

S.A.F.E.-Feinkonzept - Dokument 4: Dokumentation Proof-of-Concept

Thema:	Secure Access to Federated E-Justice/E-Government
Verantwortlich:	Bund-Länderkommission für Datenverarbeitung und Rationalisierung in der Justiz
Version.Release:	1.3
Erstellt am:	03.11.2007
Zuletzt geändert am:	13.04.2008
Zustand:	in Bearbeitung / vorgelegt / <u>fertiggestellt</u>
Anzahl der Seiten:	24
Autoren:	Harald Krause (Dataport)
Dateiname:	20080331_SAFE_Dokument4_Proof-of-Concept_V1-3_veroeff.doc
Zusammenfassung:	Beschreibung der Komponenten und Dokumentation der Ergebnisse des Proof-of-Concepts
Anfragen/Hinweise:	BLK-AG „IT-Standards in der Justiz“ <ul style="list-style-type: none">• Jürgen Ehrmann (Justizministerium Baden-Württemberg) Telefon: 0711 279-2142 ehrmann@jum.bwl.de• Meinhard Wöhrmann (Oberlandesgericht Düsseldorf) Telefon: 0211 4971-647 meinhard.woehrmann@olg-duesseldorf.nrw.de
Status:	Freigegeben durch die BLK am 8. Mai 2008

Urheber- und Kennzeichenrecht

Das vorliegende Dokument wurde von Dataport, Anstalt des öffentlichen Rechts im Auftrag der Bund Länder Kommission für Datenverarbeitung und Rationalisierung in der Justiz – kurz BLK – vertreten durch das Justizministerium Baden-Württemberg erstellt. Sämtliche Inhalte sind urheberrechtlich geschützt. Die Verwendung, die Weitergabe oder Auswertung, die Vervielfältigung, Veröffentlichung oder Bearbeitung des Dokuments und/oder seiner Inhalte bedürfen der schriftlichen Genehmigung der BLK und sind mit einer Quellenangabe zu versehen.

Alle innerhalb des Dokuments genannten und ggf. durch Dritte geschützten Marken- und Warenzeichen unterliegen uneingeschränkt den Bestimmungen des jeweils gültigen Kennzeichenrechts und den Besitzrechten der jeweiligen eingetragenen Eigentümer. Allein aufgrund der bloßen Nennung im Dokument ist nicht der Schluss zu ziehen, dass Markenzeichen nicht durch Rechte Dritter geschützt sind.

Haftungsausschluss

Informationen und Verweise auf genutzte Standards und Normen, wurden nach bestem Wissen und Gewissen sorgfältig zusammengestellt und geprüft. Es wird jedoch keine Gewähr - weder ausdrücklich noch stillschweigend - für die Vollständigkeit, Richtigkeit, Aktualität oder Qualität und jederzeitige Verfügbarkeit übernommen.

Inhalt

1	AUSGANGSSITUATION UND ZIELSETZUNG	4
2	ERSTELLTE SOFTWARE-KOMPONENTEN	5
2.1	REALISIERTE KONZEPTTEILE	5
2.2	IMPLEMENTIERUNGSBASIS.....	5
2.3	BESCHREIBUNG.....	6
2.3.1	Security-Token-Service (STS).....	6
2.3.2	Attribute-Service/SPML-Provider	7
2.3.3	Identity-Store.....	7
2.3.4	Client-API	8
2.3.5	Test-Client.....	8
2.3.6	Lasttreiber	9
2.4	DEPLOYMENT-EINHEITEN.....	11
3	ERFAHRUNGSBERICHT	12
4	ERGEBNISSE DER LEISTUNGSMESSUNGEN	13
4.1	TESTDATENBANKBESTAND	13
4.2	TEST-HARDWARE.....	14
4.3	SIMULATIONS- UND MESSMETHODE	14
4.4	ERGEBNISSE ZU ANTWORTZEITEN UND DURCHSATZ	15
4.4.1	Minimale Antwortzeit	15
4.4.2	Durchsatz	16
4.4.3	Antwortzeiten in Abhängigkeit von Treffermenge	18
4.4.4	Bandbreitenabschätzung.....	19
4.5	ZUSAMMENFASSUNG	20
5	ANHANG	21
5.1	WSDL DER ATTRIBUTE-SERVICE-IMPLEMENTIERUNG (SPML-QUERY-PROVIDER)....	21
5.2	ABBILDUNGSVERZEICHNIS	24
5.3	TABELLENVERZEICHNIS.....	24
5.4	REFERENZIERTE DOKUMENTE	24

1 Ausgangssituation und Zielsetzung

Das Vorgehen zur Erstellung des S.A.F.E.-Feinkonzepts sieht vor, die Eignung der im Konzept festgelegten Architekturvorgaben durch eine begleitende exemplarische Implementierung im Sinne eines Proof-of-Concept zu validieren. Die zu diesem Zweck entwickelten Software-Komponenten decken die Funktionalität des S.A.F.E. weder in der Breite (Anwendungsfälle) noch in der vertikalen Detaillierung vollständig ab. Die Komponenten stellen daher keine erste Ausbaustufe eines künftigen S.A.F.E.-Systems dar und werden nicht mit dem Ziel erstellt, eine Basis für die Entwicklung eines produktionsfähigen Systems zu stellen.

Primär verfolgt das Proof-of-Concept das Ziel, diejenigen Teile zu implementieren, die aufgrund des technologisch innovativen Charakters schwerer einzuschätzen und daher kritisch sind. Code-Teile, deren Erstellung zwar zeitaufwendig, technisch aber als unkritisch betrachtet werden, werden nicht oder nur rudimentär bzw. schlicht implementiert.

Auch stellen die Proof-of-Concept Komponenten weder eine Referenzimplementierung noch ein verbindliches Muster für Entwickler dar. Beispielsweise ist mit der im Proof-of-Concept getroffenen Entscheidung für ein Entwicklungs-Framework keine Vorgabe oder Empfehlung für künftige Entwicklungen verbunden. Auf der anderen Seite können die Programmquellen aus dem Proof-of-Concept aber durchaus eine Hilfestellung für Entwickler oder zur Kosten- und Aufwandsschätzung liefern.

Dagegen sollen die konkreten Nachrichten, die von den Software-Komponenten erstellt oder interpretiert werden, Referenzcharakter besitzen. Sie weisen den größten Detaillierungsgrad der Schnittstellenbeschreibungen auf.

2 Erstellte Software-Komponenten

Im Rahmen des Proof-of-Concepts sind insgesamt sechs Komponenten bzw. Subsysteme erstellt worden, die im Wesentlichen die entscheidenden Schnittstellen zu WS-Trust und SPML implementieren bzw. nutzen.

2.1 Realisierte Konzeptteile

Für das P-o-C stand der Attribute-Service mit den Operationen zum Suchen von Kommunikationsteilnehmern (Adressbuch-Funktionalität) im Vordergrund. Es sind die *read-only* Operationen *lookup*, *search* und *listTargets* des SPML-2.0-Providers implementiert worden.

Zum Zwecke der Authentisierung und Autorisierung ist ein Security-Token-Service (STS) gemäß Feinkonzept implementiert worden, der auf X.509-signierte Anfragen Security-Token als SAML-1.1-Assertions herausgibt.

2.2 Implementierungsbasis

Der Auftraggeber hat aus Gründen des Investitionsschutzes als Zielplattform für die P-o-C-Komponenten den Java-basierten Tomcat-Application server (Servlet-Container) unter Linux vorgegeben, da der aktuelle EGVP-Registrierungsserver diese Technik verwendet und beabsichtigt ist, für eine künftige Lösung, Infrastruktur (insbesondere Hardware und Betriebssysteme) wiederzuverwenden.

