



SWAD-Europe: WP11.1: Using RDF for Home Network Configuration

Project name:

Semantic Web Advanced Development for Europe (SWAD-Europe)

Project Number:

IST-2001-34732

Workpackage name:

WP 11: Distributed Trust Systems

Workpackage description:

<http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-11.html>

Deliverable title:

Framework for Security and Trust Standards

URI:

Authors:

Graham Klyne, Ninebynine.org.

Brian Matthews, CCLRC.

Abstract:

This memo describes the use of RDF metadata in configuring Internet access from a home network. Its goal is to demonstrate the practical applicability of common semantic web technologies in a simple real-world based scenario. The work has been inspired in part by an architectural proposal for XML network configuration submitted to the IETF.

Status:

First draft: 2002-12-22

Deliverable release: 2004-06-15

Comments on this document should be sent to the public SWAD-Europe mailing list, public-esw@w3.org,

-
- ☞ **1. Introduction**
 - ☞ **1.1 Structure of this memo**
 - ☞ **2. Background**
 - ☞ **3. Vocabulary for network user access policies**
 - ☞ **3.1 A note about access permission logic**
 - ☞ **4. Vocabulary for network address configuration**
 - ☞ **5. Vocabulary for network device configuration**
 - ☞ **6. Rules for creating configuration data**
 - ☞ **7. Templates for creating configuration files**
 - ☞ **8. Conclusions**
 - ☞ **8.1 Some lessons learned**
 - ☞ **8.2 Ideas for better inference capabilities**
 - ☞ **8.3 Further work**
 - ☞ **9. Acknowledgements**
 - ☞ **§ References**
 - ☞ **§ Author's Address**
 - ☞ **A. Revision history**
 - ☞ **B. Unresolved issues**

1. Introduction - This memo describes the use of RDF metadata in configuring [TOC](#) Internet access from a home network. Its goal is to illustrate the practical applicability of common semantic web technologies in a simple real-world based scenario.

The work has been inspired in part by an architectural proposal [\[6\]](#) for XML network configuration submitted to the IETF. It has also been undertaken as an experiment in manipulating access control data represented using RDF, as a continuation of some survey work described in a previous memo, [Framework for Security and Trust Standards\[16\]](#).

1.1 Structure of this memo

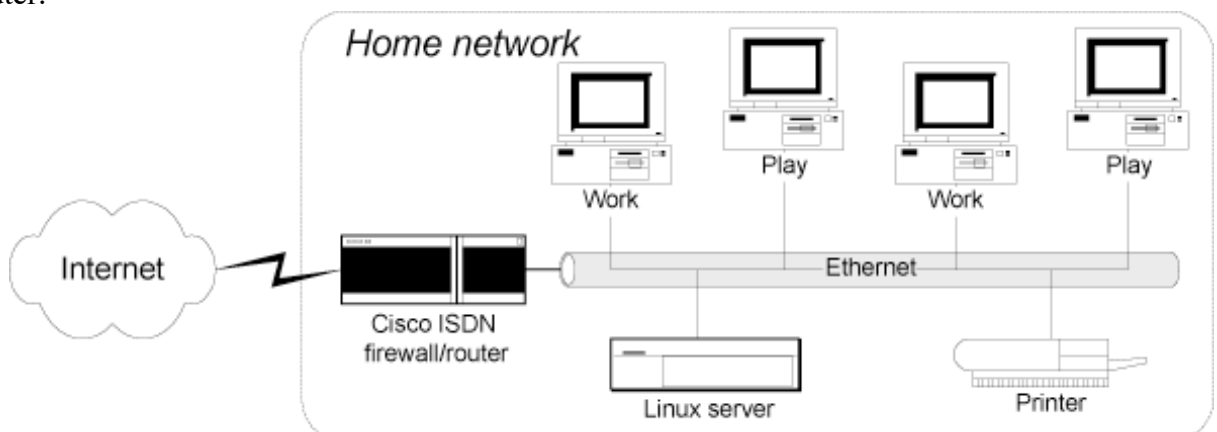
This document is organized as follows:

- Section 2 introduces the scenario addressed by this demonstration.
- Sections 3-5 introduce the RDF vocabulary used to describe the scenario and the results obtained.
- Section 6 discusses the inference process used to create RDF data that closely matches the resulting configuration files.
- Section 7 describes the software used to generate the required configuration data from RDF.
- The remaining sections discuss some implications of this work.

