Überprüfung der Gültigkeit von IF-MAP-Graphen

Bachelorarbeit im Studiengang Angewandte Informatik

Fakultät IV - Wirtschaft und Informatik

Hochschule Hannover
University of Applied Sciences and Arts

Anton Saenko
anton.saenko@gmx.de

14. August 2013

Beteiligte

Erstprüfer

Prof. Dr. rer. nat. Josef von Helden
Ricklinger Stadtweg 120
30459 Hannover
E-Mail: josef.vonhelden@hs-hannover.de

Zweitprüfer

Bastian Hellmann M.Sc.
Ricklinger Stadtweg 120
30459 Hannover
E-Mail: bastian.hellmann@hs-hannover.de

Author

Anton Saenko
Otternhagener Str. 133
31535 Neustadt
E-Mail: anton.saenko@gmx.de

Selbstständigkeitserklärung

Hiermit erkläre ich an Eides Statt, dass ich die eingereichte Bachelorarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort, Datum Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1. Einleitung

1.1. Motivation

In der Bachelorarbeit beschäftige ich mich mit dem TCG IF-MAP Protokoll, welches entwickelt wurde, um zwischen beliebigen Netzwerkkomponenten im einem Netzwerk sicherheitsrelevante Daten austauschen zu können. Das IF-MAP Protokoll wird durch Spezifikationen definiert. Diese beschreiben wie die Daten (**Metadaten**) aussehen können, welche Netzwerkkomponente **Identifier** mit welchen Metadaten zusammen veröffentlicht werden können, welche Attribute die Metadaten enthalten können, welche Funktionen für die Kommunikation zur Verfügung stehen (**publish**, **search**, **subscribe**) und wie die IF-MAP Komponenten miteinander kommunizieren sollen. Um es Jedem zu ermöglichen das IF-MAP Protokoll zu implementieren, werden die Metadaten und die IF-MAP Kommunikation mit Hilfe von XML-Schema genau definiert und anhand von zahlreichen Beispielen erklärt. Derzeit besteht das Problem, dass viele IF-MAP Komponenten die Möglichkeit bieten, nicht spezifikationskonforme Metadaten zu veröffentlichen.

1.2. Ziel der Arbeit

Ziel der Arbeit ist es, nach dem die relevanten IF-MAP Spezifikationen analysiert wurden, ein Konzept auszuarbeiten, damit veröffentlichte Metadaten auf Spezifikationskonformität überprüfen werden können. Dabei sollen Realisierungsmöglichkeiten der verschiedenen Prüfverfahren betrachtet werden, hinsichtlich des Einsatzes als IF-MAP Server oder Client. Hierzu soll betrachtet werden, ob dies im IF-MAP Server oder als IF-MAP Client realisierbar ist. Am Schluss soll eine exemplarische Implementierung des Konzepts umgesetzt und evaluiert werden.

2. Grundlagen

Zum besseren Verständnis werden in diesem Kapitel die grundlegende Technologien und Softwarekomponenten kurz vorgestellt, auf denen diese Arbeit basiert.

2.1. Verwendete Technologien

2.1.1. XML / XML-Schema

Die IF-MAP Nachrichten werden in Form von **XML**-Dokumenten über das Netzwerk ausgetauscht. In der IF-MAP Spezifikation [2] werden die IF-MAP Nachrichten mit Hilfe von **XML-Schema** definiert.

XML (Extensible Markup Language) ist ein W3C-Standard zur Dokumentenauszeichnung. XML ermöglicht den plattform- und implementationsunabhängigen Austausch von hierarchisch strukturierter Daten zwischen verschiedenen Komponenten in einem Netzwerk. Mehr Informationen zu XML finden sich auf der W3C Webseite [8].

XML-Schema ist eine Sprache zur Definition XML-basierter Sprachen. XML-Schema sind wohlgeformte und gültige XML-Dokumente, in denen beschrieben wird, wie die XML-Dokumente auszusehen haben und welche Elemente das Dokument enthalten kann. Mehr Information zu XML-Schema finden sich auf der W3C Webseite [9].

Listing 1: XML-Schema Beschreibung der IF-MAP Nachricht (publish delete)

```
1  <xsd:complexType name="DeleteType">
2   <xsd:sequence>
3     <xsd:choice minOccurs="1" maxOccurs="2">
4       <xsd:element name="access-request"
5         type="AccessRequestType"/>
6       <xsd:element name="identity" type="IdentityType"/>
7       ...
8     </xsd:choice>
9   </xsd:sequence>
10  <xsd:attribute name="filter" type="FilterType" use="
      optional"/>
11 </xsd:complexType>
```

Wie an dem Beispielcode Listing 1 zu sehen ist, wird mit Hilfe des XML-Schema ein Element definiert, das mindestens einen und maximal zwei Unterelemente enthalten kann, die in den Zeilen 4 und folgende definiert sind. Ebenso kann das definiertes Element einen Attribut „filter“ enthalten deren Wert ein „FilterTyp“ sein muss.

2.1.2. SOAP

Mit Hilfe von SOAP werden die IF-MAP Nachrichten zwischen den IF-MAP Komponenten ausgetauscht. SOAP (Simple Object Access Protocol) ist ein leichtgewichtiges, programmiersprachenunabhängiges Protokoll, was durch W3C standardisiert wurde. Das Protokoll dient zum Austausch von XML-Nachrichten über das Netzwerk. Jede SOAP-Nachricht besteht aus drei Teilen: dem Umschlag (Envelope), dem Kopf (Header) und dem Datenkörper (Body), der die eigentliche Daten enthält. Während der Header optional ist, müssen die Elemente Envelope und Body in jeder SOAP-Nachricht vorhanden sein. Der Aufbau der SOAP-Nachricht sieht wie folgt aus:

- **SOAP Envelope** - Besteht aus zwei Teilen: Dem SOAP Header und dem SOAP Body.
- **SOAP Header** - Enthält die infrastrukturellen Daten wie zum Beispiel: Session-Number, Transaction-Number oder die Signatur (XML-Security).
- **SOAP Body** - Der Body enthält die eigentlichen Informationen (Anwendungsdaten).

Ausgetauscht können SOAP-Nachrichten über die verschiedene Transport-Protokolle wie zum Beispiel HTTP, SMTP, JMS werben. Mehr Information zu SOAP auf der W3C Webseite [7].

Listing 2: Eine SOAP-Nachricht

```
1  <?xml version="1.0"?>
2  <env:Envelope
3    xmlns:env="http://www.w3.org/2003/05/soap-envelope"
4    xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/
       IFMAP/2"
5    xmlns:meta="http://www.trustedcomputinggroup.org/2010/IFMAP
       -
6  METADATA/2">
7    <env:Body>
8      <ifmap:publish session-id="128738734">
9  ...
10   </ifmap:publish>
11   </env:Body>
12 </env:Envelope>
```

In dem Codebeispiel Listing 2 ist die Struktur der SOAP-Nachrichten gut zusehen. Zuerst ist das Envelope Element deklariert, der das Body Element enthält. In dem Body befindet sich die erwartete Information.

2.2. Trusted Computing Group

Die Trusted Computing Group (TCG) ist ein „non-profit“ Konsortium von Firmen, welche das Ziel hat, offene Spezifikationen für vertrauenswürdige Rechensysteme zu entwickeln, um die Sicherheit verteilter Anwendungen zu erhöhen. Aktuell sind in der TCG über 170 Hersteller aus aller Wellt vertreten. Darunter befinden sich große Hersteller wie Intel, AMD, IBM, Microsoft, HP, Juniper und Andere.

Trusted Computing (TC) beschreibt eine hardwarebasierte Lösung, die ein konsistentes Verhalten nicht nur von Computern, sondern auch von anderen digitalen Geräten, wie zum Beispiel Smartphons, sicherstellen kann. Das Kernelement dieser Lösung ist ein Hardware-Chip „Trusted Platform Module“ (TPM), der mittels kryptographischer Verfahren die Integrität der Soft- und Hardwarekonfiguration ermitteln kann und die werte nachprüfbar abspeichert.

Trusted Network Connect (TNC) ist ein Konzept für Netzwerksicherheit, der von der Trusting Computing Group standardisiert wurde. Mit TNC sollen Netzwerkadministratoren die Integrität und Konformität der Endgeräte und deren Verbindung in Bezug auf die vereinbarten Richtlinien sicherstellen. Ein Client, der einen TNC-Handshake erfolgreich ausgeführt hat, wird als vertrauenswürdig angesehen. Wie auf der Abbildung 1 zu sehen ist, besteht das TNC aus mehreren Komponenten: **AR**, **PDP**, **PEP**, **MAP** und **MAPC**.

- **AR** (Access Requestor) - Ist ein Gerät, welches auf ein Netzwerk zugreifen möchte.
- **PDP** (Policy Decision Point) - Nachdem **AR** sich gegenüber dem **PDP** authentifiziert hat, entscheidet der **PDP**, ob ein **AR** Zugang zu einem Netzwerk erlaubt wird oder nicht.
- **PEP** (Policy Enforcement Point) - In dem Verbindungsaufbau des **AR** zu dem TNC spielt das **PEP** eine aktive Rolle, indem es anhand der Entscheidung des **PDP**, die Verbindung zu dem Netz regelt.
- **MAP** (Metadata Access Point) - Ist ein MAP-Server, der die Metadaten speichert und verwaltet.
- **MAPC** - (Metadata Access Point Client) - Ein MAP Client kann auf dem **MAP** Server Metadaten veröffentlichen, kann nach Metadaten suchen, löschen und mit Hilfe von Abonnements über Änderungen an Metadaten benachrichtigt werden.

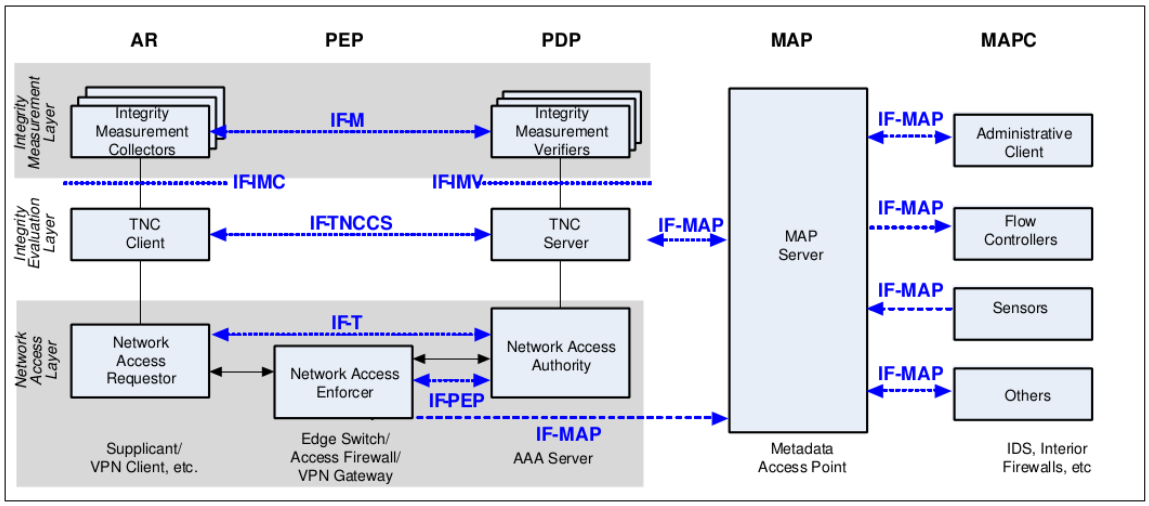


Abbildung 1: Übersicht über die TNC Struktur. Quelle: IF-MAP Spezifikation [2]

Mehr Information zu Trusted Computing Group unter [4]

2.3. IF-MAP

IF-MAP (Interface for Metadata Access Points) ist ein standardisiertes, Client/Server Protokoll zum Austausch von Metadaten. Mit dem Protokoll sollen sicherheitsrelevante Daten zwischen Komponenten in einem Netzwerk ausgetauscht werden können. Wie bereits im Abschnitt 2.1.2 beschrieben, werden die Nachrichten zwischen den IF-MAP Komponenten mit Hilfe von SOAP übermittelt. Wie auf der Abbildung 2 zu sehen ist, melden sich alle IF-MAP Clients auf dem IF-MAP Server an, um Daten austauschen zu können. Dadurch wird erreicht, dass die Kommunikation zwischen den einzelnen IF-MAP Clients ausschließlich über den IF-MAP Server stattfindet. Ebenso ist auf der Abbildung zu sehen, welche Komponenten es typischerweise geben kann.

Das IF-MAP Protokoll ist in der „TNC IF-MAP Binding for SOAP“ [2] Spezifikation durch TCG festgelegt. In der genannten Spezifikation werden sowohl das Datenalsauch das Kommunikationsmodell definiert. Von der „Trusted Computing Group“ wurden weitere Spezifikationen herausgeben, die für die Beschreibung der Metadaten zuständig sind. In der Spezifikation „TNC IF-MAP Metadata for Network Security“ [1] werden die Metadaten für die Netzwerksicherheit vorgestellt und in der „TNC IF-MAP Metadata for ICS Security“ [3] die Metadaten für den industriellen Einsatz.

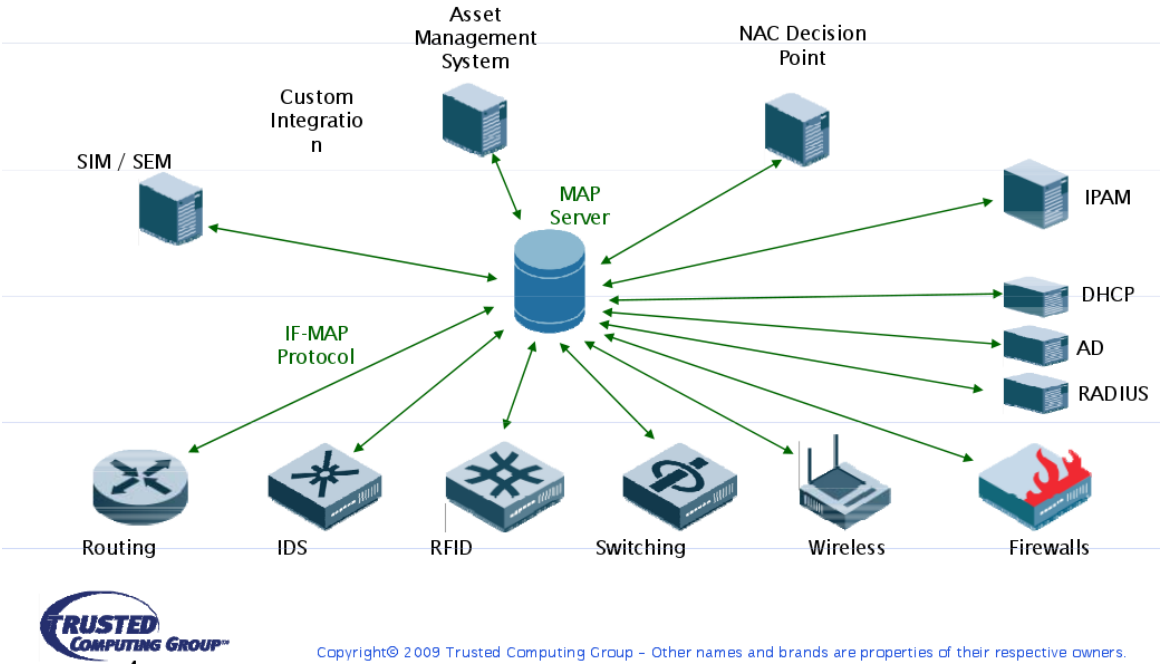


Abbildung 2: Kommunikation der IF-MAP Komponenten.
Quelle: IF-MAP Spezifikation [2]

Für die Kommunikation mit dem IF-MAP Server stehen dem Client folgende Operationen zu Verfügung: **publish**, **search** und **subscribe**.