Zur beschleunigten Implementierung der Web Services, insbesondere der Protokolle zum WS-Stack (WS-Addressing, WS-Security, WS-Trust), sind Frameworks unabdingbar. Für Java existieren mehrere freie oder Open-Source-Frameworks zum WS-Stack, z. B. „Metro“ von Sun Microsystems und „Axis2“-basierende Frameworks von der Apache Software Foundation. Eine qualifizierte vergleichende Bewertung wurde aus Zeitgründen nicht vorgenommen. Aufgrund der hohen Verbreitung und Akzeptanz der Axis2-Plattform ist dies als Basis für die P-o-C-Implementierung gewählt worden:

- Apache Axis2 1.3 als Web Service Basis-Framework inkl. WS-Addressing
- Axis2-Module Rampart 1.3 (mit WSS4J) für WS-Security und WS-Policy
- Axis2-Module Rahas 1.3 für WS-Trust

Als Entwicklungs- und Laufzeitumgebungen wurden folgende Komponenten verwendet:

- Java JDK 1.5.0_12

- Apache Tomcat 6.0.13 Server
- MySQL 5.0

2.3 Beschreibung

In den folgenden Abschnitten werden die im P-o-C implementierten Komponenten kurz dargestellt.

2.3.1 Security-Token-Service (STS)

Das System STS nach WS-Trust 1.3 stellt im S.A.F.E.-Feinkonzept die zentrale Infrastrukturkomponente zur Gewährleistung der Authentizität von Identitäten dar. Der STS muss spezifikationskonform komplexe Nachrichten interpretieren (*RequestSecurityToken*) und generieren (*RequestSecurityTokenResponse*).

Für das Proof-of-Concept ist ein STS-Service implementiert worden, der zwar hinsichtlich Konfigurationsfähigkeit rudimentär ist (hard-coding), aber die Schnittstelle zur Token Herausgabe (Action *issue*) korrekt bedient. Folgende technische Anforderungen bestehen an die Proof-of-Concept Implementierung:

- Nachrichten RST und RSTR gemäß WS-Trust Februar 2005
- RequestType: *http://schemas.xmlsoap.org/ws/2005/02/trust/Issue*
- Request-Credentials: X.509-Authentisierungszertifikat.
- Auszustellendes Token: SAML-Assertion V1.1
- Zugriff auf Identitätsdatenbank gemäß EGVP-Registrierungsserver

Folgendes Verhalten des anfragenden Clients und des STS ist realisiert worden:

- Client erzeugt RST-Nachricht mit `<AppliesTo>`-EndpointReference des Attribute-Service und signiert sie mit seinem X.509-Software-Zertifikat und verschlüsselt sie mit dem Zertifikat des STS.
- STS nimmt RST-Nachricht entgegen, entschlüsselt sie und prüft Signatur.
- STS prüft, ob das X.509-Zertifikat, mit dem die Nachricht signiert wurde, als Authentisierungszertifikat in der Identitätsdatenbank hinterlegt ist (Authentifizierung).
- STS erzeugt bei erfolgreicher Authentifizierung signierte SAML-Assertion mit dem eindeutigen subject-ID (Govello-ID) und Attributen zu Name, und Rolle (egvp_buerger, egvp_backend, egvp_slave) aus der Identitätsdatenbank.

- STS schickt SAML-Assertion als Security-Token in der RSTR-Nachricht zurück. Die Nachricht ist verschlüsselt mit dem X.509-Zertifikat des Clients.
- Client nimmt RSTR-Antwort entgegen, entschlüsselt sie und extrahiert die SAML-Assertion zur Verwendung im nachfolgenden Aufruf des Attribute-Service.

2.3.2 Attribute-Service/SPML-Provider

Der Attribute-Service ist gemäß Feinkonzept als SPML-Provider realisiert, der die nur-lesenden SPML-Operationen *lookup*, *search* und *listTargets* implementiert. Der Service ist als Axis2-Komponente (.aar-Datei/Axis2-Archive) ausgehend von der im Anhang befindlichen WSDL-Datei implementiert, d. h. die Code-Rümpfe sind mit dem Axis2-Tool WSDL2Java generiert worden.

Der Attribute-Service nimmt SPML-Requests entgegen, die vom Requestor mit dem symmetrischen Schlüssel des SAML-Tokens signiert und verschlüsselt sind. Er wertet das Attribut „RoleID“ aus dem SAML-Token aus und beschränkt in Abhängigkeit von dessen Wert die Sichtbarkeit (Selektion) gemäß der Rollen-/Sichtbarkeitsmatrix des IT-Feinkonzepts von S.A.F.E.

Als XML-Schema für die SPML-Provisioning-Service-Objects (PSO) ist gemäß Feinkonzept das Liberty *ID-SIS Personal Profile* mit der Erweiterung für OSCI 1.2 Postfächer verwendet worden.

Als Anfragesprache werden Comparison-Ausdrücke gemäß XPath 2.0 - wie von SPML 2.0/xsd-Profilen empfohlen - verwendet. Implementiert sind Anfragen unter Nutzung der Operatoren „=“, „<“, „>“ sowie den XPath-Funktionen *contains()*, *starts-with()* und *ends-with()* für Text-Attribute. Die P-o-C-Implementierung verarbeitet XPath-Ausdrücke mit einer beliebigen Anzahl von UND- oder OR-verknüpften Teilausdrücken zu beliebigen Attributen des Identity-Stores. Die Implementierung wertet nicht das Element `<spml:namespacePrefixMap>` aus, sondern realisiert ein hart-kodiertes Mapping von XML-Präfixen.

Die im Abschnitt zu Filter-Klauseln des S.A.F.E.-Konzepts (Abschnitt 4.2.3.4 im S.A.F.E.-Basiskonzept) aufgeführten Beispiele zu XPath-Filterausdrücken werden durch die P-o-C-Implementierung des Attribute-Services unterstützt.

2.3.3 Identity-Store

Die im P-o-C verwendete Identitätsdatenbank ist hinsichtlich Datenmodell und Datenbank-Technologie (MySQL) identisch. Die Befüllung ist per Skript mit einem fiktiven

Bestand und zu diesem Zweck erzeugten Zertifikaten für Verschlüsselung und Authentisierung erfolgt.

2.3.4 Client-API

Die P-o-C-Implementierungen des SPML-Testclients und Lasttreibers stützen sich zur Kommunikation mit dem SPML-Provider (Attribute-Service) auf eine Fassadenklasse, die SOAP-Aufrufe der SPML-Operationen durch vereinfachte Methoden gestattet. Die Implementierung fasst im Wesentlichen Client-Funktionalitäten des Axis2-, Rampart- und Rahas-Frameworks zusammen.

Die Klasse `de.dataport.spml.query.client.SPMLClientFacade` besitzt folgende öffentliche Methoden und Konstruktoren:

- ```
public SPMLClientFacade(ConfigurationContext ctx,
 String stsEndpoint,
 Policy stsPolicy
 String serviceEndpoint,
 Policy servicePolicy)
```
- ```
public Set<String> listTargets()
```
- ```
public PPType lookup(String subjectNameID,
 boolean fetchEverything)
```
- ```
public List<PPType> search(String xPath)
```
- ```
public void updateToken(boolean always)
```

Nähere Informationen zur Programmierschnittstelle sind in der Dokumentation zum P-o-C-Programmcode („Javadoc-API“) enthalten.

### 2.3.5 Test-Client

Als Testclient dient ein einfaches Java-Konsolenprogramm. Mit Hilfe von Kommandozeilenparametern können die Operationen *lookup*, *search* und *listTargets* gegen den Attribute-Service ausgeführt werden. Zusätzlich bietet der Test-Client ein schlichtes Konsolen-Menü, mit dessen Hilfe – bei bestehender SecureConversation-Session zum Attribute-Service – beliebige Requests bestehend aus einem der drei Operationen ausgeführt werden können. Dieser Modus entspricht der typischen Situation eines gestarteten und authentisierten EGVP-Clients.