The files used by this demonstration can be found by following the links from <http://www.ninebynine.org/SWAD-E/Intro.html#HomeNetAccessDemo>. In particular, the RDF/N3 (Notation3) source files used are:

- Network and access policy description: [users.n3](#)
- Access policy transformation rules: [configrules.n3](#)
- Configuration files generation template: [configfiles.n3](#)

2. Background - The background scenario for this example use of RDF metadata [TOC](#) is a home business network connected to the Internet by a Cisco dial-on-demand ISDN router.



This illustration is also available in [PDF](#) format.

Network access is provided by an ISP dial-up account that provides unmetered access up to a specified weekly limit.

The network users are two parents who use the Internet for business, and two children who use it for play and social purposes. The working adults require occasional access at any time. The children's access has to be restricted, otherwise they would quickly exceed the total connect time allowed by the ISP.

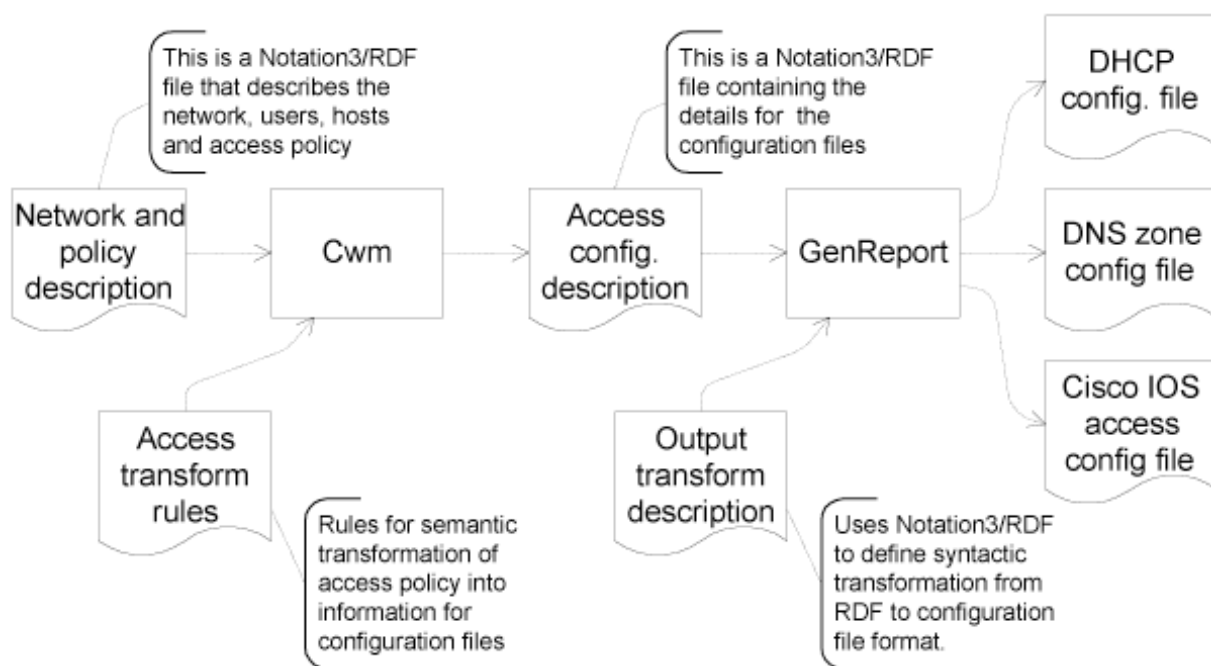
The Cisco ISDN router runs Cisco's proprietary IOS software, which has a system of IP-addressed based filters that are quite capable of restricting access at different times based on the IP address of the internal host. Creating the correct IOS configuration files for this kind of selective filtering is quite a tricky task. The router can accept an externally defined configuration via the TFTP protocol.