- **publish** - Mit dieser Operation kann ein IF-MAP Client mehrere Aufgaben erledigen.
  - Mit dem **publish update** werden neue Metadaten auf dem IF-MAP Server veröffentlicht oder, wenn die Metadaten schon vorhanden sind, aktualisiert.
  - Mit dem **publish notify** werden die Metadaten ebenfalls veröffentlicht, jedoch werden diese nicht in die eigentliche Datenstruktur aufgenommen, sondern nur von den IF-MAP Clients, die ein Abonnement auf bestimmte Metadaten haben, empfangen.
  - Mit **publish delete** können Metadaten auf dem Server gelöscht werden
- **search** - Mit der Operation können IF-MAP Clients auf dem IF-MAP Server nach bestimmten Metadaten suchen. Die Suche kann mit Hilfe einer Filterfunktion, durch Eingabe zusätzlicher Parameter, verfeinert werden.
- **subscribe** - Mt subscribe können IF-MAP Clients Informationen auf bestimmte Metadaten abonnieren und soweit die Metadaten von dem Publisher verändert

werden, informiert sie der Server über die Veränderungen. Auch bei subscribe steht eine Filterfunktion zur Verfügung.

Für die Kommunikation zwischen einem IF-MAP Server und einem IF-MAP Client stehen zwei verschiedene Kommunikationskanäle zur Verfügung: **SSRC** und **ARC**.

Synchronous Send-Receive Channel (SSRC)
Ist ein synchroner Sende- und Empfangskanal, auf dem die Verbindung zwischen dem Client und dem Server aufgebaut wird und die synchronen Operationen **publish** und **search** ausgeführt werden. Bei der Operation **subscribe** wird die Anfrage (**subscribeRequest**) über SSRC an den Server gesendet und parallel zu SSRC eine ARC Verbindung aufgebaut, über die der Server die Antwort (**pollResult**) zurücksendet.

Asynchronous Receive Channel (ARC)
Wie oben erläutert, wird mit der Operation **subscribe** ein Client über Veränderungen der abonnierten Metadaten informiert. Damit der Client in der Zeit andere Aufgaben erledigen kann, ist eine asynchrone Verbindung notwendig. **ARC** ist eine asynchrone Verbindung über die der Client mit Hilfe der *Poll-Methode* über Veränderungen auf dem Server informiert wird.

In der Abbildung 3 wird die Kommunikation zwischen dem IF-MAP Client und dem IF-MAP Server bei der **subscribe** Operation verdeutlicht. Nach dem der Client, über den SSRC Kanal, einen (**subscribeRequest**) gesendet hat und dieser von dem Server bestätigt wurde, baut der Client parallel zu der SSRC Verbindung eine ARC Verbindung auf und ruft die *Poll-Methode* auf, worauf er unmittelbar eine Antwort in Form von **pollResult** bekommt. Danach wartet der Client auf dem ARC Kanal auf weitere Anworten von dem Server. Bei der Zeitmarkierung **t2**, nachdem die abonnierten Metadaten auf dem Server verändert wurden, informiert der Server den Client über die Veränderung. Bei der Zeitmarkierung **t3** wiederholt der Client die **subscribe** Operation und der Ablauf auf dem ARC Kanal wiederholt sich so lange, bis der Client das Abonnement zurückgezogen hat.

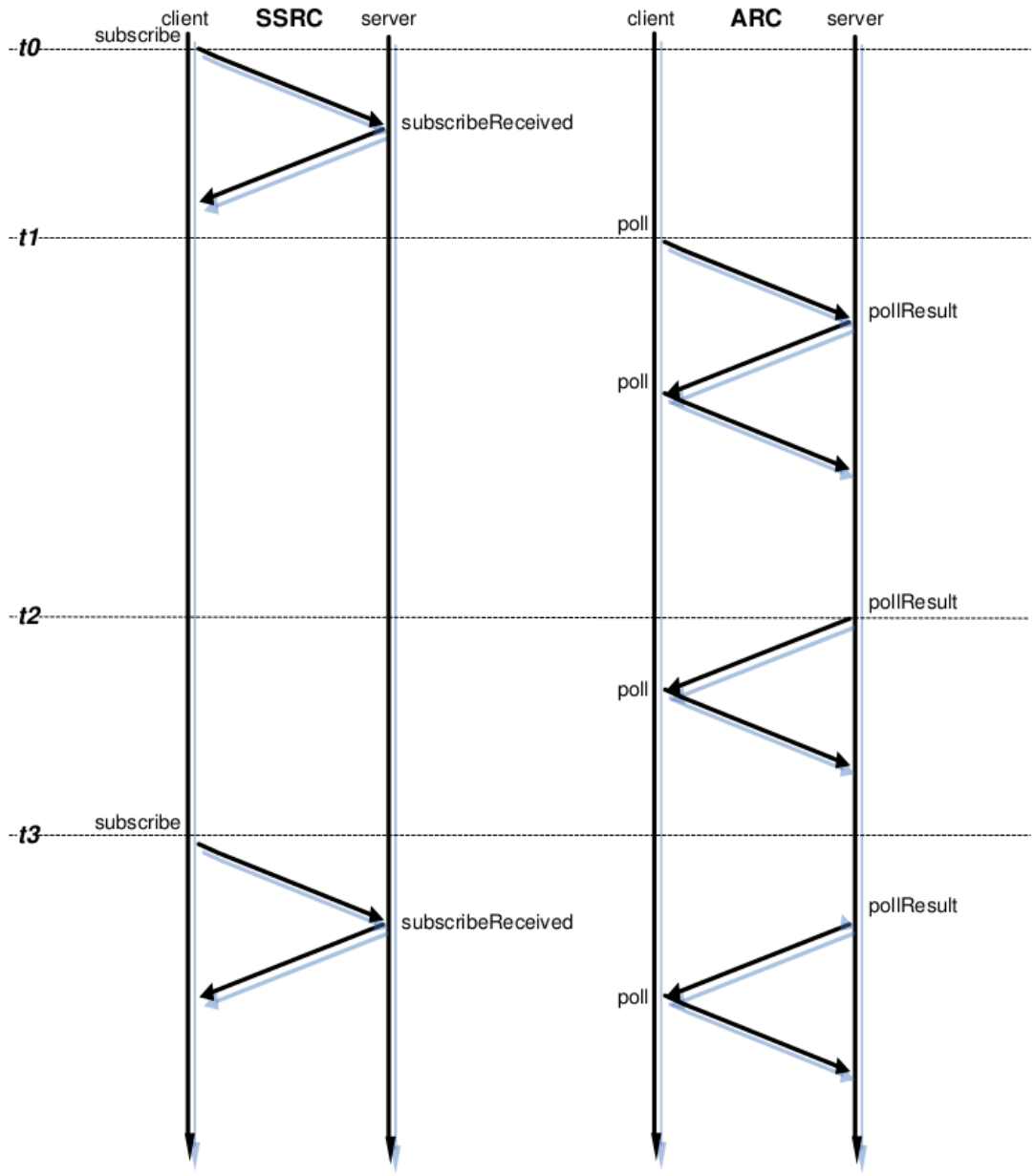


Abbildung 3: Zeitlicher Ablauf der „subscribe Operation“.
Quelle: IF-MAP Spezifikation [2]

2.4. Aufbau des IF-MAP Graphen

Die Informationen, die von einem IF-MAP Client veröffentlicht werden, werden auf dem IF-MAP Server als ungerichteter Graph abgelegt. Wie auf dem Bild 4 zu sehen ist, werden die Metadata und die Identifier als Knoten und die Links als Kanten

abgebildet.

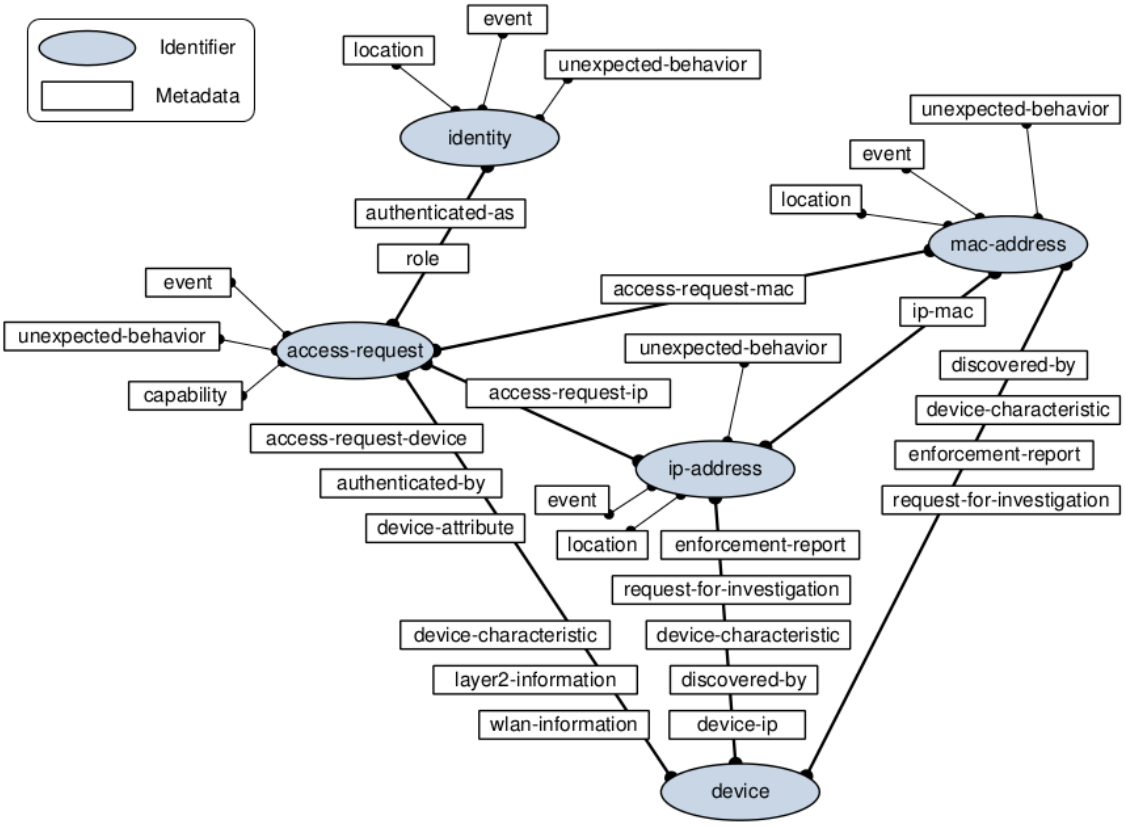


Abbildung 4: TNC IF-MAP Metadata for network security, *Specification Version 1.1*. Quelle: IF-MAP Spezifikation [1]

2.4.1. Identifier

Identifier sind eindeutige Datentypen über die die Netzwerkkomponenten aus der realen Welt auf der IF-MAP Struktur abgebildet werden. In den IF-MAP Spezifikationen werden die fünf Standard-Identifier-Typen genauso wie der **Extended Identifier** definiert. Mit Hilfe von **Extended Identifier** können neue herstellerspezifische Identifier über eigene XML-Schema definiert werden. Im Folgenden werden die Standard-Identifier-Typen vorgestellt.

1. **access-request** - Stellt den Versuch eines Verbindungsaufbaus einer Netzwerkkomponente dar.

2. **identity** - Repräsentiert eine Identität in dem Netzwerk, beispielsweise Benutzer, Applikation oder Gerät.

3. **ip-address** - Repräsentiert eine gültige IPv4 oder IPv6 Adresse.

4. **mac-address** - Repräsentiert eine gültige MAC-Adresse des jeweiligen Gerätes.

5. **device** - Repräsentiert eine physikalische oder virtuelle Identität, die versucht eine Verbindung zu dem Netzwerk aufzubauen. Die Authentifizierungselemente des TNC werden auch als **device** bezeichnet.

2.4.2. Metadata

Metadata (Metadaten) beschreiben den Zustand der Identifier und beinhalten weitere strukturell-beschreibende Daten. In der Spezifikation „TNC IF-MAP Metadata for Network Security“ [1] werden mit Hilfe von XML-Schema die Standard-Metadaten-Typen für den Bereich der Netzwerksicherheit definiert, sowie ein Metadaten-Typ „Vendor-specific Metadata“, der, es ermöglicht eigene Metadaten zu definieren. Jeder Metadatensatz kann durch seine Attribute genauer beschrieben werden. Es gibt optionale Attribute, die ein Metadatensatz enthalten können, außerdem gibt es erforderliche Attribute. Zu den erforderlichen Attributen, die jeder Metadatensatz enthalten muss, zählen unter anderem: *ifmap-publisher-id*, *ifmap-timestamp*, *ifmap-cardinality* und *lifetime*. Mehr Information zu den einzelnen Attributen sind im Abschnitt 4.1 zu entnehmen.

Laut der Spezifikation „TNC IF-MAP Metadata for Network Security“ [1] unterstützt das aktuelle IF-MAP 2.1 Protokoll zwanzig verschiedene Metadaten-Typen inklusive des „Vendor-specific Metadata“. Hier ein Ausschnitt aus der Spezifikation, in dem ein Metadaten-Typ *device-ip* definiert wird.

Listing 3: XML-Schema, Definition des Metadaten-Typs *device-ip*.

```
<xsd:element name="device-ip">
 <xsd:complexType>
   <xsd:attributeGroup
      ref="ifmap:singleValueMetadataAttributes"/>
 </xsd:complexType>
</xsd:element>
```

2.4.3. Links

Zwei Identifier können über Links aneinander gebunden werden, wobei der Link einen oder mehrere unterschiedliche Metadaten enthalten kann.

2.4.4. Gültigkeit des Graphen

Laut den Spezifikationen ist ein IF-MAP Graph gültig, wenn alle enthalten Metadaten mit dem dazugehörigen Identifier verknüpft sind und es keine Links gibt, die keine Metadaten enthalten.

2.5. Trust@FHH Iron* Softwarekomponenten

In diesem Abschnitt werden einige Softwarekomponenten der Trust@FHH Forschungsgruppe 2.5.1 vorgestellt und genauer beschrieben. iron* ist eine Sammlung von Open-Source Softwarekomponenten, die den TCG IF-MAP-Standard implementieren.

Intelligent Reaction on Network Events (IRON) war ein Bachelorprojekt indem vierzehn Studenten der Fachhochschule Hannover sich mit dem IF-MAP Protokoll beschäftigt haben. Als Ergebnis des Projekts, das im Juli 2010 abgeschlossen wurde, entstand der IF-MAP Server Irond. Seitdem wurden weitere IF-MAP Softwarekomponenten unter dem iron-Namensschema veröffentlicht.
Die Softwarekomponenten, die unter Open-Source-Lizenz veröffentlicht werden, können von der Webseite der Trust@FHH Forschungsgruppe [6] bezogen werden.

2.5.1. Trust@FHH

Trust@FHH ist eine Forschungsgruppe der Hochschule Hannover, die von Herrn Prof. Dr. Josef von Helden geleitet wird. Die Trust@FHH forscht auf dem Gebiet der IT-Sicherheit. Die Schwerpunkte der Forschung liegen in den Bereichen Trusted Computing (TC) und Trusted Network Connect (TNC). Trust@FHH ist ein Liasion Member des TCG Verbandes und dort aktiv an der Festlegung der verschiedenen TCG Spezifikation beteiligt.

2.5.2. Ifmapj

Ifmapj (aktuelle Version 0.1.5) [5] ist eine Java Klassenbibliothek, die das IF-MAP 2.1 Protokoll implementiert. Ifmapj wurde entwickelt, um den Programmierer/in beim Entwickeln von IF-MAP Clients zu entlasten. Mit Ifmapj kann der Programmierer/in Java Software erstellen, in dem er/sie die von Ifmapj zu Verfügung gestellte Schnittstelle verwendet. Aufgaben wie: Das Generieren und Versenden von SOAP-Nachrichten, das Empfangen und Interpretieren der SOAP-Nachrichten, sowie der Aufbau und die Verwaltung der Verbindungen werden von Ifmapj für den Programmierer/in übernommen.

Wie am Anfang beschrieben, unterstützt die aktuelle Ifmapj Version 0.1.5 das IF-MAP 2.1 Protokoll, was bedeutet, dass mit Hilfe von Ifmapj eigene Identifier „Exten-

ded Identifier“, sowie eigene Metadaten „Vendor-specific Metadata“ erstellt werden können.

2.5.3. Irond

Irond (aktuelle Version 0.4.0)[5] ist ein IF-MAP Server, der in Java implementiert wurde. In der aktuellen Version implementiert Irond das IF-MAP 2.1 Protokoll.

Derzeit werden die empfangenen Daten der Clients auf dem irond nicht persistent gespeichert, sondern werden in dem Arbeitsspeicher gehalten, bis der Server beendet wird. Bei einem Ausfall bedeutet dies, dass alle auf dem Server veröffentlichten Daten verlorengehen gehen.