## 2.3.6 Lasttreiber

Zur Simulation von Last am Attribute-Service und STS kann der SPML-Testclient per Kommandozeilenschalter als Lasttreiber fungieren. Über eine Konfigurationsdatei kann ein Lastprofil definiert werden. Es lässt sich die Anzahl gleichzeitiger Clients, die „Feuerrate“ (Requests pro Minute), das Verhältnis von *search*- zu *lookup*-Operationen, die beabsichtigte Treffermenge von *search*-Operationen sowie die zufällige, zeitliche Streuung der Requests konfigurieren (siehe Abbildung 1).

Der Lasttreiber startet eine konfigurierbare Anzahl von nebenläufigen Threads, die jeweils einen SPML-Client simulieren. Jeder Client-Thread protokolliert alle Operation mit Antwortzeit in msec, Treffermenge und Erfolgsstatus in eine gemeinsame Log-Datei.

```
#
Parameter für SPML-Lasttreiber
#
AXIS2-Clientconfiguration-Directory
default: ClientRepository
ctx.directory =ClientRepository

STS-Endpoint URL
Kann durch Aufrufargumente überschrieben werden.
default: http://localhost:8000/axis2/services/STS
sts.endpoint =http://10.1.4.195:8080/axis2/services/STS

SPML-Provider-Endpoint URL
Kann durch Aufrufargumente überschrieben werden.
service.endpoint =http:// 10.1.4.195:8080/axis2/services/SPMLQueryService

STS-Policy-Datei
Kann durch Aufrufargumente überschrieben werden.
default: null
sts.policyfile =resources/STSPolicy.xml

SPML-Provider-Policy-Datei
Kann durch Aufrufargumente überschrieben werden.
default: .\SPMLPolicy.xml
service.policyfile =resources/SPMLPolicy-SAML.xml

Anzahl gleichzeitiger Client-Threads
#default: 1
concurrent.threads =64

Anzahl von lesenden Requests pro Client-Thread
#default: 100
max.loops =100

Verhältnis von "search"- zu "lookup"-Requests in Prozent gesteuert durch
Wahrscheinlichkeit
(z. B. bedeutet "80", dass im Mittel 80% search- und 20% lookup-Requests
generiert werden)
Ein Wert kleiner Null bedeutet, dass weder search- noch lookup-Request aus-
geführt werden,
sondern nur Token-Issue-Requests (unabhängig vom Wert "token.renew", s.u.)
#default: 50
search.ratio =50
```

```
EGVP-User-Typ ("client", "backend", "slave")
default: client
user.type =backend

Nummernrepräsentanten für User-Namen. Der Nummer "5" wird z. B. der Username "00005Name" und
der Keystore-Dateiname "keystore_backend_20.p12" zugeordnet. Der Keystore-Dateiname wird gebildet aus
'keystore_<client.type>_'<N>'.p12' mit N = (((n-1)/20)+1)*20.
Wenn "concurrent.threads" größer als die Anzahl aller Nummern ist, werden die Nummern wiederholt.
Syntax: (<Nummer>|<von-Nummer>-<bis-Nummer>)[, (<Nummer>|<von-Nummer>-<bis-Nummer>),...]
Beispiel: 1,3,10-20,2020-2050,25000-25100
default: 1-10
user.numbers =1-500

Nummernbereich für User-Namen, über die gesucht wird.
Syntax: <von-Nummer>-<bis-Nummer>
default: 1-1000
user.search.range =1501-50000

Mittlere Anzahl von Requests pro Minute als Fließkommazahl
default: 10.0 also etwa alle 6 Sekunden
requests.per.minute =6.0

Steuert ob bzw. wann ein neues SAML-Token angefordert wird.
Syntax: (always|never|expired)
default: never
token.renew =expired

Streuung der Request-Intervalle (60/"requests.per.minute") in Prozent
"0" bedeutet, dass exakt nach Ablauf des Intervalls I gefeuert wird,
"50" bedeutet, dass zwischen 0,5 I und 1,5 I gefeuert wird und
"100" bedeutet, dass zwischen 0 und 2 I gefeuert wird.
default: 33
max.variance =50

Angestrebte bzw. erwartete Treffermenge einer SPML-Search Operation. In Abhängigkeit von diesem Wert
wird der XPath-Ausdruck mit dem Wildcard-Ausdruck angepasst. Es ist nur die Größenordnung (1,10,100,...)
relevant. Die tatsächliche Treffermenge hängt vom Datenbestand ab und kann abweichen.
default: 10
anticipated.hits.count =10

Log-Datei inkl. Pfad, in die die Messergebnisse aller Threads geschrieben werden.
default: ./SPML-Load.log
log.filename =/tmp/SPML-Load.log

Verzeichnis, in dem sich die keystore-Dateien befinden
default: .
keystore.dir =/tmp/p12Keystores

Flag für Consolen-Output ("true" oder "false")
default: true
do.console.output =true
```

Abbildung 1: Konfigurationsdatei zur Festlegung des Lastprofils

## 2.4 Deployment-Einheiten

Folgende Java-Deployment-Einheiten sind durch den Build-Prozess generiert worden:

- **STS.aar**  
Axis2-Archive für das Subsystem „Security-Token-Service“ (STS). Es ist im Verzeichnis `${tomcat.home}/webapps/axis2/WEB-INF/services` zu deployen.
- **SafeSecurityHandler.jar**  
Das Java-Archive enthält eine Implementierung eines *PolyBasedResultsValidator* gemäß Rampart-Framework und abhängige Klassen. Die Klasse führt den Zertifikats-*lookup* im Identity-Store durch und wird vom Axis2-Framework in der „Security-Phase“ aufgerufen.  
Das .jar-Archive ist unter `${tomcat.home}/webapps/axis2/WEB-INF/lib` zu deployen.
- **spmlQueryProvider.aar**  
Axis2-Archive für das Subsystem „SPML-QueryProvider“, also der Web Service, der die SPML-Operationen implementiert. Es ist im Verzeichnis `${tomcat.home}/webapps/axis2/WEB-INF/services` zu deployen.
- **liberty-sis-pp.jar**  
Das Java-Archive enthält die generierten Klassenrepräsentationen zum XML-Schema des *Liberty ID-SIS Personal Profile*. Das .jar-Archive wird sowohl vom SPML-QueryProvider als auch von den SPML-Clientsystemen benötigt.  
Es ist unter `${tomcat.home}/webapps/axis2/WEB-INF/lib` zu deployen.
- **SPMLClient.jar**  
Das Java-Archive enthält sowohl einen rudimentären Client zur interaktiven SPML-Abfrage als auch ein Treiberprogramm zur Simulation und Messung von Lastszenarien.  
Die *main*-Klasse ist `de.dataport.spml.query.client.SPMLClient`.

### 3 Erfahrungsbericht

Für das Proof-of-Concept ist als Web Service Framework Axis2 der Apache Software Foundation ausgewählt worden. Axis2 ist stark verbreitet und besitzt eine anerkannt gute Architektur, die das Framework flexibel und leistungsfähig macht. Der Dokumentationsstand für das Axis2-Framework selbst ist als ausreichend zu bezeichnen.

Das Axis2-Framework bietet allerdings nur Unterstützung für Basisfunktionalitäten zu Web Services. Es ist ausgelegt und entworfen für Erweiterungen – gerade zum Web Service-Stack. Aus dem WS-Stack bringt Axis2 selbst nur eine Unterstützung für WS-Addressing mit. Für die anspruchsvollen Protokolle WS-Security und WS-Trust sind bei Apache die Module Rampart (WS-Security und WS-Policy) und Rahas (WS-Trust) verfügbar. Hier ist die Situation hinsichtlich Dokumentation zu den Modulen deutlich ungünstiger.