Within the local network, machines are assigned IP addresses using DHCP. The DHCP service is provided by a Linux host running DHCPD and a local DNS service.

Given all this, the goal is to use RDF to specify the access policy and device configuration requirements, and use some available RDF tools to generate the desired configuration files for the Cisco ISDN router and the Linux-based services.

The general idea is illustrated thus:

Network access configuration process and files used



This illustration is also available in [PDF format](#).

3. Vocabulary for network user access policies - This section presents the [TOC](#) metadata used to describe the network users and access policies.

Vocabularies used:

rdf:

[RDF core vocabulary\[1\]: essential RDF terms.](#)
 Namespace <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

rdfs:

[RDF schema\[2\]: RDF schema \(vocabulary description\) terms.](#)
 Namespace <http://www.w3.org/2000/01/rdf-schema#>.

foaf:

[friend-of-a-friend\[9\]: for describing basic information about people.](#)
 Namespace <http://xmlns.com/foaf/0.1/>.

ical:

[RDF schema for iCalendar\[10\]: for describing Internet access schedules.](#)
 Namespace
<http://www.ilrt.bris.ac.uk/discovery/2001/06/schemas/ical-full/hybrid.rdf#>.

icalutil:

[iCalendar ancillary terms\[10\]: Some additional vocabulary defining for use in conjunction with some of the ical: terms. There are two](#)

namespace URIs in circulation.

Namespace <http://ilrt.org/discovery/2001/06/schemas/ical-util#> or <http://ilrt.org/discovery/2001/06/schemas/swws/index.rdf#>. The demonstration code described here uses the former value.

user:

a new vocabulary for describing network user characteristics.

Namespace <http://id.ninebynine.org/wip/2002/user/>.

homenet:

a new namespace for components and devices on the local network that is the target of this demonstration.

Namespace <http://id.ninebynine.org/wip/2002/user/>.

Describing network users and access policies:

Class foaf:Person

Describes a network user with the following properties:

foaf:name

Name of user.

foaf:mailbox

Email address of user (mailto: URI).

user:usesHost

A user:HostSystem resource that corresponds to a network host that is commonly used by this person. The user's access rights are applied to this host. This property may be repeated if the person described commonly uses more than one network host.

user:accessType

A user:AccessPolicy resource that represents the person's access rights.

Class user:HostSystem

Describes a network host system with the following properties:

rdfs:label

A brief descriptive label for this machine.

rdfs:comment

A description of this machine.

user:hostname

Local name of machine (prepend to network domain to get FQDN).

user:localNet

A user:LocalNetwork resource that represents the local network, and to which network-wide information is attached.

user:hostIP

IP address of this machine on local network, as "x.x.x.x", where each x is a 0-255 numeral.

user:hostMAC

MAC (Ethernet) hardware address of this machine, as "hh:hh:hh:hh:hh:hh", where each hh is a 2-digit hexadecimal numeral.

user:systemAdmin

A foaf:Person resource that indicates the person who is generally responsible for administration of this system.

user:accessType

A user:AccessPolicy resource that represents the

access rights to be allowed for this host. This may be specified directly, or inferred from properties of any declared user.

user:usedBy

A foaf:Person resource that represents a person who generally uses this host, and whose access permissions should be applied to this host. This may be specified directly, or inferred.

user:usedByName

The name (i.e. foaf:name property) of a person who generally uses this host (cf. user:usedBy).