Besonderheit von Irond
Zusätzlich zu den Operationen, die in den Spezifikationen definiert sind, besitzt irond eine „**Dump**“ Operation. Mit Hilfe von „**Dump**“ können die IF-MAP Clients alle Informationen, die sich auf dem Server befinden, empfangen ohne den Startidentifier eingeben zu müssen.

2.5.4. Ifmapcli

Ifmapcli (aktuelle Version 0.0.3)[5] ist eine Sammlung von IF-MAP Command Line Interface Tools. Mit Hilfe der Tools können über die Konsole oder mit einem Skript beliebige Metadaten auf einem IF-MAP Server veröffentlicht (**publish**) und gelöscht (**delete**) werden, nach Metadaten auf dem Server gesucht (**search**) und Information über bestimmte Metadaten abonniert (**subscribe**) werden.

2.5.5. Irongui

Irongui (aktuelle Version 0.4.1)[5] ist ein IF-MAP Client, der auf einem IF-MAP Server veröffentlichte Daten, visuell darstellen kann.
Das Softwaretool wurde in der Bachelorarbeit „Visualisierung von Informationen einer zentralen Netzwerk-Datenbank“ von Tobias Ruhe entwickelt.

Aufbau von Irongui (Abbildung 5)

1. Hier werden alle Verbindungen aufgelistet über die Irongui mit einem IF-MAP Server kommunizieren kann.

2. In diesem Fenster werden die veröffentlichten Daten als ungerichteter Graph visuell dargestellt.

3. Hier können Attribute mitsamt ihren Werten zu jedem Metadatensatz angezeigt werden.

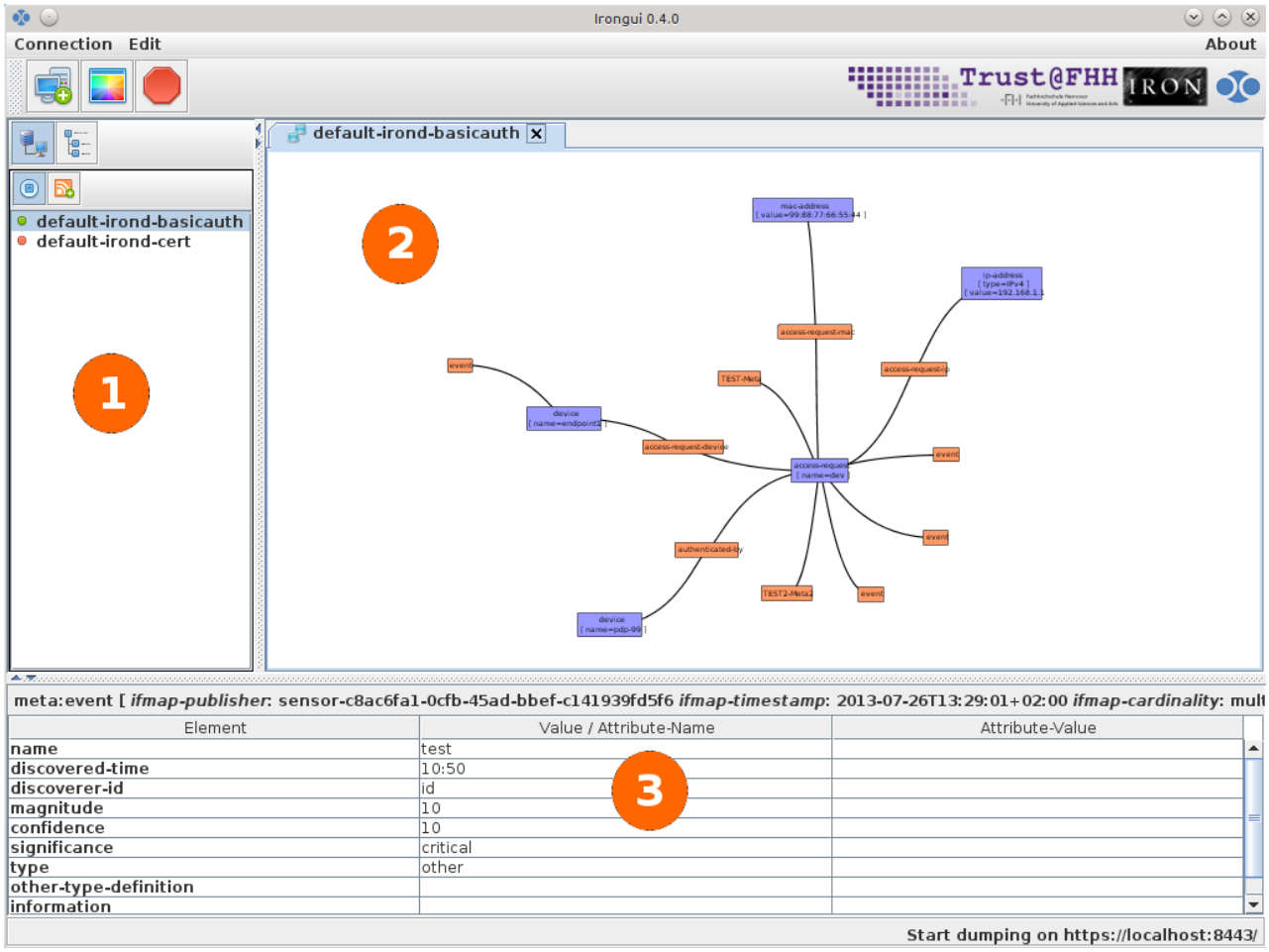


Abbildung 5: IF-MAP Client Irongui. Version 0.4.1

2.5.6. Ironcontrol

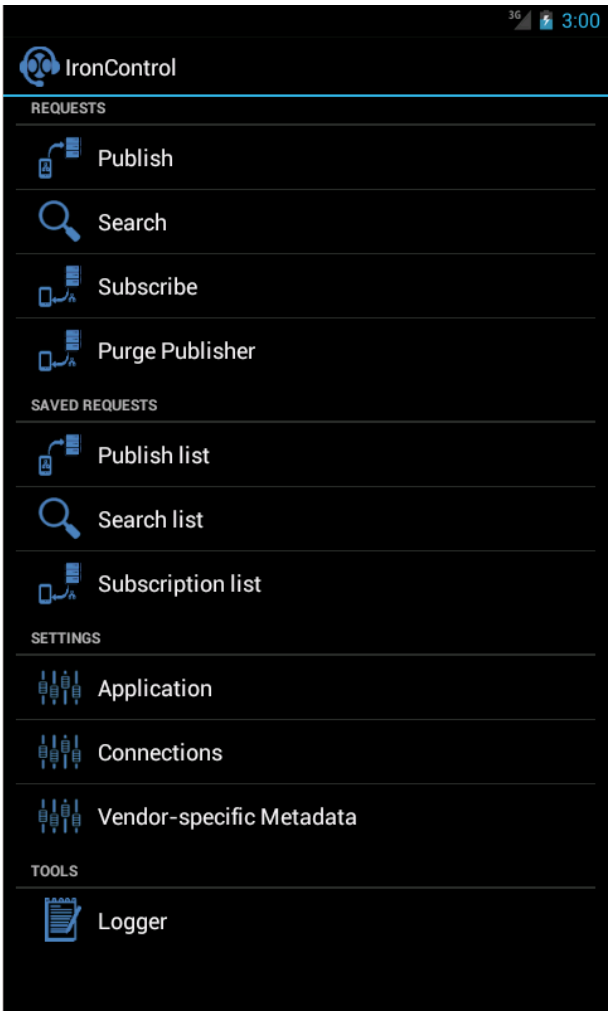


Abbildung 6: Android App ironcontrol

Ironcontrol ist ein IF-MAP Client, der für mobile Android Geräte entwickelt wurde. Ironcontrol ist ein Tool zur Administration einer IF-MAP-Umgebung. In der aktuellen "Beta" Version, kann das Tool die Standard IF-MAP Operationen wie: **publish**, **search** und **subscribe** ausführen und mit denn dazugehörigen Ergebnissen persistent speichern. Bei Verwendung der **subscribe** Operation, kann der Benutzer über die Veränderungen der Daten auf seinem Smartphone durch die Android-eigenen Funktionen wie Vibrationen und Audiobenachrichtigungen, informiert werden.

3. Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an das Prüfverfahren des IF-MAP Graphen analysiert und beschrieben.

3.1. Funktionale Anforderungen

3.1.1. Anforderung A1: Prüfung der Konformität an den Vorgaben der Spezifikation

- Die wichtigste Anforderung bei der Gültigkeitsüberprüfung ist die Feststellung, inwiefern die Metadaten die Vorgaben der Spezifikation einhalten. Dazu müssen die Prüfregeln der Gültigkeitsüberprüfung den Vorgaben der Trusted Computing Group Spezifikationen entsprechen.

3.1.2. Anforderung A2: Erweiterbarkeit des Prüfverfahrens

- Als Vorbereitung für mögliche Weiterentwicklungen des IF-MAP Protokolls ist es wichtig, dass eine Möglichkeit besteht, die Gültigkeitsüberprüfung zu erweitern, damit auch die Vorgaben aus nachfolgenden Spezifikationen mit der Lösung abgedeckt werden können.

- Die Prüfregeln müssen einfach und übersichtlich strukturiert sein um, auch bei Änderungen der Spezifikation, leicht angepasst werden zu können.

3.1.3. Anforderung A3: Die Schnittstelle

- Die Schnittstelle soll zukunftssicher und einfach gestaltet werden. Der Benutzer muss ohne großen Aufwand und ohne viele Eingaben ein IF-MAP Graph auf Spezifikationskonformität überprüfen können. Deshalb ist schon bei der Planung der Kommunikationsschnittstelle zu dem Prüfverfahren darauf zu achten, dass diese so einfach wie möglich konstruiert wird. Dazu ist die Anzahl der Übergabeparameter zu beschränken.

3.1.4. Anforderung A4: Das Prüfergebnis

- Damit das Ergebnis der Gültigkeitsüberprüfung weiter analysiert werden kann sind genug Informationen bereitzuhalten. Das Ergebnis muss mindestens folgende Komponente enthalten: Einen auswertbaren Wert, ob der überprüfte IF-MAP Graph spezifikationskonform ist oder nicht. Ebenso so wichtig ist es, dass bei einem negativen Ergebnis der Überprüfung, wenn der IF-MAP Graph nicht der Spezifikation entspricht, die Information enthalten ist, welche Komponenten in dem Graph betroffen sind und welche Regeln verletzt wurden.

3.2. Nichtfunktionale Anforderungen

3.2.1. Anforderung A5: Plattformunabhängigkeit

- Die Überprüfung des IF-MAP Graphen soll nach Möglichkeit unabhängig von dem Betriebssystem durchführbar sein, um möglichst hohe Flexibilität zu gewähren.

3.2.2. Anforderung A6: Performance

- Um große Mengen an Metadaten überprüfen zu können, muss das Prüfverfahren effizient sein.
- Um auch auf mobilen Endgeräten betrieben werden zu können, sollte die Überprüfung Ressourcenschonend ablaufen.

3.3. Klassifizierung der Anforderungen

In der nachfolgenden Tabelle 7 werden die oben genannten Anforderungen klassifiziert.

Anforderungen	**Priorität**		
A1	Hoch		
A2	Hoch		
A3	Hoch		
A4	Hoch		
A5		Mittel	
A6			Niedrig

Abbildung 7: Prioritäten der Anforderungen an das Prüfverfahren.

Bei der Klassifizierung der Anforderung wurde großer Wert darauf gelegt, dass die Gültigkeitsüberprüfung zuverlässig, zukunftssicher und benutzerfreundlich sein soll. Die Anforderung **A6 Performance** wurde mit niedrig eingestuft, da mit den aktuellen Rechnern und Netzwerksystemen die Performance der Software keine große Rolle mehr spielt.

4. Analyse

In dem ersten Teil des Kapitels werden die für die Umsetzung des Prüfverfahren bzw. für die Konzipierung der Prüfregeln relevanten IF-MAP Spezifikationen analysiert. Hierbei werden die beiden betreffenden Spezifikationen der Trusted Computing Group [1][2] untersucht. Desweiteren werden die Softwarekomponenten der Trust@FHH iron* betrachtet und im Zusammenhang mit der Umsetzung des Prüfverfahrens analysiert.

4.1. Analyse der relevanten IF-MAP Spezifikationen

In den beiden öffentlich zugänglichen IF-MAP Spezifikationen „TNC IF-MAP Metadata for Network Security“ [1] und „TNC IF-MAP Binding for SOAP“ [2] definiert die Trusted Computing Group [4] ein Modell zum Austausch von Metadaten in einem Netzwerk. Informationen dazu finden sich in dem Abschnitt 2.3.
In der Spezifikation „IF-MAP Metadata for Network Security“ werden die Standard-Metadaten für den Bereich der Netzwerksicherheit definiert und durch XML-Schema spezifiziert. Dort wird beschrieben, welche Metadaten mit welchen Identifier zusammen auf einem Server veröffentlicht werden dürfen. Ein Beispiel hierfür ist:
Ein Standard Metadatensatz *ip-mac* darf nur zwischen Identifier *mac-address* und dem Identifier *ip-address* platziert werden.

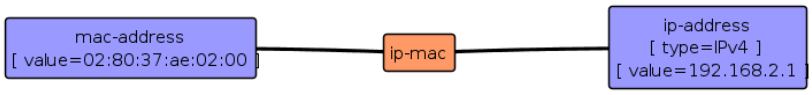


Abbildung 8: Standard Metadatum *ip-mac*, Ausschnitt aus Irongui

Es wird auch festgelegt, welche Attribute die Metadaten enthalten können oder müssen. Wichtige Attribute, die jeder Metadatensatz enthalten muss, sind: *ifmap-publisher-Id*, *ifmap-timestamp*, *ifmap-cardinality* und *lifetime*.

- **ifmap-publisher-id** - Sie ist eine eindeutige ID, die nicht in den Spezifikationen definiert wird, sondern die implementationsabhängig ist. Diese ID identifiziert den IF-MAP Client von dem die Metadaten veröffentlicht wurden.

- **ifmap-timestamp** - Der IF-MAP Server erzeugt eine Zeitmarkierung, an dem die Metadaten veröffentlicht wurden.

- **ifmap-cardinality** - Dieses Attribut kann nur einen der beiden Werte enthalten: *singleValue* oder *multiValue*.

  singleValue - Ein Metadatensatz mit dem Wert *singleValue* darf in einem IF-MAP Graph mit den selben Identifier nur einmal vorkommen, d.h. wenn ein

Metadatensatz in der Konstellation bereits existiert, wird es mit dem neuen Datensatz überschrieben.

multiValue - Anders als bei *„singleValue"* werden hier die alten Metadaten nicht überschrieben, sondern mit den neuen Metadaten ergänzt.

- **lifetime** - Dieses Attribut kann ebenfalls nur einen der beiden Werte: *session* oder *forever* enthalten.

  session - Ein Metadatensatz mit dem Wert *session* wird von dem IF-MAP Server gelöscht, sobald der IF-MAP Client, der die Metadaten veröffentlicht hat, die Verbindung (Session) zu dem IF-MAP Server unterbricht.

  forever - Die Metadaten mit dem Wert *forever* werden so lange auf dem Server gespeichert, bis der IF-MAP Server neu gestartet wird. Dies gilt auch für den Fall, dass der IF-MAP Client, von dem die Metadaten veröffentlicht wurden, die Verbindung beendet hat.

Um die Gültigkeitsüberprüfung realisieren zu können, müssen folgende Kriterien in die Lösung eingearbeitet werden:

- Es muss die Konformität der Zuordnung von Metadaten zu Identifiern und Links, so wie sie in beiden Spezifikationen angegeben sind, überprüft werden.
- Für jedes Metadatum müssen die obligatorischen Attribute, sowohl die genannten allgemeingültigen (**ifmap-publisher-Id**, **ifmap-timestamp** und **ifmap-cardinality**), als auch die jeweils durch das XML-Schema definierten, für jeden Metadatentypen spezifischen, überprüft werden.

4.2. Iron* Softwarekomponenten der Trust@FHH Forschungsgruppe

In dem Kapitel 2.5 werden die Iron* Softwarekomponenten ausführlich beschrieben. Nachfolgend wird im Hinblick auf die Realisierung der Gültigkeitsüberprüfung, die Vor- und Nachteile der jeweiligen Komponenten betrachtet.

4.2.1. Ifmapj

Bei der Klassenbibliothek Ifmapj ergeben sich mehrere Möglichkeiten um die Gültigkeitsüberprüfung zu realisieren.