Bei Rampart und Rahas hat es von Version 1.1 zu 1.3 erhebliche Änderungen gegeben, insbesondere der vollständige Umstieg auf WS-Policy anstelle von proprietären Konfigurationen. Allerdings sind Dokumentationen hierzu nahezu nicht vorhanden, so dass auf Basis der Programmquellen ein Reengineering betrieben werden musste, um entscheidende Teile des Frameworks nutzen zu können. Auch die Programmqualität ist noch nicht in einem Zustand, der als ausreichend bezeichnet werden kann.

Es wird gegenwärtig aber intensiv an den Modulen bei der Apache Software Foundation gearbeitet (im Zeitraum Juli bis September 2007 allein zwei Versionssprünge), so dass erwartet werden kann, dass in wenigen Monaten ausgereifere und besser dokumentierte Komponenten verfügbar sind. Ein Evaluieren anderer Frameworks - insbesondere Suns Metro – erscheint aber für eine Release-Entwicklung angebracht.

Festzuhalten ist, dass die entscheidenden Elemente des S.A.F.E.-Feinkonzepts bereits heute mit Frameworks umzusetzen sind, wenngleich noch mit höheren Programmieraufwänden. Die sicherheitskritischen Kernfunktionalitäten, die die kryptographischen Absicherungen der Nachrichten auf Basis von WS-Security (Signatur und Verschlüsselung mit X.509, SAML-Token und symmetrischen Session-Keys) gewährleisten, machen aber einen bereits ausgereifteren Eindruck.

Das Proof-of-Concept hat gezeigt, dass die fachlichen Anforderungen an ein Registrierungssystem als Web Service basiertes Identity Management mithilfe einer Profilierung existierender Standards sehr gut umsetzbar sind. Insbesondere die Kombination von SAML-Assertion und SPML sowie das WS-Trust-Binding für den Attribute-Service erweist sich in der exemplarischen Implementierung als sinnvolles und geschlossenes Modell.

## 4 Ergebnisse der Leistungsmessungen

Die Leistungs- und Lastmessungen hatten den Zweck, praktisch untermauerte Erfahrungen zu Antwortzeiten, Durchsatzverhalten und Datenvolumen für einen Attribute-Service zu gewinnen, der gemäß S.A.F.E.-Konzept auf Basis SPML 2.0 mit WS-Security-Binding realisiert ist. Die Messungen liefern eine Orientierung, erheben aber nicht den Anspruch, repräsentative und hinsichtlich ihres statistischen Ansatzes her belastbare Referenzergebnisse zu liefern. Dieser Anspruch kann aus mehreren Gründen nicht erfüllt werden:

- Die P-o-C-Implementierung ist dem Anspruch nach eine exemplarische Durchstich-Realisierung und nicht auf Performance optimiert (insbesondere wurde kein Laufzeit-Profiling durchgeführt).
- Der Bestand der Testdatenbank ist zwar hinsichtlich Volumen und Datensatzanzahl realistisch, es ist aber keine statistische Analyse bezüglich Streuung der Datenbankinhalte vorgenommen worden.
- Das Anfrageverhalten ist gegenwärtig weitgehend unbekannt bzw. beruht auf Annahmen und Abschätzungen.
- Bei der zur Lastmessung verwendeten Hardware handelt es sich um keine Server-Hardware wie sie in realen Szenarien eingesetzt werden würde.

Dennoch ist ein nicht unerheblicher Aufwand betrieben worden, um aussagefähige Ergebnisse zum Antwortzeitverhalten und Netzvolumen zu erhalten.

### 4.1 Testdatenbankbestand

Das relationale Datenbankschema ist identisch mit dem des aktuellen EGVP-Registrierungsservers. Das Datenbanksystem ist MySQL 5.0 (InnoDB, utf8-Zeichensatz).

Ausgehend von der aktuellen Nutzeranzahl im EGVP-Registrierungsserver (knapp 30.000) und einer Extrapolation der Nutzerzuwächse (unter 20.000 pro Jahr) bis zum Jahr 2009 sind als Bestandsgröße der P-o-C-Testdatenbank 50.000 Identitäten gesetzt worden. Der Datenbestand hat folgendes Profil:

- 500 Identitäten mit RoleID = „egvp\_behoerde“
- 1.000 Identitäten mit RoleID = „egvp\_slave“
- 48.500 Identitäten mit RoleID = „egvp\_buerger“
- 1.000 Städte

- 2.500 Organisationen
- 50.000 unterschiedliche Authentisierungs- und Verschlüsselungszertifikate
- 50.000 unterschiedliche Nachnamen („00001Name“ bis „50000Name“)

Die übrigen Datenbankfelder (Vorname, Adresse, Anrede, Land, Telefonnummern etc.) sind für jede Identität identisch.

## 4.2 Test-Hardware

Die Last- und Performance-Messungen sind auf folgenden Systemen durchgeführt worden:

Server (Identity-Provider/STS und Attribute-Service):

- Notebook SFC Lifebook E, Intel Core2 Duo, 2.2GHz, 4MB SLC, 800MHz FB
- 2 GB RAM
- Hard disk verschlüsselt
- Betriebssystem: Windows XP SP2

Lasttreiber-System:

- Notebook SFC Lifebook E, Intel Core2 Duo, 1.7GHz
- 2 GB RAM
- Hard disk verschlüsselt
- Betriebssystem: Windows XP SP2

Netzwerk:

- 1 Gigabit

## 4.3 Simulations- und Messmethode

Für jeden der Client-Threads werden Wartezeiten (zufällig gestreut) so generiert, dass im Mittel eine konfigurierte Request-Anzahl pro Minute eingehalten wird. Die Anzahl der Client-Threads multipliziert mit der Request-Anzahl bestimmt daher den beabsichtigten Durchsatz des Systems. Überschreiten Antwortzeiten (vereinzelt oder regelmäßig) die (zufällig variierende) Wartezeit, reduziert sich der tatsächliche Durchsatz – das System ist überlastet.

Das Verhältnis von *search*- zu *lookup*-Operationen ist konfigurierbar. Die *lookup*-Operationen greifen auf Identitäten mit IDs (Govello-ID) aus vorangegangenen *search*-Operationen zu. Die Treffermenge der *search*-Operationen ist konfigurierbar (geändertes Muster in XPath-Funktion *ends-with()*).

