FederationMetadata/2007-06/FederationMetadata.xml auf dem Namen deines Federation Service, im Beispiel auf sts.contoso.de. Die Datei enthält Bezeichner, Endpunkte und das Token-Signing-Zertifikat.

Welches NameID-Format soll ich verwenden?

Das, was der Anbieter verlangt – meist E-Mail-Adresse oder Unspecified. Entscheidend ist, dass das Quellattribut eindeutig und möglichst stabil ist. Kann die App mit Persistent umgehen, ist das fachlich die robusteste Wahl.

Warum kommt in der Assertion keine NameID an?

Meist fehlt die Transformationsregel, die einen Claim in den Name-ID-Claim mit Format-Eigenschaft umwandelt, oder das Quellattribut ist beim Benutzer leer, etwa ein fehlendes mail-Attribut.

Wie sende ich AD-Gruppen an eine SaaS-Anwendung?

Ermittle die Gruppen mit einer add-Regel über tokenGroups und stelle mit einer zweiten Regel nur die Gruppen aus, deren Name einem festen Präfix folgt, typischerweise als Rollen-Claim.

Was ist der Unterschied zwischen SP- und IdP-initiiertter Anmeldung?

SP-initiiert startet die Anmeldung in der App mit einem AuthnRequest, auf den sich die Antwort bezieht. IdP-initiiert startet bei ADFS, und die App erhält eine unaufgeforderte Antwort. SP-initiiert ist der empfohlene Standard.

Wie aktiviere ich die Seite idpinitiatedsignon.aspx?

Mit Set-AdfsProperties -EnableIdpInitiatedSignonPage \$true. Seit ADFS unter Windows Server 2016 ist die Seite standardmäßig deaktiviert.

Wie lese ich eine SAMLResponse aus?

Mit den Entwicklertools des Browsers im Reiter Netzwerk, bei aktiviertem Beibehalten des Protokolls, oder komfortabler mit einer SAML-Tracer-Erweiterung, die die Nachricht direkt dekodiert anzeigt.

Was tun bei „Assertion not yet valid“?

Zuerst die Zeitsynchronisation der ADFS-Server prüfen. Geht die Uhr des Anbieters nach, hilft als Notbehelf ein kleiner Wert für -NotBeforeSkew am Relying Party Trust.

Die Signaturprüfung schlägt nach einem Zertifikatswechsel fehl – was nun?

Der Anbieter prüft noch gegen das alte Token-Signing-Zertifikat. Hinterlege das neue Zertifikat oder besser die Metadaten-URL deiner Farm beim Anbieter, damit er künftige Wechsel selbst übernimmt.