Class user:LocalNetwork

Describes network-wide characteristics of the local network, with the following properties:

rdfs:label

A brief descriptive label for this network.

rdfs:comment

A description of this network.

user:dhcpHostName

Local name of machine that provides DHCP and other host configuration services for this network (prepend to network domain to get FQDN).

user:netbiosServer

Local name of machine that provides netbios name services for this network (prepend to network domain to get FQDN).

user:networkDomain

The DNS domain name for this network, without leading or trailing '.' characters.

user:networkAddr

The IP subnet address for this network, as "x.x.x.x".

user:networkMask

The IP subnet mask for this network, as "x.x.x.x".

user:broadcastAddr

The IP local net broadcast address for this network, as "x.x.x.x".

user:defaultGateway

A user:HostSystem resource that represents the default gateway (IP router) for the current network.

user:defaultDNS

A list of IP addresses of DNS servers that are available to hosts on the current network.

user:dhcpPoolStart

The first IP address in the pool of addresses allocated by DHCP for hosts on this network, as "x.x.x.x".

user:dhcpPoolEnd

The first IP address in the pool of addresses allocated by DHCP for hosts on this network, as "x.x.x.x".

user:addressPool

The range of IP addresses allocated by DHCP for hosts on this network, as "x.x.x.x y.y.y.y". (This information duplicates user:dhcpPoolStart and user:dhcpPoolEnd, and should be eliminated, but has

been defined here to simplify some internal processing. There is a general issue here that address values are not easily manipulated by general RDF tools like cwm, so additional inputs may be provided as a way to avoid having to define difficult and complex inference rules.)

user:defaultAccess

A user:AccessPolicy resource that represents the access rights to be allowed for hosts that do not otherwise have any access rights defined. (This is not currently used, as it represents a kind of default reasoning that doesn't sit comfortably in RDF.)

Class user:AccessPolicy

Describes an access policy that may be applied to a network user or host. Ultimately it is enforced on a per-host basis. An access policy is described using the following properties:

rdfs:label

Display name for this access policy.

rdfs:comment

A description of this access policy.

user:access

references a user:AccessRule resource describing a network access permission. This property may appear several times to specify multiple access permissions, all of which are granted.

Class user:AccessRule

Describes an access rule that may be applied. The rule controls protocols that may be used (by IP protocol and port number) and the times at which access may be granted. An access rule permits access if the user:accessTimes and user:accessServices properties are satisfied. An access rule is described using the following properties:

user:accessTimes

ical:VCALENDAR resource describing allowed access times. To be specified exactly once.

user:accessServices

references a user:ServiceGroup resource describing the external services that may be accessed. To be specified exactly once.

Class user:ServiceGroup

Describes a group of services that are associated with an access rule:

user:accessProtocol

references a user:ServiceProtocol resources, which describes a collection of permitted services associated with a specified IP protocol. This property may be repeated to specify additional permitted services.

Class user:ServiceProtocol

Describes a collection of services that are associated with a specified IP protocol.

user:serviceProtocol

Specifies an IP protocol number of name ICMP(1), IGMP(2), TCP(6), UDP(17), SCTP(132), etc. May be specified as numeric values or protocol names. Normally, this should be specified exactly once. In some cases, it might make sense to specify more than once; e.g. specify both UDP and TCP for a collection of service protocols that are all made available via both TCP and UDP on the same port numbers.

user:excludePorts

Specifies a list of port numbers to which access is not to be permitted. Access to all ports other than those specified in the list is permitted. Should be specified at most once, and not when user:includePort is used on the same subject.

user:includePorts

Specifies a port number to which access is permitted. May be specified multiple times to permit access to multiple ports. Should not be specified when user:excludePorts is used.

Each occurrence of user:includePorts and user:excludePorts used specifies a port or ports to which access is permitted. All of the ports thus specified are permitted for access. The recommendations given above concerning use of multiple properties are not absolute requirements, but are suggested to avoid unexpected results. E.g., the result of two user:excludePorts properties each specifying a different port number is that access is permitted for all port numbers.

Class ical:VCALENDAR

Describes a schedule of times at which access may be permitted. Most of the properties used are defined by the [RDF](#) schema for iCalendar[10], which itself is derived quite closely from the [iCalendar core object specification](#)[3]. Some additional properties are used:

rdfs:label

A display name for this time range value.

user:rangeName

Specifies a unique alphanumeric name for this time range that can be used when generating configuration data.

In addition to the network, users, hosts and access policy definitions described above, the input data also contains some hints that are used to guide the creation of Cisco IOS configuration files. Ideally, these hints would not be required, and with sufficient effort some logic processing code could be developed to deduce the detailed information provided by the hints. But the object of this exercise was an experiment to generate configuration data with some available tools.