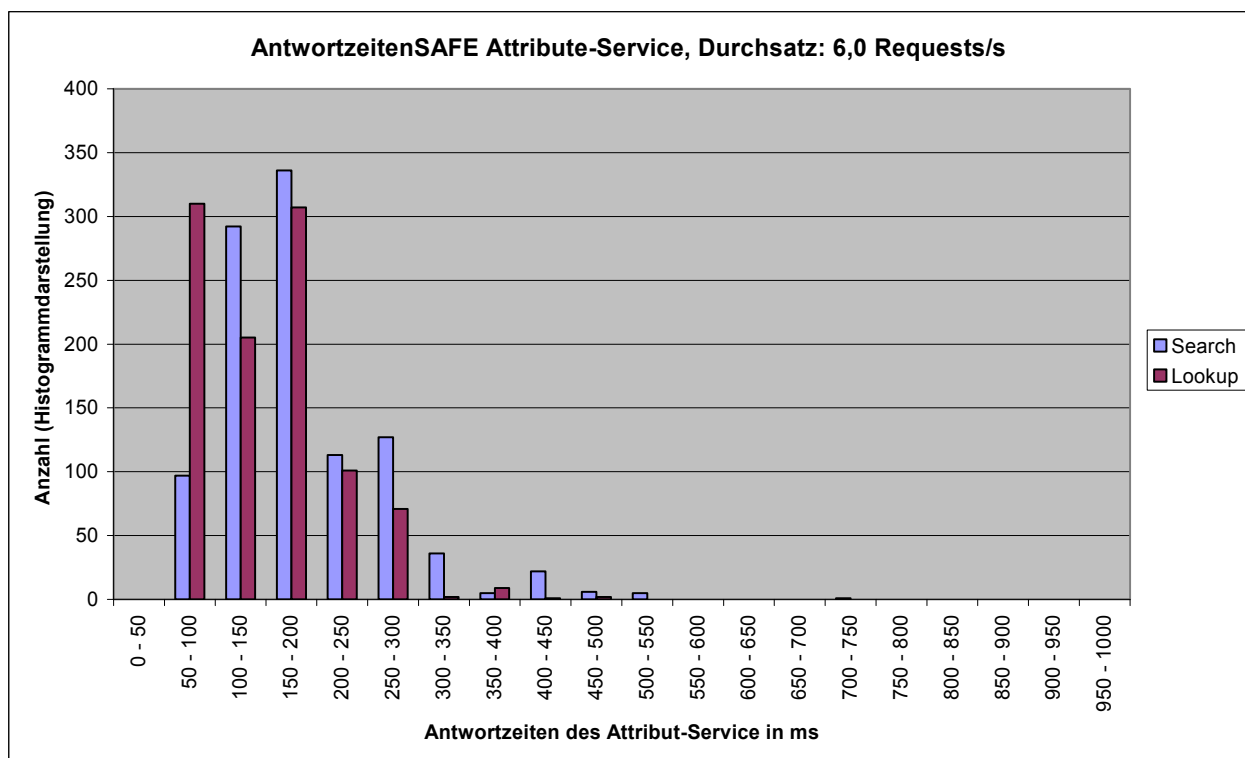
## 4.4 Ergebnisse zu Antwortzeiten und Durchsatz

In den konkret durchgeführten Nebenläufigkeitstests besaßen die Clients die fachliche Rolle „egvp\_behoerde“ und führten im Mittel je zur Hälfte *lookup* und *search*-Operationen durch. Die durchschnittliche Treffermenge der Suchen war gleich 10. Die Suche erfolgte zufällig gestreut über den gesamten Bestand von 48.500 Identitäten der Rolle „egvp\_buerger“.

Die privaten Schlüssel der generierten X.509-Software-Authentisierungszertifikate waren in PKCS#12-Keystore-Dateien zu je 20 Schlüsseln pro Datei dem Lasttreibersystem zugänglich. SAML-Token wurden durch die Clients nur erneut vom STS angefordert, wenn die Token nicht mehr gültig waren.

### 4.4.1 Minimale Antwortzeit

Zur Ermittlung der minimalen Antwortzeit bzw. der Roundtrip-Zeit von SPML-Anfragen sind zunächst Anfragen von nur einem Client-Thread an den Attribute-Service gemessen worden. Der Roundtrip wurde gemessen an der Programmierschnittstelle des Clients (siehe Abschnitt 2.3.4), d. h. die Zeiten für Konstruktion des Requests inkl. Verschlüsselung und Signatur sowie fürs Parsen des Responses inkl. Entschlüsselung und Signaturprüfung waren in der Messung mit eingeschlossen. Auf der für die Tests zur Verfügung stehenden Hardware ist als mittlere Roundtrip-Zeit 166 msec gemessen worden.



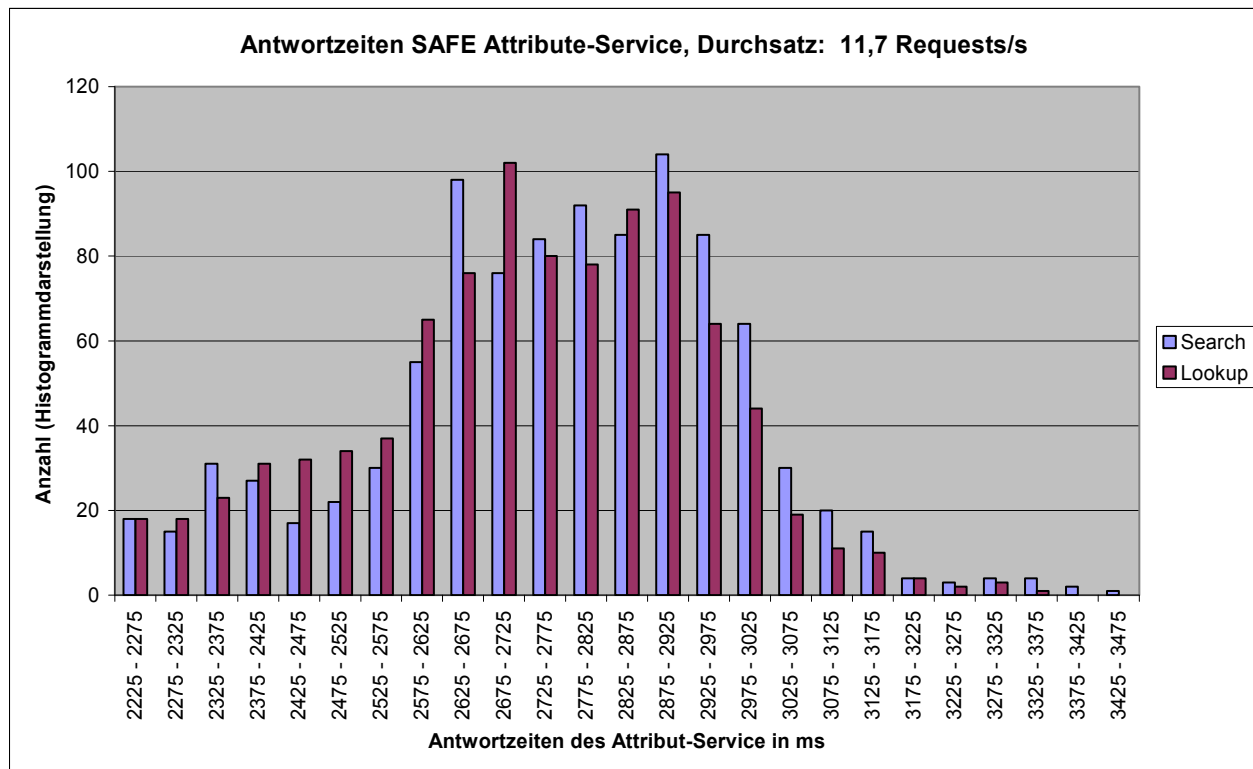
**Abbildung 2: Messung von 1024 Anfragen mit einem Client**

Der Testclient führte die Requests bei dieser Messung (Abbildung 2) in direkter Folge ohne Wartezeiten aus, d. h. der dabei erzielte Durchsatz ergibt sich direkt aus  $1/166\text{ms} = 6 \text{ Requests/s}$ . Dies entsprach in etwa der Hälfte des maximal zu erzielenden Durchsatzes von 11,7 Requests/s bei 32 gleichzeitigen Client-Threads.

#### 4.4.2 Durchsatz

Hinsichtlich Betrachtungen zum Durchsatzverhalten ist zunächst ermittelt worden, welchen maximalen Durchsatz das System auf der zur Verfügung stehenden Test-Hardware (siehe Abschnitt 4.2) erreichen konnte. Dazu ist schrittweise die Anzahl der Client-Threads verdoppelt worden. Wie bei der vorangegangenen Messung haben hierbei alle Clients ohne Wartezeit in direkter Folge die Requests „gefeuert“ („Burst“-Modus). Der jeweilige Durchsatz hat sich von 6 Requests/s bei einem Client-Thread auf 11,7 bei 32 Clients-Threads erhöht.

Da die Clients direkt hintereinander „feuerten“, verlängerten sich zwangsläufig die Antwortzeiten auf Anzahl Clients/Durchsatz (z. B.  $32/11,7=2,73$ , siehe Abbildung 3).