Integrierung der Gültigkeitsüberprüfung in Ifmapj

Möglichkeit 1 - Bei dieser Lösung wird die Schnittstelle der Ifmapj Klassenbibliothek um weitere Methoden erweitert, über die eine Gültigkeitsüberprüfung explizit durchgeführt werden kann.

Was spricht für die Erweiterung der „Ifmapj“ Schnittstelle?

- Der Entwickler, der Ifmapj verwendet um die IF-MAP Komponente zu entwickeln, kann selber entscheiden, ob er die Gültigkeitsüberprüfung in seiner Implementierung verwenden möchte.
- Die Klassenbibliothek bleibt abwärtskompatibel, da bei der Erweiterung der Schnittstelle die vorhandenen Methoden inkl. der Geschäftslogik unverändert bleiben.

Was spricht gegen die Erweiterung der „Ifmapj“ Schnittstelle?

- Die Erweiterung der Schnittstelle kann dazu führen, das alle bisherigen Clients, die Ifmapj verwenden, in ihrem Code angepasst werden müssen.
- Ifmapj wird schwergewichtiger.

Möglichkeit 2 - Bei dieser Lösung wird die Überprüfungslogik direkt in Ifmapj implementiert ohne die Schnittstelle zu erweitern, sodas die Gültigkeitsüberprüfung automatisch durchgeführt wird.

Was spricht für die implizite Gültigkeitsüberprüfung durch „Ifmapj”?

- Die IF-MAP Komponenten, die bereits mit Ifmapj entwickelt wurden, können durch Austausch der Klassenbibliothek gegen eine neuere Version,
  in die eine Gültigkeitsüberprüfung implementiert wurde, die Metadaten automatisch überprüfen.
- Die mit Ifmapj entwickelte IF-MAP Komponenten können nur die spezifikationskonformen Metadaten veröffentlichen.

Was spricht gegen die implizite Gültigkeitsüberprüfung durch „Ifmapj”?

- Auch bei dieser Lösung kann es zu Problemen mit den Clients, die mit Hilfe von Ifmapj implementiert wurden, führen.
- Ifmapj kann bei Entwicklung von speziellen IF-MAP Clients, die absichtlich fehlerhaften Daten veröffentlichen sollen, zu Schwierigkeiten führen. Die nicht spezifikationskonformen Daten können nicht veröffentlicht werden.

4.2.2. Irond

Was spricht für die Erweiterung des „Irond“ durch die Gültigkeitsüberprüfung?

- Bei der Umsetzung der Gültigkeitsüberprüfung in dem Irond, würde der IF-MAP Server die empfangenden Metadaten, die von einem Client kommen, erst überprüfen und wenn diese spezifikationkonform sind, in dem Server übernehmen. Dadurch kann der Betreiber des IF-MAP Server „Irond“ sicher sein, dass auf dem Server veröffentlichte Metadaten spezifikationskonform sind.

- Als Ergebnis der Überprüfung kann der IF-MAP Server die fehlerhaften Metadaten markieren und den zuständigen Netzwerkadministrator informieren, dass auf dem Server nicht spezifikationskonforme Metadaten veröffentlicht werden sollten.

Was spricht gegen die Erweiterung des „Irond“ durch die Gültigkeitsüberprüfung?

- Bei der Umsetzung der Überprüfung im Irond würde der IF-MAP Server nicht mehr spezifikationskonform sein.

- Wenn ein IF-MAP Client, der ebenfalls einen IF-MAP Graph auf Spezifikationskonformität überprüft, gewollt fehlerhafte Metadaten veröffentlichen will, kann es zu Problemen führen, falls der Server die fehlerhaften Metadaten ausfiltert.

4.2.3. Ifmapcli

Was spricht für die Erweiterung des „Ifmapcli”durch die Gültigkeitsüberprüfung?

- Die vorhandenen Ifmapcli Komponenten müssen nicht angepasst werden, sondern es entsteht ein eigenes schlankes CLI [1] Tool, welches die Gültigkeitsüberprüfung des IF-MAP Graphen durchführt.

Was spricht gegen die Erweiterung des „Ifmapcli“ durch die Gültigkeitsüberprüfung?

- Eine Überprüfung des IF-MAP Graphen ist nur mit Hilfe einer Konsole oder eines Shell Skripts möglich.

[1] CLI = Befehlszeilenschnittstelle (Command Line Interface)

4.2.4. Irongui

Was spricht für die Erweiterung des „Irongui"durch die Gültigkeitsüberprüfung?

- Es entsteht ein multifunktionales Werkzeug mit dem es möglich wird die Metadaten auf Gültigkeit zu überprüfen und diese im Anschluss gefiltert zu visualisieren.
- Das Ergebnis der Überprüfung kann ebenfalls visuell dargestellt werden.

Was spricht gegen die Erweiterung des „Irongui" durch die Gültigkeitsüberprüfung?

- Anpassungen an der GUI[2] und in der logischen Schicht sind notwendig.

4.2.5. Ironcontrol

Was spricht für die Erweiterung des „Ironcontrol"durch die Gültigkeitsüberprüfung?

- Erweitert das Administratorwerkzeug „Ironcontrol" um eine wichtige Funktion.
- Die Überprüfung des IF-MAP Graphen ist auch über mobile Android Geräte möglich.

Was spricht gegen die Erweiterung des „Ironcontrol" durch die Gültigkeitsüberprüfung?

- Da die Smartphones nur begrenzte Ressourcen an Arbeitsspeicher und CPU-Leistung zur Verfügung haben, kann die Performance der Gültigkeitsüberprüfung darunter leiden.

[2]GUI = Grafische Oberfläche(Graphical user interface)

5. Konzept zur Überprüfung eines IF-MAP Graphen

In diesem Kapitel werden verschiedene Lösungsansätze vorgestellt und erläutert, wie die Gültigkeitsüberprüfung am sinnvollsten umgesetzt werden kann. Am Schluss wird eine Entscheidung getroffen, welche Lösung für die exemplarische Implementierung verwendet werden soll.

5.1. Das Prüfverfahren

Die umzusetzende Lösung soll die IF-MAP Metadaten innerhalb eines IF-MAP Graphen analysieren können und feststellen, ob diese den Vorgaben aus den Spezifikationen entsprechen. Es wird erwartet, dass die im Abschnitt 4.1 ausgearbeiteten Aspekte überprüft werden. Für die Lösung bedeutet dies:

1. Es muss überprüft werden, ob die Metadaten an bzw. zwischen den richtigen Identifier-Typen angehängt wurden. Hierfür müssen alle Metadaten mit den passenden Identifiern in den Prüfregeln zusammengefasst werden und für die Überprüfungslogik zur Verfügung gestellt werden.

2. Als nächstes ist es erforderlich, dass die Attribute, die jeder Metadatensatz enthalten muss, überprüft werden. Ein Lösungsansatz dafür wäre die Attribute ebenfalls mit in die oben genannten Prüfregeln aufzunehmen.

   - wenn die geprüften Metadaten die vier Attribute (*ifmap-publisher-Id*, *ifmap-timestamp*, *ifmap-cardinality* und *lifetime*) enthalten, muss bei dem Attribut *ifmap-publisher-Id* nur geprüft werden, ob der dazugehöriger Wert nicht **NULL** ist. Bei dem Attribut *ifmap-timestamp* kann der Wert zusätzlich auf Konformität und auf Plausibilität überprüft werden.
     In dem Zusammenhang muss unterschieden werden, welche Lösung in der Lage ist, welche Attribute zu überprüfen. Zum Beispiel ein Client, der eine Gültigkeitsüberprüfung der zur veröffentlichten Daten durchführt, kann das Attribut **ifmap-timestamp** nicht überprüfen, weil dieses erst auf dem Server gesetzt wird.
   - Wenn ein Metadatensatz überprüft wird, bei dem das Attribut *ifmap-cardinality* den Wert *singleValue* hat, sollte zusätzlich kontrolliert werden, ob an demselben Identifier sich keine weiteren Metadaten mit demselben Typ befinden.
   - Bei dem Attribut *lifetime* muss geprüft werden, ob es den richtigen Wert (**session** bzw. **forever**) besitzt.
   - Ebenso ist es möglich die metadatenspezifischen Attribute, die nur bestimmte Metadaten enthalten, zu überprüfen. Zum Beispiel kann geprüft werden, ob alle Pflichtattribute vorhanden sind, die Min- und Max-Occurence

eingehalten wird oder ob Attribute vorhanden sind, die da nicht hingehören. Dazu werden diese Attribute in den oben genannten Prüfregeln übernommen.

Für die Realisierung der Gültigkeitsüberprüfung stehen mehrere Möglichkeiten zur Verfügung. Von diesen Möglichkeiten werden drei in diesem Abschnitt vorgestellt.

1. Die Gültigkeitsüberprüfung als IF-MAP Client. (Seite 23)
2. Die Gültigkeitsüberprüfung als Erweiterung eines IF-MAP Servers. (Seite 24)
3. Eigene Klassenbibliothek zur Überprüfung des IF-MAP Graphen. (Seite 26)

5.2. Möglichkeit 1: Umsetzung als IF-MAP Client

Bei der Lösung „Gültigkeitsüberprüfung als IF-MAP Client“ wird ein neuer Client entwickelt, der in der Lage sein wird, nach Metadaten auf einem IF-MAP Server zu suchen, diese Metadaten auf Spezifikationskonformität zu prüfen und den Benutzer über das Ergebnis der Prüfung zu informieren.

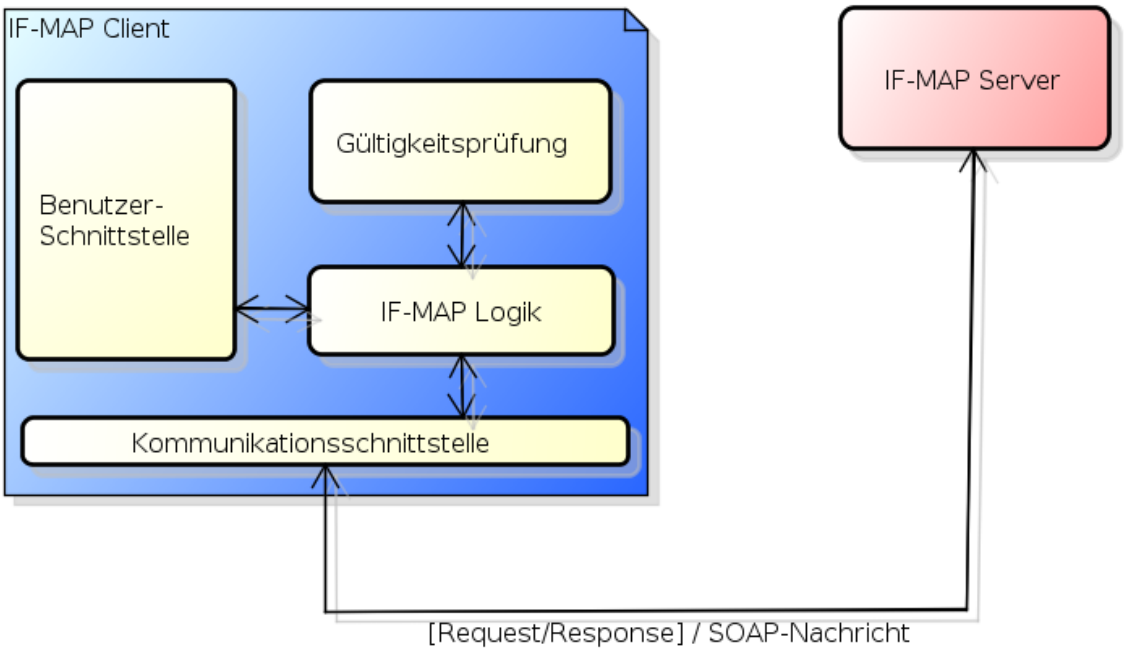


Abbildung 9: Gültigkeitsüberprüfung als IF-MAP Client

Der IF-MAP Client kann mit Hilfe von Ifmapj realisiert werden. Wie im Abschnitt 2.5.2 erwähnt, bietet Ifmapj mehrere Möglichkeiten um die Metadaten auf einem IF-MAP Server auszulesen. Mit der Ifmapj Operation **search** kann der IF-MAP Graph

mit Hilfe der Filterfunktion oder durch Angabe der Suchtiefe benutzerspezifisch geformt als ein **SearchResult**[3] ausgegeben werden.
Bei der Verwendung der **subscribe** Operation kann der Client über Veränderungen der abonnierten Daten informiert werden. Die Information, die der Client empfängt, wird von dem IF-MAP Server in Form eines **PollResult** [4] bereit gestellt.
Nach dem Zerlegen und der Auswertung des **Results** kann das Ergebnis visuell, als Log Datei oder als Rückgabeparameter ausgegeben werden. Der Vorteil dieser Lösung besteht darin, dass sie realisierbar ist, ohne das der IF-MAP Server und das IF-MAP Protokoll an sich angepasst werden müssen.

Ein Nachteil ist, dass der Client von den nicht spezifikationskonformen Daten erst informiert wird, wenn die Daten auf dem IF-MAP Server veröffentlicht sind.

5.3. Möglichkeit 2: Erweiterung des IF-MAP Servers

In diesem Abschnitt wird die Lösung „Gültigkeitsüberprüfung als Erweiterung des IF-MAP Servers“ vorgestellt. Bei dieser Lösung wird ein IF-MAP Server um die Prüfungskomponente erweitert. Diese Erweiterung ermöglicht dem Server, die Metadaten die von einem IF-MAP Client versendet werden, auf Spezifikationskonformität zu überprüfen und anhand des Prüfergebnisses eine Entscheidung zu treffen, ob diese Daten veröffentlicht werden.

Anhand der Abbildung 10 wird der Ablauf der Gültigkeitsüberprüfung erläutert.

1. Ein IF-MAP Client sendet an den IF-MAP Server ein **PublishRequest**.
2. Nach dem der **PublishRequest** empfangen wurde, validiert die Prüfungslogik die empfangenen Daten.
3. Wenn die Metadaten nicht spezifikationskonform sind, kann der Server mit Hilfe des **ErrorResults**, der in der Spezifikation [2] definiert ist, den Client darüber informieren.
4. Wenn die Metadaten den Spezifikationen entsprechen, werden diese auf dem Server veröffentlicht.

[3] SearchResult = Das Ergebnis der ifmapj *„search“* Funktion und auch das allererste Ergebnis der *subscribe* Funktion.
[4] PollResult = Das Ergebnis der ifmapj *subscribe* Funktion.

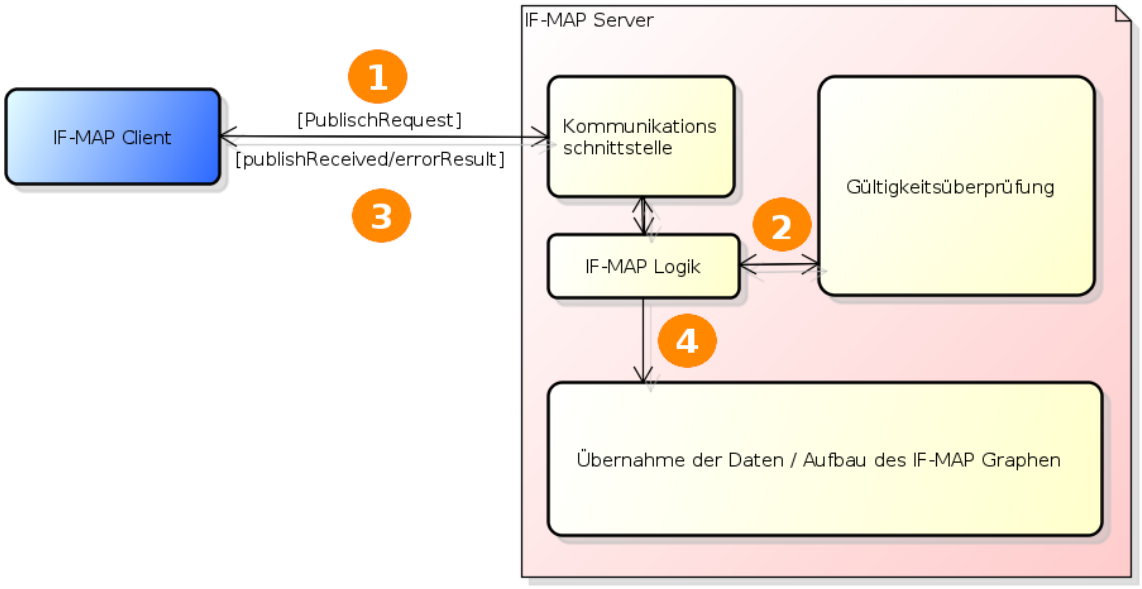


Abbildung 10: Gültigkeitsüberprüfung als Erweiterung des IF-MAP Servers.