**Abbildung 3: Messung mit 32 Client-Threads im Burst-Modus**

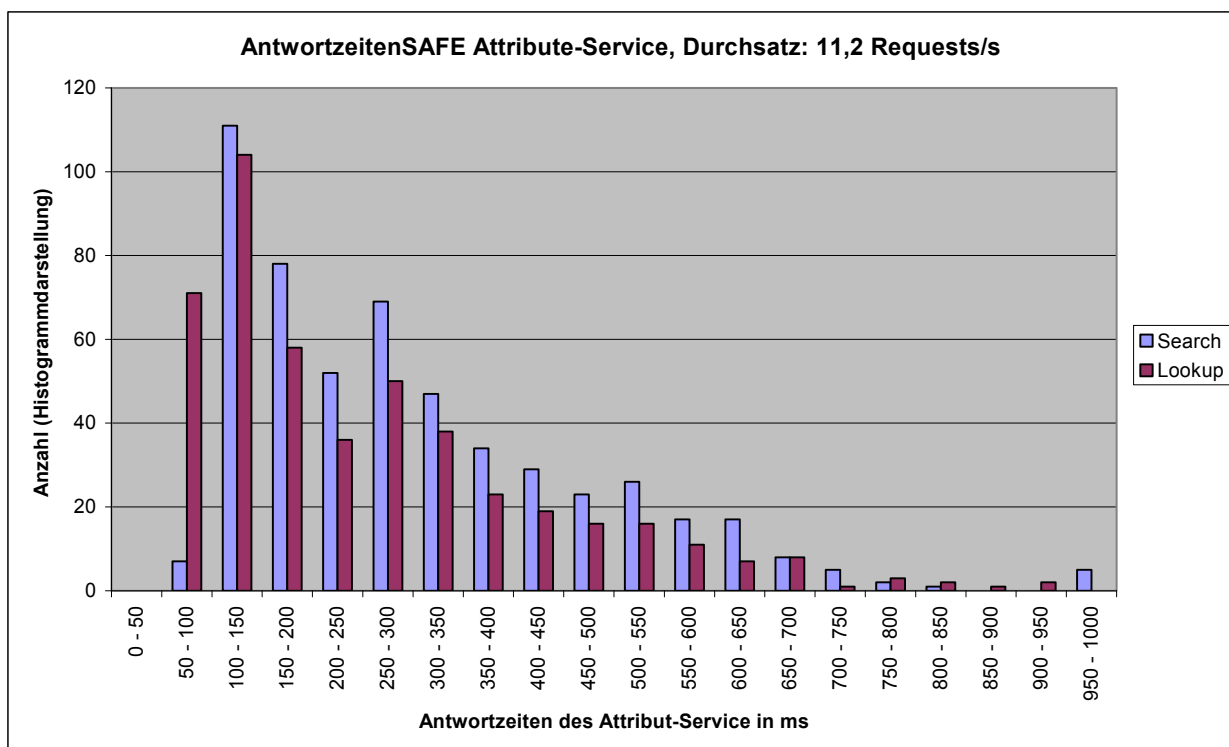
Eine weitere Erhöhung ließ den Durchsatz allmählich wieder sinken (9,5 Requests/s bei 128 Client-Threads), was allerdings auf eine Überlastung des Lasttreibersystems zurückzuführen war, dessen CPU-Auslastung 100% betrug. Würde die „Befeuern“ durch leistungsfähigere oder mehrere Lasttreibersysteme erfolgen, wäre eine konstante maximale Durchsatzrate zu erwarten.

Um eine bessere Einschätzung eines realistischen Durchsatzverhaltens zu erlangen, sind Messungen mit einer unterschiedlichen Anzahl von Client-Threads durchgeführt worden, die jeweils eine (zufällig gestreute) mittlere Request-Häufigkeit einhielten, so dass sich rechnerisch ein Durchsatz von 6 Requests/s ergab (Anzahl Clients \* Request-Häufigkeit = Durchsatz). Die Messungen ergaben bei insgesamt 2.000 Requests:

| Anzahl Client-Threads | Request-Häufigkeit je Client-Thread | Mittlere Antwortzeit (Roundtrip) |
|-----------------------|-------------------------------------|----------------------------------|
| 10                    | 36.0/min                            | 220 ms                           |
| 20                    | 18.0/min                            | 215 ms                           |
| 30                    | 12.0/min                            | 204 ms                           |
| 40                    | 9.0/ min                            | 227 ms                           |

**Tabelle 1: Antwortzeiten bei einem Durchsatz von 6 Request/s**

Als Ergebnis ergab sich das erwartete Verhalten, dass die Antwortzeiten bei gegebenem Durchsatz unabhängig von der Anzahl der Client-Threads in etwa konstant blieben (gut 0,2 s). Dieses Verhalten galt, solange der Attribute-Service nicht in einen Überlast-Bereich gefahren wurde. Aber selbst bei einer Last von 11,2 Requests/s (nahe dem Maximum) verlängerte sich die mittlere Antwortzeit um nur ca. 35%. Die Streuung der Antwortzeiten (Standardabweichung) wuchs dann allerdings etwa um den gleichen Prozentsatz (siehe Abbildung 4).



**Abbildung 4: Messung von 20 Client-Threads mit je 36 Request/min**

### 4.4.3 Antwortzeiten in Abhängigkeit von Treffermenge

Die oben beschriebenen Messungen sind mit Suchen, die Treffermengen von 10 Identitäten lieferten, durchgeführt worden. Die Ausführungszeit einer *search*-Operation hängt ab von der Anzahl der Treffer (Datenbankzugriff, Signatur und Signaturvalidierung, Ver- und Entschlüsselung, Transfer).

Zur Ermittlung der Abhängigkeit und zur Abschätzung absoluter Zahlen sind *search*-Anfragen, die 1, 10, 100 und 1000 Treffer lieferten, ausgeführt worden:

| Trefferanzahl | Antwortzeit |
|---------------|-------------|
| 1             | 0,2 s       |
| 10            | 0,3 s       |
| 100           | 0,7 s       |
| 1000          | 5,6 s       |

Tabelle 2: Gemittelte *search*-Antwortzeiten gemessen bei einem Client-Thread

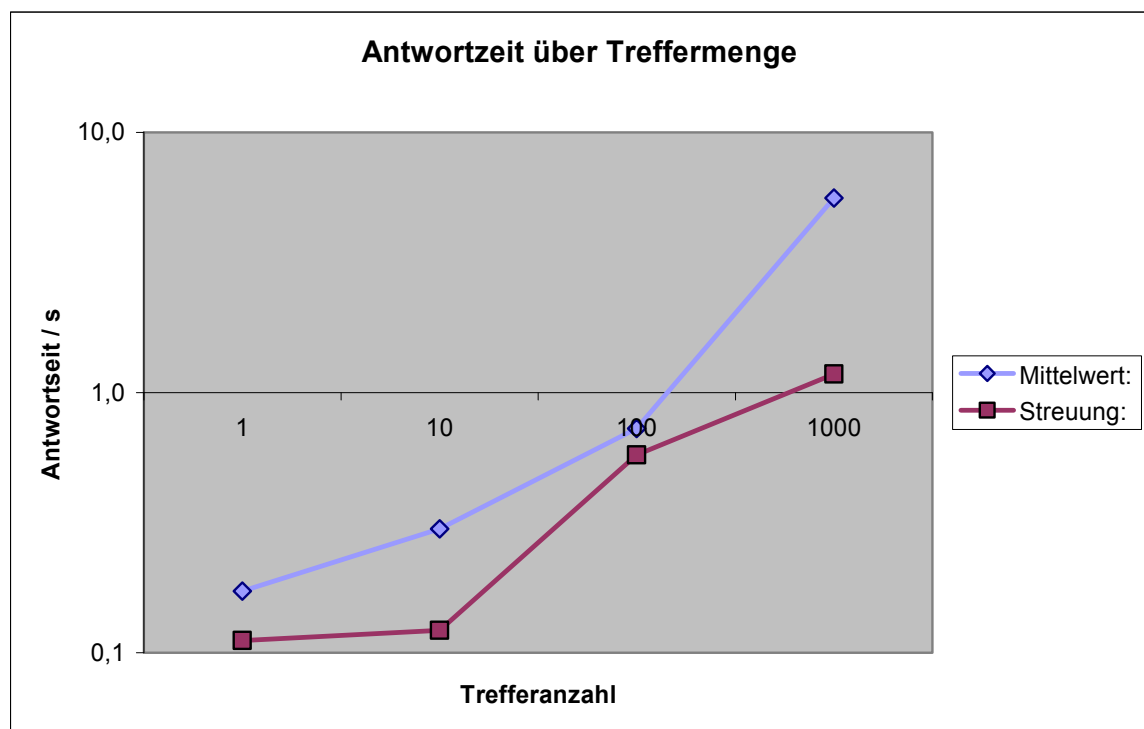


Abbildung 5: Logarithmische Darstellung der *search*-Antwortzeiten

#### 4.4.4 Bandbreitenabschätzung

Die Messungen zu Antwortzeiten und Durchsatz erfolgten in einem schnellen lokalen Netzwerk, so dass keine Einschränkungen durch Bandbreitenbegrenzungen auftraten. Zur Einschätzung eines Betriebes im Internet sind Aussagen zu Nachrichtenvolumen und daraus ableitbaren Bandbreitenbedarf zu treffen. Dazu wurden die Größen von Nachrichten – insbesondere der Antwortnachrichten von *search*-Anfragen – untersucht.

Aus der Untersuchung der Nachrichtengrößen von Suchantworten mit 1, 10, 100 und 1000 Treffern ergibt sich rechnerisch ein konstanter Overhead von ca. 4,4kB und ca. 1,2kB pro Trefferdatensatz. Bei Verwendung von WS-SecureConversation reduziert sich der Overhead um ca. 50%, da das SAML-Token nicht transportiert werden muss.

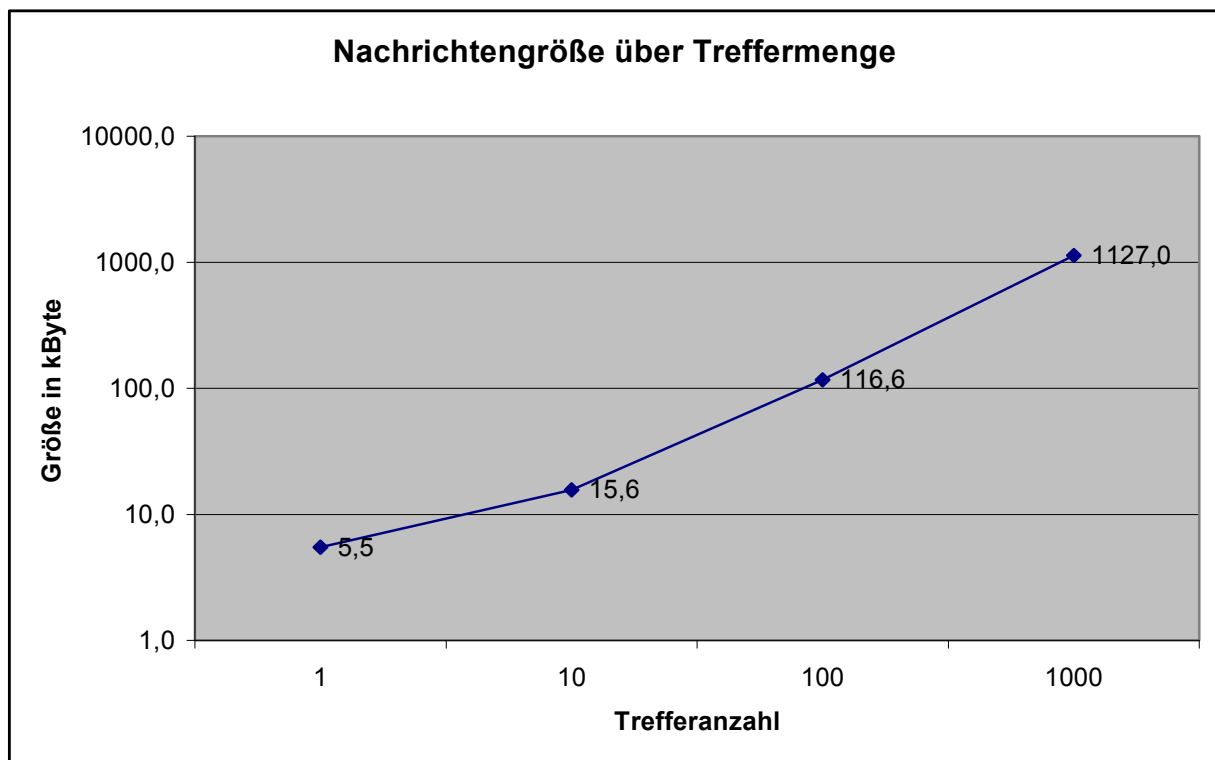


Abbildung 6: Größe von *search*-Antwortnachrichten

Die Antwortnachricht einer *lookup*-Operation betrug ca. 9kB, da hierbei zusätzlich die Attribute zum OSCI-Postfach, insb. die Zertifikate des Empfängers und OSCI-Intermediärs, mit übertragen werden.

Geht man für eine Bandbreitenabschätzung der Server-Internetanbindung davon aus, dass im Mittel gleich viele *search*- und *lookup*-Operationen genutzt werden, und setzt man als mittlere Trefferanzahl bei *search*-Operationen 25 an, so bleibt das Response-Datenvolumen für *lookup* und *search* zusammen unter 50kB. Eine Upstream-Bandbreite von 1MBit/s ließe daher einen Durchsatz von gut 4 Requests/s zu.

## 4.5 Zusammenfassung

Bei den durchgeführten Leistungsmessungen ist zu berücksichtigen, dass die Testläufe auf Notebooks aktueller Leistungsklasse durchgeführt wurden. Echte Serversysteme bieten erheblich bessere CPU- und IO-Leistungen; eine quantitative Extrapolation der Messergebnisse ist allerdings schwierig.

Dennoch lässt sich zusammenfassen, dass basierend auf den Lastabschätzungen für die Justiz aus dem IT-Feinkonzept zu S.A.F.E. die selbst auf der bescheidenen Hardware erzielten Werte zu Durchsatz und Antwortzeit realistischerweise erreichbar sind.