Um die Gültigkeitsüberprüfung in einem IF-MAP Server realisieren zu können, sollen die empfangene Daten vor der Veröffentlichung durch eine Überprüfungslogik validiert werden. Hierfür wird eine zusätzliche Schnittstelle implementiert, über die die empfangene Daten an die Überprüfungslogik transportiert werden. In dem Fall, wenn die Überprüfung negativ ausfällt, wird das passende **ErrorResults** erzeugt und an den Client gesendet.

Der Vorteil dieser Lösung ist, dass die auf diesem IF-MAP Server veröffentlichte Daten zuvor auf Konformität geprüft werden. Dadurch wird sicher gestellt, dass in dem betriebenem Netzwerk nur die Metadaten veröffentlicht werden, die den Vorgaben der TCG Spezifikationen entsprechen, was mit der Möglichkeit 1 „Umsetzung als IF-MAP Client“ nicht abgedeckt werden kann.

Die Nachteile dieser Lösung sind zum Einem, dass der IF-MAP Server durch die genannte Umsetzung nicht mehr spezifikationskonform ist und zum Anderen, dass falsch implementierte IF-MAP Komponenten keine Daten auf diesem Server veröffentlichen können, was den Betrieb von TNC[5] einschränkt.

[5]TNC = Trusted Network Connect

5.4. Möglichkeit 3: Eigene Java Klassenbibliothek zur Überprüfung des IF-MAP Graphen

Als dritte Möglichkeit wird eine Lösung vorgestellt, wo eine eigene Java Klassenbibliothek entwickelt wird, die Methoden für die Überprüfung des IF-MAP Graphen zur Verfügung stellt. Bei der Auswahl der Programmiersprache fiel die Entscheidung auf Java, da Erstens die Anforderung **A5** damit abgedeckt wird und Zweitens, weil Ifmapj auch auf Java basiert.

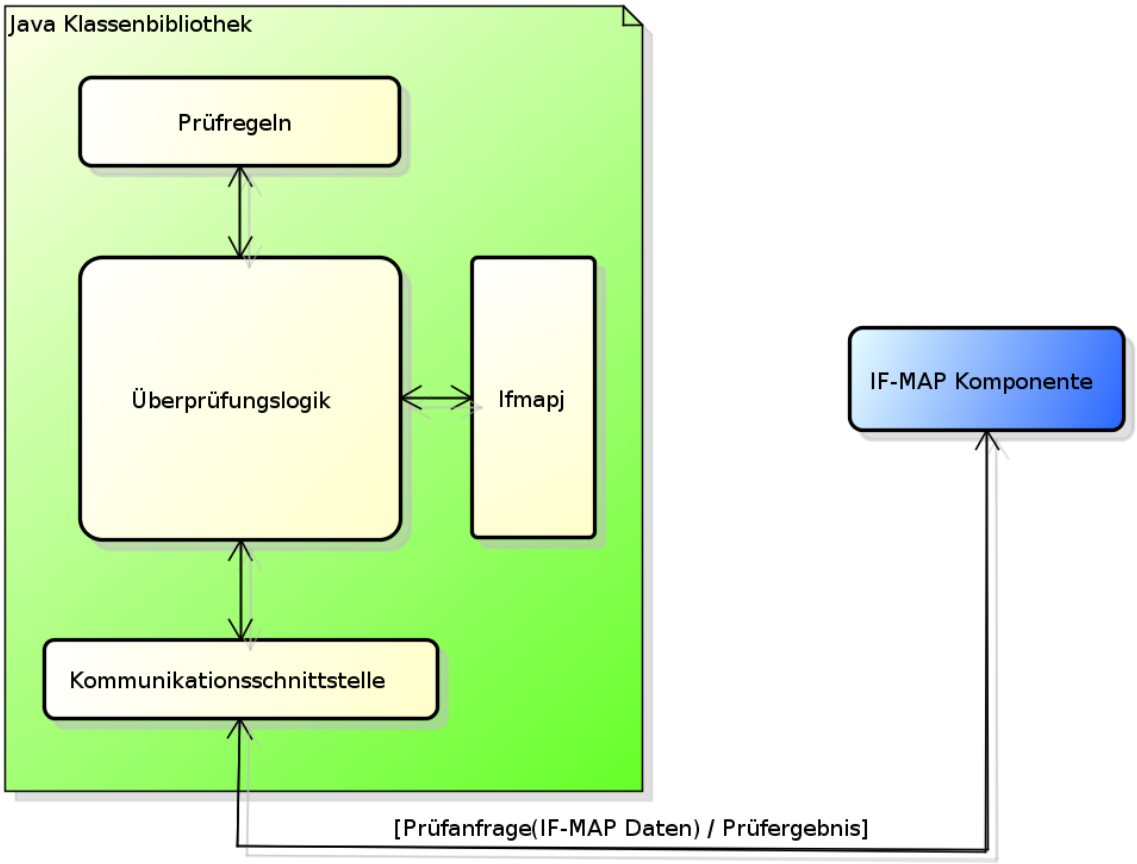


Abbildung 11: Java Klassenbibliothek für die Überprüfung des IF-MAP Graphen.

Als Schnittstelle kann eine Methode implementiert werden, die als Übergabeparameter einen IF-MAP Graphen erwartet, der beispielsweise in Form von einem **SearchResult** übergegeben werden kann. Durch den Einsatz von Ifmapj kann der **SearchResult** zerlegt werden und die einzelne Metadaten durch die Überprüfungslogik validiert werden. Als Rückgabeparameter kann das Prüfergebnis in Form einer Textnachricht (String) zurückgegeben werden. Damit die Anforderung **A4** mit dieser Lösung abgedeckt wird,

kann ein eigenes Java Objekt erzeugt werden, welches die oben genannte Textnachricht und zusätzlich die fehlerhaften Metadaten enthält.
Ein Vorteil dieser Lösung gegenüber den zuvor vorgestellten Möglichkeiten ist, dass die Gültigkeitsüberprüfung als ein Werkzeug zur Verfügung steht, welches von beliebigen IF-MAP Komponenten genutzt werden kann.

Ein Nachteil dieser Lösung ist, dass für die Gültigkeitsüberprüfung eine IF-MAP Komponente implementiert werden muss, die diese Klassenbibliohtek verwendet.

5.5. Erweiterbarkeit des Prüfverfahren

Die Erweiterbarkeit der Gültigkeitsüberprüfung soll durch Auslagerung der Prüfkriterien (Prüfregeln) in eine XML Datei ermöglicht werden. Mit Hilfe der neu erstellten XML Schema können die vom Benutzer eingegebenen Regeln validiert werden.

Durch die Auslagerung der Prüfregeln kann die Gültigkeitsüberprüfung mit neu erstellten Metadaten erweitert werden. Dazu werden die IF-MAP Metadaten in eine XML Datei eingefügt und stehen automatisch für die Überprüfung zur Verfügung.

5.6. Bewertung der Lösungsansätze

In diesem Abschnitt werden die drei vorgestellte Lösungseinsätze (5.2 bis 5.4) im Bezug auf die im Kapitel 3 festgelegte Anforderungen bewertet.

Übersicht über die Anforderungen an die Gültigkeitsprüfung.

- **Anforderung A1** - Die Prüfung der Konformität an den Vorgaben der Spezifikation
- **Anforderung A2** - Die Erweiterbarkeit des Prüfverfahrens
- **Anforderung A3** - Die Schnittstelle
- **Anforderung A4** - Das Prüfergebnis
- **Anforderung A5** - Die Plattformunabhängigkeit
- **Anforderung A6** - Performance

Die Anforderung **A1** kann mühelos in allen drei Lösungsansätzen realisiert werden indem die Gültigkeitsüberprüfung in die vorhandene Logik integriert wird, beziehungsweise, als eigene Logik implementiert wird. Ebenso, wie in dem Abschnitt 5.5 beschrieben, kann die Anforderung **A2** in den vorgestellten Lösungsansätzen umgesetzt

werden. Für die Anforderung **A3** muss zwischen den drei Lösungsansätzen unterschieden werden.

- Lösungsansatz 1 - „Gültigkeitsüberprüfung als IF-MAP Client“. Bei dieser Lösung kann die Schnittstelle beliebig gestaltet werden.
  Es kann zum Beispiel eine GUI[6] sein, die als Eingabeparameter die für die Gültigkeitsüberprüfung notwendigen Daten oder bei einer CLI [7] Anwendung Eingaben einer Konsole erwartet.

- Lösungsansatz 2 - „Gültigkeitsüberprüfung als Erweiterung des IF-MAP Servers“. Hier wird keine Schnittstelle benötigt, da die Gültigkeitsüberprüfung in die vorhandene Logik integriert wird und die Validierung im Hintergrund automatisiert abläuft.

- Lösungsansatz 3 - „Eigene Java Klassenbibliothek zur Überprüfung des IF-MAP Graphen“. Wie in dem Abschnitt 5.4 erwähnt, kann eine Schnittstelle realisiert werden, die als Übergabeparameter einen **SearchResult**[8] erwartet, der überprüft werden kann. Als Rückgabeparameter kann das Prüfergebnis in Form einer Textnachricht (String) zurückgegeben werden.
  Die Schnittstelle verwendet die bereits vorhandenen Ifmapj Objekte.

Bei der Anforderung **A4** kann das Prüfergebnis mit der ersten Lösung über der GUI beliebig kompliziert ausgegeben werden. Bei der zweiten Lösung wird das Prüfergebnis nur im negativen Fall in einer abstrakten Form zur Verfügung gestellt. Die dritte Lösung kann das Prüfergebnis beliebig gestalten.

Die Anforderung **A5** kann dadurch realisiert werden, indem alle drei Lösungen in Java implementiert werden und deshalb auf vielen verfügbaren Betriebssystemen, ohne zusätzliche Anpassung, eingesetzt werden können.

Entscheidung für die Implementierung

Für die Umsetzung der exemplarischen Implementierung habe ich mich, unter Berücksichtigung der Bewertung der einzelnen Lösungsansätze, für die dritte Möglichkeit „Eigene Java Klassenbibliothek zur Überprüfung des IF-MAP Graphen“ entschieden. Mit dieser Lösung können die, in der Anforderungsanalyse gestellte Anforderungen mit der höchsten Priorität 3, mit wenig Aufwand umgesetzt werden. Diese Lösung bietet die größte Flexibilität im Vergleich zu den anderen Möglichkeiten.

[6] GUI = Grafische Oberfläche(Graphical user interface)
[7] CLI = Befehlszeilenschnittstelle (Command Line Interface)
[8] SearchResult = Das Ergebnis der ifmapj *search* Funktion und auch das allererste Ergebnis der „*subscribe*“ Funktion.

6. Design und Implementierung

In diesem Kapitel wird die Realisierung des Konzepts 5 genauer erläutert. Am Anfang wird die Klassenbibliothek „IfmapValidityChecker“ vorgestellt. Als nächstes wird die Schnittstelle definiert und die Architektur an Hand einer Abbildung erklärt. Am Schluss werden die Prüfregeln behandelt und eine Erweiterung der Gültigkeitsüberprüfung mit „Vendor-specific Metadata“ an einem Beispiel vorgeführt.

6.1. Klassenbibliothek „IfmapValidityChecker“

Die Java Klassenbibliothek zur Umsetzung der Prüfung von IF-MAP Metadaten wurde unter dem Namen „IfmapValidityChecker“ entwickelt. Diese Klassenbibliothek enthält Methoden, mit deren Hilfe eine Gültigkeitsüberprüfung der IF-MAP Metadaten durchgeführt werden kann. Bei der Realisierung der Klassenbibliothek wurde die vollständige IF-MAP Kommunikation mit Hilfe von Ifmapj umgesetzt.

IfmapValidityChecker im Überblick

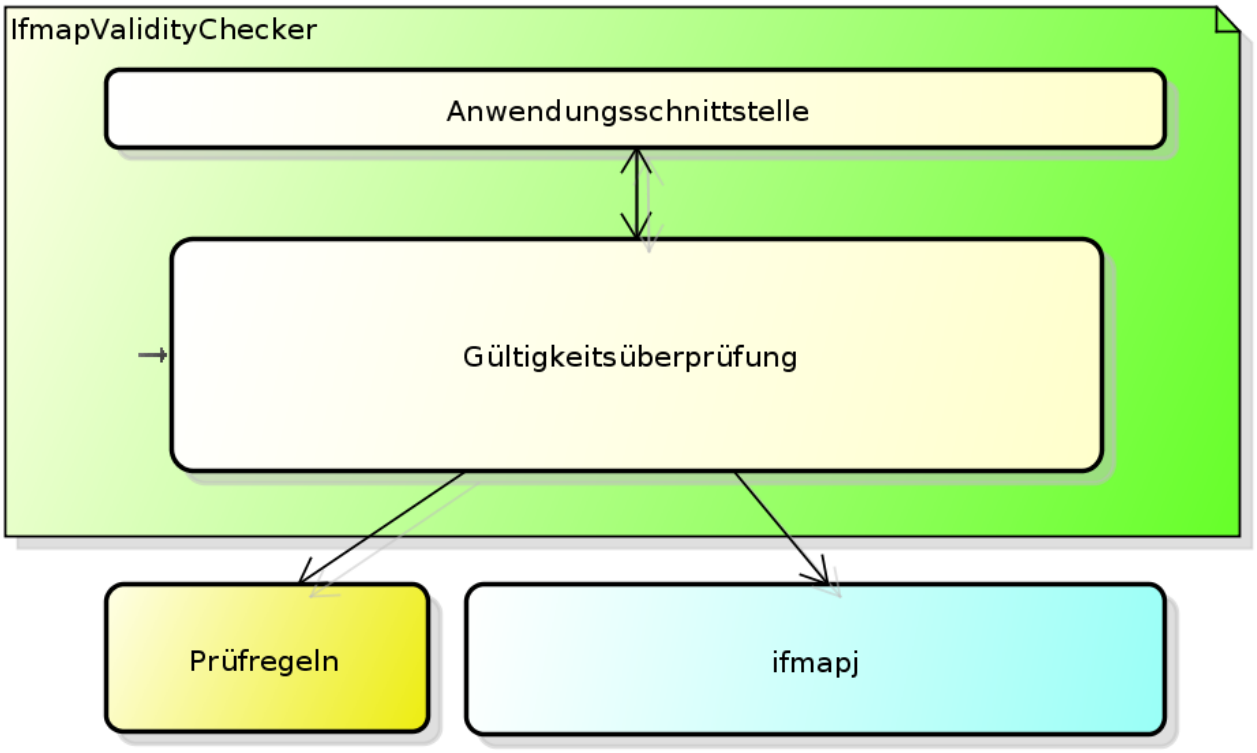


Abbildung 12: Grobe Struktur der Klassenbibliothek

Die Klassenbibliothek besteht aus einer Anwendungsschnittstelle über die die Gültigkeitsüberprüfung durchgeführt werden kann. Die Prüfregeln, die in eine XML Datei ausgelagert wurden, ermöglichen Erweiterungen ohne den Code zu verändern und neu kompilieren zu müssen. Die Überprüfungslogik validiert mit Hilfe der Prüfregeln die empfangenen Metadaten. Zur Anbindung an IF-MAP wurde die Bibliothek Ifmapj verwendet.

6.2. Definition der Schnittstelle

Bei der Schnittstelle werden zwei statische Methoden implementiert, über die eine Gültigkeitsüberprüfung gesteuert werden kann. Bevor die Schnittstelle im Einzelnen vorgestellt wird, sind noch folgende Begriffe zu erläutern.

- **SearchResult** - Ist das Ergebnis der Ifmapj *search* Funktion und auch das allererste Ergebnis einer *subscription* Funktion.

- **ResultItem** - Ist ein Element des SearchResults, der einen Ausschnitt aus dem IF-MAP Graph enthält, einen Metadatensatz und den dazugehörigen Identifier.