## 5 Anhang

### 5.1 WSDL der Attribute-Service-Implementierung (SPML-Query-Provider)

```

<definitions name="SPMLQueryService" targetNamespace="http://www.dataport.de/wsd/poc/spml/query/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">
 <xs:import namespace="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/"

 <wsp:Policy wsu:Id="SamlTokenPolicy"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
 xmlns:wsp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
 <wsp:ExactlyOne>
 <wsp>All>
 <sp:SymmetricBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
 <wsp:Policy>
 <sp:ProtectionToken>
 <wsp:Policy>
 <sp:IssuedToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
 <Issuer xmlns="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
 <Address xmlns="http://www.w3.org/2005/08/addressing">
 http://10.1.2.190:8080/axis2/services/STS</Address>
 </Issuer>
 <sp:RequestSecurityTokenTemplate>
 <t:TokenType
xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">urn:oasis:names:tc:SAML:1.0:assertion</t:TokenType>
 <t:KeyType
xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
</t:KeyType>
 <t:KeySize xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">
 256</t:KeySize>
 </sp:RequestSecurityTokenTemplate>
 <wsp:Policy>
 <sp:RequireInternalReference/>
 </wsp:Policy>
 </sp:IssuedToken>
 </wsp:Policy>
 </sp:ProtectionToken>
 <sp:AlgorithmSuite>
 <wsp:Policy>
 <sp:Basic256/>
 </wsp:Policy>
 </sp:AlgorithmSuite>
 <sp:Layout>
 <wsp:Policy>
 <sp:Strict/>
 </wsp:Policy>
 </sp:Layout>
 <sp:IncludeTimestamp/>
 <sp:OnlySignEntireHeadersAndBody/>
 </wsp:Policy>
 </sp:SymmetricBinding>
 <sp:Wss11 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
 <wsp:Policy>
 <sp:MustSupportRefKeyIdentifier/>
 <sp:MustSupportRefIssuerSerial/>
 <sp:MustSupportRefThumbprint/>
 <sp:MustSupportRefEncryptedKey/>
 </wsp:Policy>
 </sp:Wss11>
 <sp:Trust10 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
 <wsp:Policy>
 <sp:MustSupportIssuedTokens/>
 </wsp:Policy>
 </sp:Trust10>
 </wsp>All>
 </wsp:ExactlyOne>
 </wsp:Policy>

```

```

</sp:Trust10>
<sp:SignedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
 <sp:Body/>
 <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
 <sp:Header Name="From" Namespace="http://www.w3.org/2005/08/addressing"/>
 <sp:Header Name="FaultTo" Namespace="http://www.w3.org/2005/08/addressing"/>
 <sp:Header Name="ReplyTo" Namespace="http://www.w3.org/2005/08/addressing"/>
 <sp:Header Name="MessageID" Namespace="http://www.w3.org/2005/08/addressing"/>
 <sp:Header Name="RelatesTo" Namespace="http://www.w3.org/2005/08/addressing"/>
 <sp:Header Name="Action" Namespace="http://www.w3.org/2005/08/addressing"/>
</sp:SignedParts>
<sp:EncryptedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
 <sp:Body/>
</sp:EncryptedParts>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>

<types>
 <xs:schema>
 <xs:import namespace="urn:oasis:names:tc:SPML:2:0"/>
 <xs:import namespace="urn:oasis:names:tc:SPML:2:0:search"/>
 </xs:schema>
</types>

<message name="SPMLIterateResponseMessage">
 <part name="body" element="spmls:iterateResponse"/>
</message>
<message name="SPMLSearchResponseMessage">
 <part name="body" element="spmls:searchResponse"/>
</message>
<message name="SPMLLookupRequestMessage">
 <part name="body" element="spml:lookupRequest"/>
</message>
<message name="SPMLIterateRequestMessage">
 <part name="body" element="spmls:iterateRequest"/>
</message>
<message name="SPMLLookupResponseMessage">
 <part name="body" element="spml:lookupResponse"/>
</message>
<message name="SPMLListTargetsResponseMessage">
 <part name="body" element="spml:listTargetsResponse"/>
</message>
<message name="SPMLSearchRequestMessage">
 <part name="body" element="spmls:searchRequest"/>
</message>
<message name="SPMLListTargetsRequestMessage">
 <part name="body" element="spml:listTargetsRequest"/>
</message>

<portType name="SPMLQueryPortType">
 <operation name="SPMLListTargetsRequest">
 <input name="SPMLListTargetsRequestInput" message="tns:SPMLListTargetsRequestMessage"/>
 <output name="SPMLListTargetsRequestOutput" message="tns:SPMLListTargetsResponseMessage"/>
 </operation>
 <operation name="SPMLLookupRequest">
 <input name="SPMLLookupRequestInput" message="tns:SPMLLookupRequestMessage"/>
 <output name="SPMLLookupRequestOutput" message="tns:SPMLLookupResponseMessage"/>
 </operation>
 <operation name="SPMLSearchRequest">
 <input name="SPMLSearchRequestInput" message="tns:SPMLSearchRequestMessage"/>
 <output name="SPMLSearchRequestOutput" message="tns:SPMLSearchResponseMessage"/>
 </operation>
 <operation name="SPMLIterateRequest">
 <input name="SPMLIterateRequestInput" message="tns:SPMLIterateRequestMessage"/>
 <output name="SPMLIterateRequestOutput" message="tns:SPMLIterateResponseMessage"/>
 </operation>
</portType>

```

```

<binding name="SPMLSoapBinding" type="tns:SPMLQueryPortType">
 <wsp:PolicyReference URI="#SamlTokenPolicy" wsdl:required="true"/>
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="SPMLListTargetsRequest">
 <soap:operation soapAction="urn:oasis:names:tc:SPML:2:0:req/listTargetsRequest" style="document"/>
 <input name="SPMLListTargetsRequestInput">
 <soap:body use="literal"/>
 </input>
 <output name="SPMLListTargetsRequestOutput">
 <soap:body use="literal"/>
 </output>
 </operation>
 <operation name="SPMLLookupRequest">
 <soap:operation soapAction="urn:oasis:names:tc:SPML:2:0:req/lookupRequest" style="document"/>
 <input name="SPMLLookupRequestInput">
 <soap:body use="literal"/>
 </input>
 <output name="SPMLLookupRequestOutput">
 <soap:body use="literal"/>
 </output>
 </operation>
 <operation name="SPMLSearchRequest">
 <soap:operation soapAction="urn:oasis:names:tc:SPML:2:0:req/searchRequest" style="document"/>
 <input name="SPMLSearchRequestInput">
 <soap:body use="literal"/>
 </input>
 <output name="SPMLSearchRequestOutput">
 <soap:body use="literal"/>
 </output>
 </operation>
 <operation name="SPMLIterateRequest">
 <soap:operation soapAction="urn:oasis:names:tc:SPML:2:0:req/iterateRequest" style="document"/>
 <input name="SPMLIterateRequestInput">
 <soap:body use="literal"/>
 </input>
 <output name="SPMLIterateRequestOutput">
 <soap:body use="literal"/>
 </output>
 </operation>
</binding>

<service name="SPMLQueryService">
 <port name="SPMLQueryServicePort" binding="tns:SPMLSoapBinding">
 <soap:address location="http://10.1.2.190:8080/axis2/services/SPMLQueryService"/>
 </port>
</service>
</definitions>

```

## 5.2 Abbildungsverzeichnis

|                                                                                |    |
|--------------------------------------------------------------------------------|----|
| Abbildung 1: Konfigurationsdatei zur Festlegung des Lastprofils.....           | 10 |
| Abbildung 2: Messung von 1024 Anfragen mit einem Client.....                   | 15 |
| Abbildung 3: Messung mit 32 Client-Threads im Burst-Modus .....                | 16 |
| Abbildung 4: Messung von 20 Client-Threads mit je 36 Request/min.....          | 17 |
| Abbildung 5: Logarithmische Darstellung der <i>search</i> -Antwortzeiten ..... | 18 |
| Abbildung 6: Größe von <i>search</i> -Antwortnachrichten.....                  | 19 |

## 5.3 Tabellenverzeichnis

|                                                                                          |    |
|------------------------------------------------------------------------------------------|----|
| Tabelle 1: Antwortzeiten bei einem Durchsatz von 6 Request/s .....                       | 17 |
| Tabelle 2: Gemittelte <i>search</i> -Antwortzeiten gemessen bei einem Client-Thread..... | 18 |

## 5.4 Referenzierte Dokumente

[1] S.A.F.E.-Feinkonzept - Dokument 1: System- und Schnittstellenspezifikation Föderiertes Identity Management

[2] S.A.F.E.-Feinkonzept - Dokument 2: IT-Feinkonzept.