- **CheckResult** - Ist das Ergebnis der Gültigkeitsüberprüfung, welches bei Verwendung der Klassenbibliothek „IfmapValidityChecker“ zur Verfügung gestellt wird. Ein **CheckResult** besteht aus einer Textnachricht, die aus der XML Datei ausgelesen wird, die die Prüfregeln enthält und aus einem **ResultItem** mit nicht spezifikationskonformen Daten. Wenn die Überprüfung keine Abweichung festgestellt hat, wird das **CheckResult** nur mit der Textnachricht ausgegeben.

- **CheckRules.xml** - Ist eine XML Datei, die die Prüfregeln für die Gültigkeitsüberprüfung enthält. Diese ist für die „Standard-Metadaten für den Bereich der Netzwerksicherheit“ zusammen mit der dazugehörigen **CheckRulesSchema.xsd** in der Bibliothek integriert. Diese XML Datei kann von dem Benutzer modifiziert werden und mit der dafür vorgesehenen Methode für die Überprüfung geladen werden.

- **CheckRulesSchema.xsd** -Ist ein XML Schema, das die Struktur der **CheckRules.xml** definiert. Dieses XML Schema wird für die Validierung der Prüfregeln verwendet und soll den Benutzer bei der Modifikation der **CheckRules.xml** unterstützen.

Wie bereits beschrieben, besteht die Schnittstelle aus zwei statischen Methoden: **checkStandartmetadata** und **check**.

Method Summary

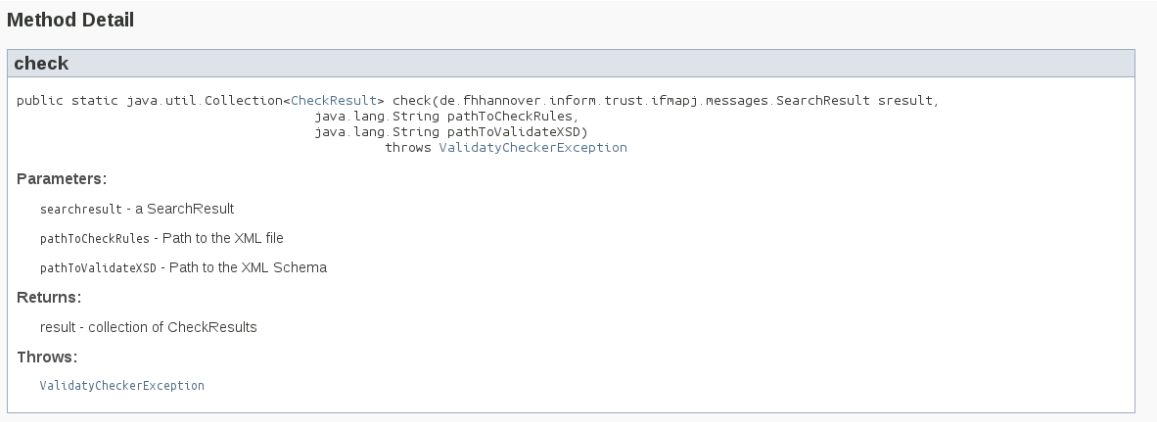
Methods

Modifier and Type	Method and Description
static java.util.Collection<CheckResult>	check(de.fhhannover.inform.trust.ifmapj.messages.SearchResult sresult, java.lang.String pathToCheckRules, java.lang.String pathToValidateXSD)
static java.util.Collection<CheckResult>	checkStandartNSmetadata(de.fhhannover.inform.trust.ifmapj.messages.SearchResult sresult) This method use the internal CheckRules

Abbildung 13: Die Schnittstelle im Überblick. Quelle: IfmapValidityChecker javadoc

1. **checkStandartNSmetadata** - Diese Methode ist für die „Standard-Metadaten für den Bereich der Netzwerksicherheit“ gedacht und verwendet die in der Bibliothek enthaltenen Prüfregeln, die in der CheckRules.xml festgelegt wurden. Als Übergabeparameter erwartet diese Methode ein **SearchResult**, der eine IF-MAP Graph enthält, welcher durch den Benutzer mit Hilfe der **Search** Operation erzeugt wurde.
   Als Ergebnis liefert die Methode eine Liste von **CheckResults**.

2. **check** - Diese Methode verwendet dieselbe Überprüfungslogik wie **checkStandartNSmetadata**. Sie erwartet zwei zusätzliche Übergabeparameter: **pathToCheckRules** und **pathToValidateXSD**, die den Pfad zu der **CheckRules.xml** und zu der **StandardMetadata.xsd** enthalten.
   Diese Methode soll dem Benutzer die Möglichkeit geben eigenen Prüfregeln zu definieren, um die Gültigkeitsüberprüfung um eigene Metadaten beziehungsweise eigene Identifier zu erweitern. Hierdurch kann die Überprüfung der „Vendor-specific Metadata“ und den „Extended Identifiers“ abgedeckt werden.



Method Detail

check

```
public static java.util.Collection<CheckResult> check(de.fhhannover.inform.trust.ifmapj.messages.SearchResult sresult,
                                                      java.lang.String pathToCheckRules,
                                                      java.lang.String pathToValidateXSD)
                                                               throws ValidatyCheckerException
```

Parameters:

`searchresult` - a SearchResult

`pathToCheckRules` - Path to the XML file

`pathToValidateXSD` - Path to the XML Schema

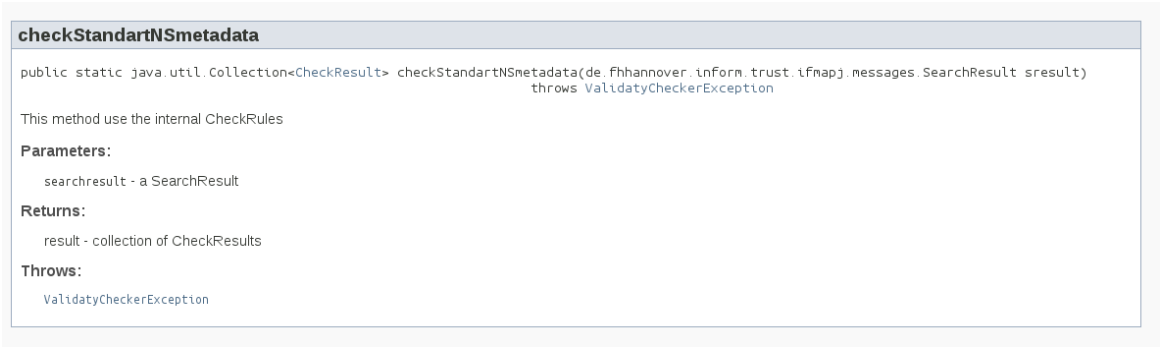
Returns:

result - collection of CheckResults

Throws:

`ValidatyCheckerException`

Abbildung 14: Methode check(...) in Details. Quelle: IfmapValidityChecker javadoc



checkStandartNSmetadata

```
public static java.util.Collection<CheckResult> checkStandartNSmetadata(de.fhhannover.inform.trust.ifmapj.messages.SearchResult sresult)
                                                                         throws ValidatyCheckerException
```

This method use the internal CheckRules

Parameters:

searchresult - a SearchResult

Returns:

result - collection of CheckResults

Throws:

ValidatyCheckerException

Abbildung 15: Methode checkStandartNSmetadata(...) in Details.
Quelle: IfmapValidityChecker javadoc

6.3. Architektur

Für das besseres Verständnis wird die Klassenbibliothek „IfmapValidityChecker“ in Teilsysteme zerlegt und genauer erläutert.

Wie in der Abbildung 16 zu sehen ist, ist die Klassenbibliothek in folgende Teilbereiche unterteilt.

- **Schnittstelle** - Die im Abschnitt 6.2 beschriebene Schnittstelle wird mit der Klasse IfmapValidityChecker realisiert.
- **Überprüfungslogik** - Der Teilbereich enthält zwei Klassen:
  CheckResultImpl, die das Interface **CheckResult** implementiert, ist für den Inhalt des Prüfungsergebnis zuständig. Die Klasse **Check** bildet die eigentliche Logik ab und verwendet die Hilfsklassen für die Durchführung der Gültigkeitsüberprüfung.
- **Hilfsklassen** - Hier sind alle Hilfsklassen untergebracht, die von dem Prüfalgorithmus genutzt wird.
  - **CheckHelper** - Stellt dem Prüfalgorithmus Methoden bereit, die für eine Überprüfung des IF-MAP Graphen notwendig sind.
  - **XMLHelper** - Diese Klasse stellt Methoden bereit, mit deren Hilfe die Prüfkriterien aus der XML Datei ausgelesen werden.
  - **Logging** - Ermöglicht Lognachrichten zu erstellen und in eine Log Datei abzuspeichern. Diese Funktion wird mit Hilfe der Java Logging API umgesetzt.
  - **SpecificAttribute** - Ist eine Klasse, mit der die metadatenspezifischen Attribute repräsentiert werden.

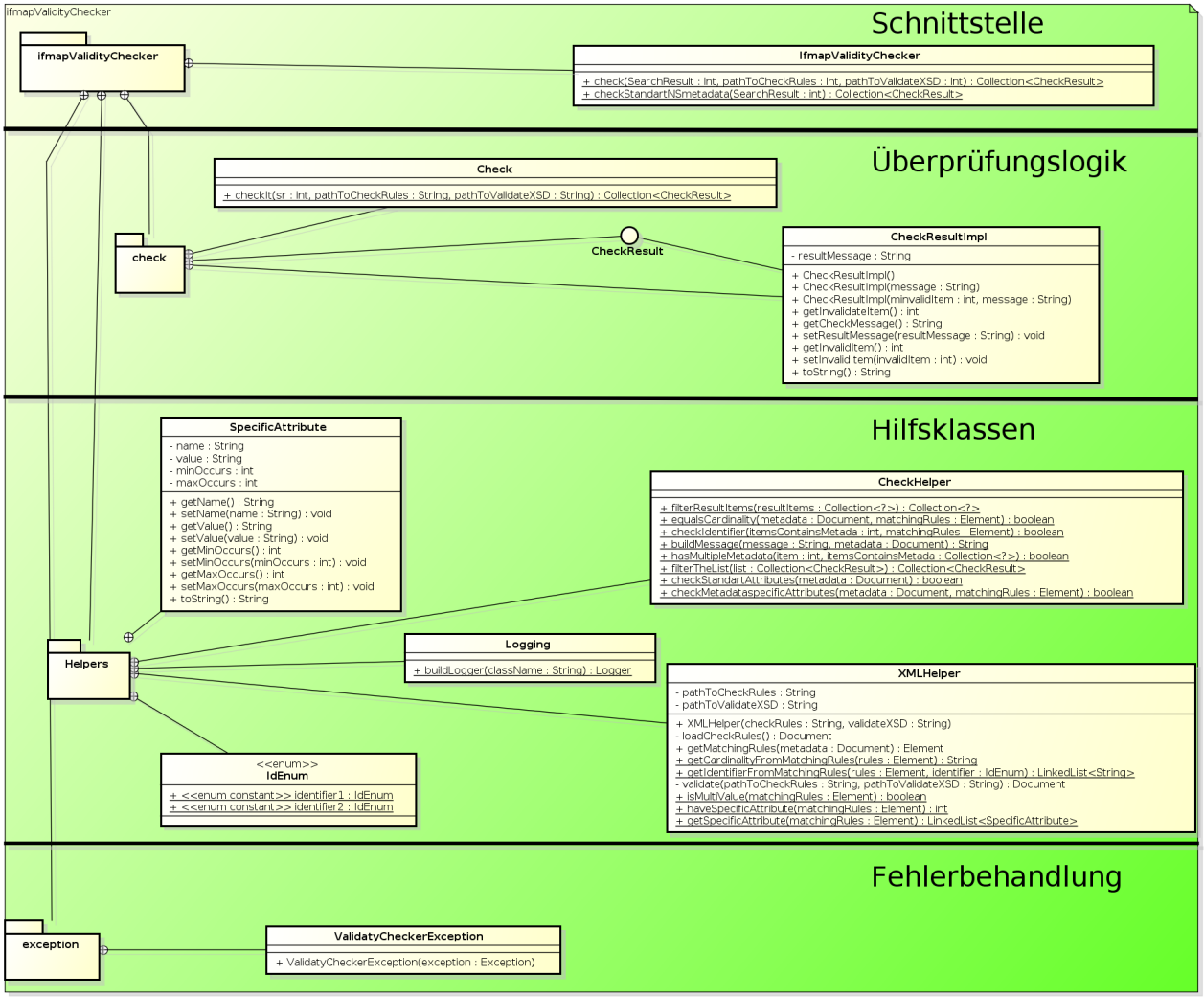


Abbildung 16: Teilbereiche der „IfmapValidityChecker“ Klassenbibliothek

- **Fehlerbehandlung** - Der Teilbereich enthält die „ValidatyCheckerException“ Klasse. Mit Hilfe dieser Klasse werden die auftretende Exceptions nach außen geleitet.

Um starke Kohäsion zu erreichen, werden die Klassen so aufgebaut, dass diese nur für bestimmte Aufgaben verantwortlich sind und man an den Namen der Klasse ableiten kann, welche Aufgaben diese Klasse hat.
Für die genaue Beschreibung der Klassen wurde im Anhang ein Klassendiagramm hinzugefügt, siehe Abschnitt A.1.

6.4. Prüfregeln

Wie in dem Abschnitt 5.5 erwähnt, werden die Prüfregeln in eine XML Datei ausgelagert. In diesem Abschnitt wird der Aufbau der **CheckRules.xml** Datei, in der die Prüfregeln enthalten sind, erläutert. Anhand eines Beispieles wird die Gültigkeitsüberprüfung um neue Metadaten erweitert.

6.4.1. Aufbau der CheckRules.xml

Die XML Datei ist so aufgebaut, dass die Standard-Metadaten für den Bereich der Netzwerksicherheit als einzelne Elemente definiert werden, die wiederum weitere Unterelemente enthalten.

Listing 4: Prüfregel für *ip-mac* (Ausschnit aus CheckRules.xml)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2   ...
3   <metadata name="ip-mac">
4       <ifmap:ifmap-cardinality name="multiValue"/>
5       <ifmap:identifier1>ip</ifmap:identifier1>
6       <ifmap:identifier2>mac</ifmap:identifier2>
7       <ifmap:message>...error message...</ifmap:message>
8   </metadata>
9   ...
```

Wie in dem Codebeispiel Listing 4 zu sehen ist, handelt es sich hier um eine Überprüfungsregel für den Metadatensatz von Typ *ip-mac* (Zeile 3). Als Erstes werden die Metadatenattribute definiert, die dieser Metadatentyp enthalten muss. Da laut der TCG Spezifikation [1], der Metadatentyp *ip-mac* keine metadatenspezifische Attribute hat, wird in dem Codebeispiel (Zeile 4) nur der Attribut *ifmap-cardinality* definiert. In den Zeilen 5 und 6 werden die Identifier definiert, zwischen denen dieser Metadatentyp veröffentlicht werden darf. Schließlich ist in der Zeile 7 eine Textnachricht definiert, die im Fehlerfall bei der Überprüfung dieses Metadatentyps ausgegeben wird.

Listing 5: Prüfregel für *wlan-information*(Ausschnitt aus CheckRules.xml)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2   ...
3   <metadata name="wlan-information">
4       <ifmap:ifmap-cardinality name="singleValue"/>
5       <ifmap:specific-attribute name="ssid-unicast-security"
            value="" min="1" max="100" />
```

```
6       <ifmap:specific-attribute name="ssid-group-security"
            value="" min="1" max="1" />
7       <ifmap:specific-attribute name="ssid-management-
            security" value="" min="1" max="100" />
8       <ifmap:identifier1>ar</ifmap:identifier1>
9       <ifmap:identifier2>dev</ifmap:identifier2>
10      <ifmap:message>...error message...</ifmap:message>
11  </metadata>
12  ...
```

Wie in dem Beispiel Listing 5 zu sehen ist, enthält die Prüfregel für dem Metadatentyp *wlan-information*, die metadatenspezifischen Attribute: *ssid-unicast-security*, *ssid-group-security* und *ssid-management-security*, die vorhanden sein müssen.
In den Zeilen 5 bis 7 sind die drei Attribute als *specific-attribute* Elemente definiert. Die Elemente enthalten nicht nur die Bezeichnung das Attribute, sondern auch den Wert (value), wie oft der Attribut mindestens (Min) und maximal (Max) vorhanden sein muss. Für die Attribute *ssid-unicast-security* und *ssid-management-security* ist der Wert (Max) auf „100“ gesetzt, was bedeuten soll, dass diese Attribute beliebig oft vorkommen können.

Listing 6: Prüfregel für *discovered-by*(Ausschnit aus CheckRules.xml)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2    ...
3    <metadata name="discovered-by">
4        <ifmap:ifmap-cardinality name="singleValue"/>
5        <ifmap:identifier1>ip</ifmap:identifier1>
6        <ifmap:identifier1>mac</ifmap:identifier1>
7        <ifmap:identifier2>dev</ifmap:identifier2>
8        <ifmap:message>...error message...</ifmap:message>
9    </metadata>
10   ...
```

Zum Schluss wird eine Prüfregel gezeigt, mit der ein Metadatentyp *discovered-by* überprüft werden soll, der zwischen mehreren Identifiern angehängt werden darf. In den Zeilen 5 bis 7 werden diese Identifier festgelegt.
Laut der Spezifikation [1] soll der Metadatentyp *discovered-by* zwischen den Identifiern *ip-address* und *device* ebenso wie zwischen *mac-address* und *device* angehängt werden können.

6.4.2. Beispiel für Erweiterung mit „Vendor-specific Metadata“

In der aktuellen Version 2.1 vom Mai 2012 der TCG Spezifikation [2] kann der Benutzer eigene Metadatentypen „Vendor-specific Metadata“ definieren. Damit diese Metadaten auf Spezifikationskonformität überprüft werden können, wird die dazu passende Überprüfungsregel in die **CheckRules.xml** eingetragen. Wie an dem nachfolgendem Beispiel zu sehen ist, wird die neue Überprüfungsregel dazu in die **CheckRules.xml** eingetragen.

Listing 7: Prüfregel für den „*vendormetadata01*“ Metadatentyp

```
1 <?xml version="1.0" encoding="UTF-8"?>
2   ...
3   <metadata name="vendormetadata01">
4       <ifmap:ifmap-cardinality name="multiValue"/>
5       <ifmap:identifier1>ar</ifmap:identifier1>
6       <ifmap:message>Recommended publish this Vendor Specific
            metadata only with access-request </ifmap:message>
7   </metadata>
8   ...
```

Wichtig:

Sind keine passende Prüfregeln in der **CheckRules.xml** enthalten, wird der Benutzer mit der Textnachricht „No matching rules for this metadata“ darüber informiert.

7. Tests

In diesem Kapitel werden Tests beschrieben, wie die Klassenbibliothek getestet wurde, um feststellen zu können, ob die Metadaten, die Spezifikationskonform sind, auch so erkannt werden und ob Metadaten, die von den Vorgeben den TCG Spezifikationen [1] [2] abweichen, als falsch erkannt werden.

7.1. Testumgebung

Alle beschriebene Test wurden auf einem Ubuntu 12.10 64 Bit Betriebssystem mit Java (Version 1.7.0_25) durchgeführt.
Um die Klassenbibliothek „IfmapValidityChecker“ testen zu können, wurde eine neuer Client implementiert, der folgende Aufgaben übernahm:

1. Verbindung zu dem IF-MAP Server aufbauen.

2. Auf dem Server nach Metadaten zu suchen.

3. Durchführung der Gültigkeitsüberprüfung.

4. Ausgabe des Prüfergebnisses auf einer Konsole.

Als MAP-Server Implementierung wurde Irond (aktuelle Version 0.4.0) verwendet und für die Veröffentlichung der Metadaten auf dem Irod kam Ironcontrol zum Einsatz.

7.2. Durchgeführte Tests

7.2.1. Test 1: Erkennung der korrekten Metadaten

Eingabe

Als Erstes wurden auf dem IF-MAP Server die Standard-Metadaten für den Bereich der Netzwerksicherheit so veröffentlicht, dass diese den Spezifikationsvorgaben entsprachen.

Ausgabe

Als Ergebnis der Gültigkeitsüberprüfung wurde folgende Nachricht ausgegeben:
„The checked searchresult is valid“.
Daraus folgt, dass richtige Metadaten auch als richtig erkannt wurden.

7.2.2. Test 2: Erkennung der fehlerhaften Metadaten

Eingabe

Als Nächstes wurden dieselben Standard-Metadaten für den Bereich der Netzwerksicherheit veröffentlicht, bei denen absichtlich Fehler eingebaut wurden. Es wurden falsche Identifier ausgewählt und metadatenspezifische Attribute, die vorhanden sein müssen, absichtlich entfernt.

Ausgabe

Bei jedem geprüften Metadatensatz wurde die richtige Fehlermeldung ausgegeben. In dem Fall, wo die Beziehung zwischen dem Metadatensatz und den Identifiern inkorrekt war, wurde folgende Meldung ausgegeben:
„Clients MUST publish this metadata only between: Identifier ... and Identifier ...“.
Für die Metadaten, bei dem die metadatenspezifische Attribute gefehlt haben, wurde folgende Fehlermeldung ausgegeben: „Please check the metadata-specific-attributes!“.

7.2.3. Test 3: Richtige Erkennung der Metadaten, die zwischen mehreren Identifiern plaziert werden können

Eingabe

Bei folgenden Metadaten wurde getestet, ob die Beziehung zwischen den Metadaten und den dazugehörigen Identifiern richtig erkannt wurde. Es wurden alle richtigen und mindesten eine fehlerhafte Kombination von Metadatensatz und Identifier auf dem Server veröffentlicht.

Ausgabe

Die Ausgabe enthielt, nur bei den fehlerhaften Kombinationen eine Fehlermeldung.

7.2.4. Test 4: Überprüfung der „Vendor-specific Metadata“

Eingabe 1

Für diesen Test wurde ein neuer Metadatentyp definiert und auf dem Server veröffentlicht.

Eingabe 2

Hierfür wurde zusätzlich eine neue Prüfregel erstellt, die für den neuen Metadatentyp zuständig war.

Ausgabe 1

Da die Gültigkeitsüberprüfung keine passende Prüfregeln für den neuen Metadatentyp gefunden hat, wurde folgende Fehlermeldung ausgegeben:
„No matching rules for this metadata“.

Ausgabe 2

Nachdem die passende Prüfregel geladen wurde, konnte die Gültigkeitsüberprüfung keinen Fehler feststellen.

8. Reflexion der Anforderungen

In diesem Kapitel wird untersucht inwieweit das Konzept und die exemplarische Implementierung, die im Abschnitt 3 festgelegte Anforderungen, umsetzen.

8.1. Funktionale Anforderungen

8.1.1. A1: Prüfung der Konformität zu den Vorgaben der Spezifikation

Diese Anforderung bestand darin feststellen zu können, inwieweit die auf einem IF-MAP Server veröffentlichte Metadaten die Vorgaben der Spezifikation einhalten.
Die Anforderung **A1** konnte dadurch umgesetzt werden, indem die Vorgaben der Trusted Computing Group Spezifikationen als Prüfregeln definiert wurden und dadurch für die Überprüfung des IF-MAP Graphen zur Verfügung standen. Wie in dem Kapitel 7 beschrieben, konnte diese Anforderung erfolgreich umgesetzt werden.

8.1.2. A2: Erweiterbarkeit des Prüfverfahrens

Bei dieser Anforderung ging es darum, dass die Überprüfung des IF-MAP Graphen, auch bei Erweiterung der Vorgaben aus nachfolgenden Spezifikationen, erweiterbar sein sollte.
Für die Umsetzung der Anforderung **A2** wurde im Abschnitt 5.5 vorgeschlagen, die Prüfregeln in eine XML Datei auszulagern. Durch die Auslagerung der Prüfregeln konnte erreicht werden, dass die Gültigkeitsüberprüfung um neue Metadaten und ihre Beziehung zu Identifiern erweitert werden konnte, ohne den Code anpassen zu müssen.

8.1.3. A3: Die Schnittstelle

Damit das Problem mit vielen Übergabeparametern gelöst werden konnte, wurde ein Lösungsansatz betrachtet, wo der Benutzer sich selber um die Anschaffung der IF-MAP Metadaten kümmert und als Übergabeparameter einen *SearchResult*[9] an die Gültigkeitsüberprüfung übergibt. Dadurch kann der Benutzer festlegen, welche IF-MAP Daten für die Gültigkeitsüberprüfung zur Verfügung gestellt werden. Durch die Implementierung der beiden Methoden **check(Searchresult, String, String)** und **checkStandartNSmetadata(Serchresult)**, die nur ein beziehungsweise drei Übergabeparameter erwarten, wurde auch die Anforderung **A3** umgesetzt.

8.1.4. A4: Das Prüfergebnis

Der Schwerpunkt dieser Anforderung lag darin, das Ergebnis der Gültigkeitsüberprüfung so zu konzipieren, dass es genügend Information enthält, um weitere Analysen durchführen zu können. Für die Umsetzung der Anforderung **A4** wurde eine eigene Datenstruktur

[9] SearchResult = Das Ergebnis der ifmapj *„search“* Funktion.

CheckResult erstellt. Dies enthält Informationen, ob die Überprüfung gemäß den Spezifikationen gültig war, welche Abweichungen es gibt, und welche IF-MAP Metadaten von den Spezifikationen abweichen.

8.2. Nichtfunktionale Anforderungen

8.2.1. A5: Plattformunabhängigkeit

Die Anforderung **A5** bestand darin, die Gültigkeitsüberprüfung plattformunabhängig zu konzipieren um die Überprüfung des IF-MAP Graphen nach Möglichkeit unabhängig von dem Betriebssystem durchzuführen. Für die Realisierung der Gültigkeitsüberprüfung wurde die Programmiersprache Java verwendet, was momentan die Ausführung der Software auf vielen Betriebssystemen ermöglicht, ohne diese an das Betriebssystem anpassen zu müssen.

8.3. Zusammenfassung

Es wurden fünf der sechs Anforderungen in der vorgestellten Lösung umgesetzt. Bei der Implementierung wurde auf Effizienz und Ressourcenschonung geachtet. Der Nachweis einer erfolgreichen Umsetzung auch der letzten Anforderung A6: Performance wurde als schwache Anforderung in diese Arbeit nicht überprüft.

9. Fazit und Ausblick

9.1. Fazit

Im unserem Leben spielt der Datenaustausch im Netzwerk eine immer größere Rolle. Gerade im Hinblick auf die zunehmende Vernetzung und der steigenden Zahl an Gefahren durch Schadsoftware steigt der Bedarf an vertrauenswürdigen Netzwerkverbindungen. Trusted Network Connect (TNC) ist ein Lösungsansatz um Netzwerksicherheit zu gewährleisten. In der Netzwerksicherheit wird der Einsatz des IF-MAP Protokoll häufiger und es ist von großer Bedeutung, dass die verwendete IF-MAP Komponenten spezifikationskonform laufen, um die Stabilität des Netzes nicht zu gefährden. Die in dieser Arbeit entwickelte Lösung „Gültigkeitsüberprüfung des IF-MAP Graphen“ ist ein Schritt in diese Richtung.

9.2. Ausblick

Aus Zeitmangel konnten in der exemplarischen Implementierung nicht alle Spezifikationsvorgaben umgesetzt werden. Bei der Überprüfung der „Metadaten-spezifischen-Attributen“ wird momentan nur das Vorhandensein der Attribute geprüft.

Der Prüfalgorithmus kann erweitert werden, damit auch die Werte dieser Attribute auf Konformität, genauso wie auf Plausibilität, überprüft werden können. Ebenso kann die Gültigkeitsüberprüfung mit neuen Komfortfunktionen erweitert werden, um den Einsatz dieser Lösung weiter zu optimieren.

Geplant war zum Beispiel die Kommunikationsschnittstelle um eine Methode zu erweitern, die zusätzlich als Übergabeparameter eine Verbindung zu dem Server synchronous send-receive channel **(SSRC)** bekommt, um darüber das Prüfergebnis auf einem IF-MAP Server veröffentlichen zu können.

Literatur

[1] TNC IF-MAP Metadata for Network Security Version 1.1 (Revision 8)
`http://www.trustedcomputinggroup.org/files/resource_files/281C050E-1A4B-B294-D018131B07BC2030/TNC_IFMAP_Metadata_For_Network_Security_v1_1r8.pdf`
[Abrufdatum 12.07.2013]

[2] TNC IF-MAP Binding for SOAP, Version 2.1 (Revision 15)
`http://www.trustedcomputinggroup.org/files/resource_files/2888CAD9-1A4B-B294-D0ED95712B121FEF/TNC_IFMAP_v2_1r15.pdf`
[Abrufdatum 12.07.2013]

[3] TNC IF-MAP Metadata for ICS Security,Version 1.0 (Reversion 39)
`https://www.trustedcomputinggroup.org/files/static_page_files/25FA4147-1A4B-B294-D0052022B90F2B06/IFMAP_Metadata_For_ICS_Security_v1_0r39.pdf`
[Abrufdatum 08.08.2013]

[4] Webseite der Trusted Computing Group
`http://www.trustedcomputinggroup.org/` [Abrufdatum 12.07.2013]

[5] Webseite der Trust@FHH Group
`http://trust.f4.hs-hannover.de/` [Abrufdatum 12.07.2013]

[6] Download Bereich der Trust@FHH Group
`http://trust.f4.hs-hannover.de/downloads/` [Abrufdatum 12.07.2013]

[7] Webseite des W3C (SOAP)
`http://www.w3.org/TR/soap/` [Abrufdatum 24.07.2013]

[8] Webseite des W3C (XML)
`http://www.w3.org/XML/` [Abrufdatum 24.07.2013]

[9] Webseite des W3C (XML-Schema)
`http://www.w3.org/XML/Schema` [Abrufdatum 24.07.2013]

[10] Trusted Computing: Ein Weg zu neuen IT-Sicherheitsarchitekturen von Norbert Pohlmann und Helmut Reimer erschienen bei Friedr. Vieweg & Sohn Verlag im Jahr 2008

[11] Prof. Dr. Ralf Bruns. Software engineering 2. Fachhochschule Hannover, Ricklinger Stadtweg 120, 30459 Hannover, Wintersemester 2012/2013. Vorlesungsskript.

[12] Prof. Dr. Arne Koschel u. Prof. Dr. Jürgen Dunkel. Software engineering 3. Fachhochschule Hannover, Ricklinger Stadtweg 120, 30459 Hannover, Sommersemester 2013. Vorlesungsskript.

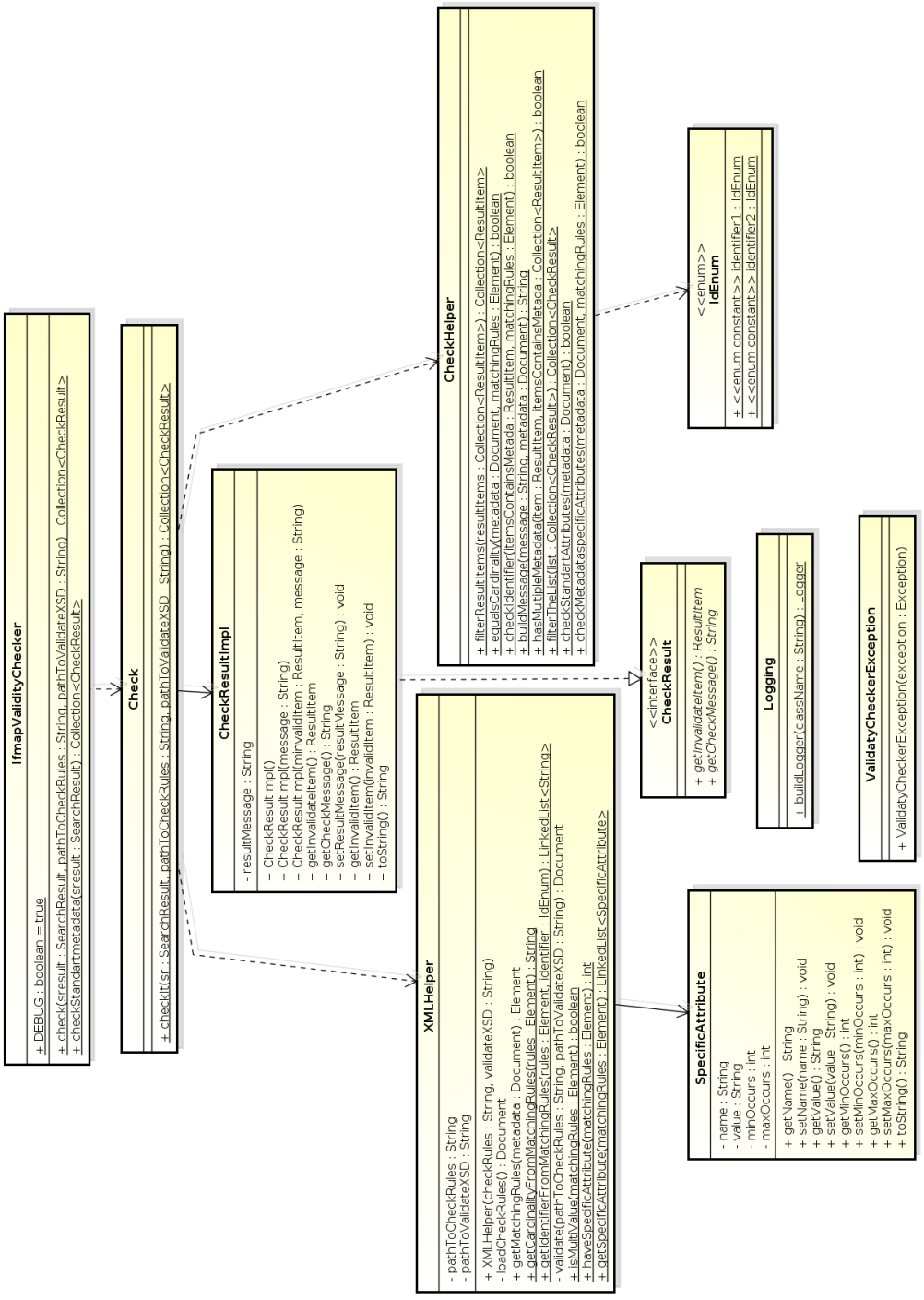
[13] Tobias Ruhe. Visualisierung von Informationen einer zentralen Netzwerk-Datenbank, 2010. Fachhochschule Hannover, Ricklinger Stadtweg 118,30459 Hannover. Bachelorarbeit

[14] Ralf Steuerwald, Anbindung von Open Vulnerability Assessment System (OpenVAS) an eine Metadata Access Point (MAP)-Infrastruktur, 2011, Fachhochschule Hannover, Ricklinger Stadtweg 118,30459 Hannover. Bachelorarbeit

[15] Joram Knaack, Analyse und Implementierung des IF-MAP 2.1 Protokolls, 2012, Fachhochschule Hannover, Ricklinger Stadtweg 118, 30459 Hannover. Bachelorarbeit

A. Anhang

A.1. Klassendiagramm



A.2. CheckRules.xml

Listing 8: CheckRules.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ifmap:IfmapValidityCheck
3     xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/
          IFMAP-METADATA/2"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://www.trustedcomputinggroup.org
          /2010/IFMAP-METADATA/2 StandardMetadata.xsd ">
6
7       <!--   !!!!!!! Template !!!!!!!!!!!!!
8     <metadata name="">
9          <ifmap:ifmap-cardinality name="singleValue/multiValue"/
               >
10         <ifmap:identifier1></ifmap:identifier1>
11         <ifmap:identifier2></ifmap:identifier2>
12         <ifmap:message></ifmap:message>
13    </metadata>
14 -->
15
16    <metadata name="access-request-device">
17       <ifmap:ifmap-cardinality name="singleValue"/>
18       <ifmap:identifier1>ar</ifmap:identifier1>
19       <ifmap:identifier2>dev</ifmap:identifier2>
20       <ifmap:message>Clients MUST publish this singleValue
             metadata only between: access-request and device</
             ifmap:message>
21    </metadata>
22
23    <metadata name="access-request-ip">
24       <ifmap:ifmap-cardinality name="singleValue"/>
25       <ifmap:identifier1>ar</ifmap:identifier1>
26       <ifmap:identifier2>ip</ifmap:identifier2>
27       <ifmap:message>Clients MUST publish this singleValue
             metadata only between: access-request and ip-address</
             ifmap:message>
28    </metadata>
29
30    <metadata name="access-request-mac">
31       <ifmap:ifmap-cardinality name="singleValue"/>
```

```
32      <ifmap:identifier1>ar</ifmap:identifier1>
33      <ifmap:identifier2>mac</ifmap:identifier2>
34      <ifmap:message>Clients MUST publish this singleValue
            metadata only between: access-request and mac-address<
            /ifmap:message>
35    </metadata>
36
37    <metadata name="authenticated-as">
38      <ifmap:ifmap-cardinality name="singleValue"/>
39      <ifmap:identifier1>ar</ifmap:identifier1>
40      <ifmap:identifier2>id</ifmap:identifier2>
41      <ifmap:message>Clients MUST publish this singleValue
            metadata only between: access-request and non-extended
             identity</ifmap:message>
42    </metadata>
43
44    <metadata name="authenticated-by">
45      <ifmap:ifmap-cardinality name="singleValue"/>
46      <ifmap:identifier1>ar</ifmap:identifier1>
47      <ifmap:identifier2>dev</ifmap:identifier2>
48      <ifmap:message>Clients MUST publish this singleValue
            metadata only between: access-request and device</
            ifmap:message>
49    </metadata>
50
51    <metadata name="capability">
52      <ifmap:ifmap-cardinality name="multiValue"/>
53      <ifmap:specific-attribute name="name" value="" min="1"
            max="1" />
54      <ifmap:identifier1>ar</ifmap:identifier1>
55      <ifmap:message>Recommended publish this multiValue
            metadata for: access-request</ifmap:message>
56    </metadata>
57
58    <metadata name="device-attribute">
59      <ifmap:ifmap-cardinality name="multiValue"/>
60      <ifmap:specific-attribute name="name" value="" min="1"
            max="1" />
61      <ifmap:identifier1>ar</ifmap:identifier1>
62      <ifmap:identifier2>dev</ifmap:identifier2>
63      <ifmap:message>Recommended publish this multiValue
            metadata for: access-request and device</ifmap:message
```

```
        >
64  </metadata>
65
66  <metadata name="device-characteristic">
67      <ifmap:ifmap-cardinality name="multiValue"/>
68      <ifmap:specific-attribute name="discovered-time" value=
            "" min="1" max="1" />
69      <ifmap:specific-attribute name="discoverer-id" value=""
            min="1" max="1" />
70      <ifmap:specific-attribute name="discovery-method" value
            ="" min="1" max="1" />
71      <ifmap:identifier1>ar</ifmap:identifier1>
72      <ifmap:identifier1>ip</ifmap:identifier1>
73      <ifmap:identifier1>mac</ifmap:identifier1>
74      <ifmap:identifier2>dev</ifmap:identifier2>
75      <ifmap:message>Recommended publish this multiValue
            metadata for: access-request and device,
76          ip-address and device, or mac-address and device</
                ifmap:message>
77  </metadata>
78
79  <metadata name="device-ip">
80      <ifmap:ifmap-cardinality name="singleValue"/>
81      <ifmap:identifier1>dev</ifmap:identifier1>
82      <ifmap:identifier2>ip</ifmap:identifier2>
83      <ifmap:message>Clients MUST publish this singleValue
            metadata only between: device and ip-address</
            ifmap:message>
84  </metadata>
85
86  <metadata name="discovered-by">
87      <ifmap:ifmap-cardinality name="singleValue"/>
88      <ifmap:identifier1>ip</ifmap:identifier1>
89      <ifmap:identifier1>mac</ifmap:identifier1>
90      <ifmap:identifier2>dev</ifmap:identifier2>
91      <ifmap:message>Clients MUST publish this singleValue
            metadata only between: ip-address and device or
92          mac-address and device</ifmap:message>
93  </metadata>
94
95  <metadata name="enforcement-report">
96      <ifmap:ifmap-cardinality name="multiValue"/>
```

```
97      <ifmap:specific-attribute name="enforcement-action"
            value="" min="1" max="1" />
98      <ifmap:identifier1>ip</ifmap:identifier1>
99      <ifmap:identifier1>mac</ifmap:identifier1>
100     <ifmap:identifier2>dev</ifmap:identifier2>
101     <ifmap:message>Clients MUST publish this multiValue
            metadata only between: ip-address and device or
102           mac-address and device</ifmap:message>
103 </metadata>
104
105 <metadata name="event">
106     <ifmap:ifmap-cardinality name="multiValue"/>
107     <ifmap:specific-attribute name="name" value="" min="1"
            max="1" />
108     <ifmap:specific-attribute name="discovered-time" value=
            "" min="1" max="1" />
109     <ifmap:specific-attribute name="discoverer-id" value=""
             min="1" max="1" />
110     <ifmap:specific-attribute name="magnitude" value="" min
            ="1" max="1" />
111     <ifmap:specific-attribute name="confidence" value=""
            min="1" max="1" />
112     <ifmap:specific-attribute name="significance" value=""
            min="1" max="1" />
113     <ifmap:identifier1>ar</ifmap:identifier1>
114   <ifmap:identifier1>id</ifmap:identifier1>
115   <ifmap:identifier1>ip</ifmap:identifier1>
116   <ifmap:identifier1>mac</ifmap:identifier1>
117     <ifmap:message>Recommended publish this multiValue
            metadata for: access-request,
118           identity, ip-address or mac-address</ifmap:message>
119 </metadata>
120
121 <metadata name="ip-mac">
122     <ifmap:ifmap-cardinality name="multiValue"/>
123     <ifmap:identifier1>ip</ifmap:identifier1>
124   <ifmap:identifier2>mac</ifmap:identifier2>
125     <ifmap:message>Recommended publish this multiValue
            metadata for: ip-address and mac-address</
            ifmap:message>
126 </metadata>
127
```

```
128   <metadata name="layer2-information">
129       <ifmap:ifmap-cardinality name="multiValue"/>
130       <ifmap:identifier1>ar</ifmap:identifier1>
131     <ifmap:identifier2>dev</ifmap:identifier2>
132       <ifmap:message>Recommended publish this multiValue
              metadata for: access-request and device</
              ifmap:message>
133   </metadata>
134
135   <metadata name="location">
136       <ifmap:ifmap-cardinality name="multiValue"/>
137       <ifmap:specific-attribute name="location-information"
              value="" min="1" max="100" />
138       <ifmap:specific-attribute name="discovered-time" value=
              "" min="1" max="1" />
139       <ifmap:specific-attribute name="discoverer-id" value=""
               min="1" max="1" />
140       <ifmap:identifier1>id</ifmap:identifier1>
141     <ifmap:identifier1>ip</ifmap:identifier1>
142     <ifmap:identifier1>mac</ifmap:identifier1>
143       <ifmap:message>Recommended publish this multiValue
              metadata for: identity, ip-address or mac-address</
              ifmap:message>
144   </metadata>
145
146   <metadata name="request-for-investigation">
147       <ifmap:ifmap-cardinality name="multiValue"/>
148       <ifmap:identifier1>ip</ifmap:identifier1>
149       <ifmap:identifier1>mac</ifmap:identifier1>
150       <ifmap:identifier2>dev</ifmap:identifier2>
151       <ifmap:message>Recommended publish this multiValue
              metadata for: device and mac-address or device and
              ip-address</ifmap:message>
152   </metadata>
153
154   <metadata name="role">
155       <ifmap:ifmap-cardinality name="multiValue"/>
156       <ifmap:specific-attribute name="name" value="" min="1"
              max="1" />
157       <ifmap:identifier1>ar</ifmap:identifier1>
158       <ifmap:identifier2>id</ifmap:identifier2>
```

```
159         <ifmap:message>Recommended publish this multiValue
                metadata for: access-request and identity</
                ifmap:message>
160     </metadata>
161
162     <metadata name="unexpected-behavior">
163         <ifmap:ifmap-cardinality name="multiValue"/>
164         <ifmap:specific-attribute name="discovered-time" value=
                "" min="1" max="1" />
165         <ifmap:specific-attribute name="discoverer-id" value=""
                 min="1" max="1" />
166         <ifmap:specific-attribute name="magnitude" value="" min
                ="1" max="1" />
167         <ifmap:specific-attribute name="significance" value=""
                min="1" max="1" />
168         <ifmap:identifier1>ar</ifmap:identifier1>
169       <ifmap:identifier1>id</ifmap:identifier1>
170       <ifmap:identifier1>ip</ifmap:identifier1>
171       <ifmap:identifier1>mac</ifmap:identifier1>
172         <ifmap:message>Recommended publish this multiValue
                metadata for: access-request, identity,
173           ip-address or mac-address</ifmap:message>
174     </metadata>
175
176     <metadata name="wlan-information">
177         <ifmap:ifmap-cardinality name="singleValue"/>
178         <ifmap:specific-attribute name="ssid-unicast-security"
                value="" min="1" max="100" />
179         <ifmap:specific-attribute name="ssid-group-security"
                value="" min="1" max="1" />
180         <ifmap:specific-attribute name="ssid-management-
                security" value="" min="1" max="100" />
181         <ifmap:identifier1>ar</ifmap:identifier1>
182         <ifmap:identifier2>dev</ifmap:identifier2>
183         <ifmap:message>Recommended publish this singleValue
                metadata for: access-request and device</
                ifmap:message>
184     </metadata>
185
186     <!-- Vendor Specific metadata-->
187     <metadata name="vendormetadata1">
188         <ifmap:ifmap-cardinality name="multiValue"/>
```

```
189         <ifmap:identifier1>ar</ifmap:identifier1>
190         <ifmap:message>!!!!!!!!1 Recommended publish this Vendor
                Specific metadata for: access-request only!!!!!!!!!
                </ifmap:message>
191     </metadata>
192 </ifmap:IfmapValidityCheck>
```

A.3. CheckRulesSchema.xsd

Listing 9: CheckRulesSchema.xsd

```
1  <?xml version="1.0" ?>
2  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3    xmlns:ifmap="http://www.trustedcomputinggroup.org/2010/
       IFMAP/2"
4    xmlns="http://www.trustedcomputinggroup.org/2010/IFMAP-
       METADATA/2"
5    targetNamespace="http://www.trustedcomputinggroup.org/2010/
       IFMAP-METADATA/2">
6
7    <!-- access-request-device is link metadata that associates
         an access-request
8       identifier with a device identifier -->
9
10   <xsd:element name="IfmapValidityCheck">
11     <xsd:complexType>
12       <xsd:choice maxOccurs="unbounded">
13         <xsd:element name="metadata">
14           <xsd:complexType>
15             <xsd:sequence>
16               <xsd:element ref="ifmap-cardinality" minOccurs=
                   "1"
17                 maxOccurs="1" />
18               <xsd:element ref="specific-attribute" minOccurs
                   ="0"
19                 maxOccurs="unbounded"/>
20               <xsd:element ref="identifier1" minOccurs="1"
21                 maxOccurs="unbounded" />
22               <xsd:element ref="identifier2" minOccurs="0"
23                 maxOccurs="1" />
24               <xsd:element ref="message" minOccurs="1"
                   maxOccurs="1" />
```

```
25              </xsd:sequence>
26              <xsd:attribute name="name" type="xsd:string" use=
                    "required" />
27            </xsd:complexType>
28          </xsd:element>
29        </xsd:choice>
30      </xsd:complexType>
31    </xsd:element>
32
33    <xsd:element name="ifmap-cardinality">
34      <xsd:complexType>
35        <xsd:attribute name="name" use="required">
36          <xsd:simpleType>
37            <xsd:restriction base="xsd:string">
38              <xsd:enumeration value="singleValue" />
39              <xsd:enumeration value="multiValue" />
40            </xsd:restriction>
41          </xsd:simpleType>
42        </xsd:attribute>
43      </xsd:complexType>
44    </xsd:element>
45
46    <xsd:element name="specific-attribute">
47      <xsd:complexType>
48        <xsd:attribute name="name" use="required" type="
              xsd:string"/>
49        <xsd:attribute name="value" use="optional" type="
              xsd:string"/>
50        <xsd:attribute name="min" use="required" type="
              xsd:nonNegativeInteger"/>
51        <xsd:attribute name="max" use="required" type="
              xsd:nonNegativeInteger"/>
52      </xsd:complexType>
53    </xsd:element>
54
55    <xsd:element name="identifier1" type="xsd:string" />
56    <xsd:element name="identifier2" type="xsd:string" />
57    <xsd:element name="message" type="xsd:string" />
58
59  </xsd:schema>
```

A.4. CD-ROM

auf der CD-ROM befinden sich:

1. Quellcode der Implementierung
2. Javadoc zu dem Quellcode
3. CheckRules.xml und dazugehörige CheckRulesSchema.xsd
4. Die kompilierte Java Bibliothek „IfmapValidityChecker“ als JAR-Datei
5. Diese Bachelorarbeit als PDF-Datei