The hints are provided by two additional resources:

user:IOS_Dialout_Rules

provides hints for generating an IOS access list to be used for dial-on-demand Internet connection. Certain protocols are blocked to prevent normal network activity from creating unneeded network connections.

user:IOS_Recv_Local_Rules

provides hints for generating an IOS access list to be used for determining what packets from the local network are to be allowed through the IOS router.

Each of these resources is a list of RDF resources, each of which is used in turn to control the generation of some IOS access list statements. In this list, order is significant. The properties for generating access list statements are:

user:permitLocal

Generates an access list entry that permits all traffic that is local to the current network (i.e. that is both from and to the local network).

user:denyPool

Generates an access list entry that blocks all traffic that originates from the local network DHCP address pool.

user:denyService

where the value is a user:ServiceItem resource. Generates an access list entry that blocks access for the specified IP protocol to any port excluded by the user:ServiceItem value.

user:permitRule

where the value is a user:AccessRule resource. Generates an access list entry that permits access for all network hosts to the indicated service at the indicated times.

user:permitHost

where the value is a user:HostSystem resource. Generates an access list entry that permits access for the specified host at all times.

user:permitUser

where the value is a foaf:Person resource. Generates an access list entry that permits access for all hosts used by the specified user at all times.

3.1 A note about access permission logic

The RDF description of access policies takes the form of access permission, all of which are taken to be granted. This is consistent with the formal semantics of RDF, in which asserting an RDF graph means that all of the statements it contains are true.

A consequence of the RDF formal semantics is that any graph entails any of its subgraphs. That is, if a given RDF graph is presumed to be true, then any other RDF that contains some subset of the statements of the first graph is also true under the same circumstances. So, if we have an RDF graph that asserts some set of permissions, and remove some statements from that graph, the resulting graph must not describe any more permissions than the original.


However, confusion can arise when considering the conditions under which access may be granted. Rather than requiring all of the stated access conditions to hold, in general only some of them are required. This may seem to be logically at odds with the conjunctive semantics of access permissions described above. But determining whether some particular access is to be granted is a different process than describing the available access permissions: technically, it can be regarded as a query against some database of access permissions. A query against a database succeeds when the database contains some information that satisfies the query, and does not require that all such information does so. Thus, there is no conflict.

Finally, it is a conscious design decision for the purposes of continuing experimentation that the logic of access permissions described here is based solely on granting access. Many real-world access control systems (e.g. XACML[5]) use a combination of access permissions and denials. This sometimes makes it easier to define access policy starting from some broad principles which can be successively refined by exception, but also makes the logic of access control more complicated to analyze. The successive refinement approach also sits less comfortably with RDF's strictly monotonic logical semantics. It would be possible to use RDF to define both permissions and denials, but then some additional issues must be considered:

- A given access may evaluate as both permitted and denied. How is this to be

treated? It may be regarded as an inconsistency in the access rules, or some other form of resolution be adopted.

- Some access may evaluate as neither permitted nor denied. How are these cases to be treated?

4. Vocabulary for network address configuration - This vocabulary is used to  TOC generate DNS and DHCP data to be served by the Linux network server.

Vocabularies used:

user:

a new vocabulary for describing network user characteristics.

Namespace  <http://id.ninebynine.org/wip/2002/user/>.

dnsa:

new vocabulary for describing DNS address records.

Namespace  <http://id.ninebynine.org/wip/2002/dnsa/>.

dhcp:

new vocabulary for describing DHCP configuration, to associate known IP addresses with hardware MAC addresses.

Namespace  <http://id.ninebynine.org/wip/2002/dhcp/>.

Describing DNS address records:

Class user:HostSystem

Describes a DNS host address record.

dnsa:hostDomainName

Domain name of host.

dnsa:hostIPAddress

An IP address, or list of IP addresses if the host is multi-homed.

Describing DHCP configuration:

Class user:LocalNetwork

DHCP parameters relating the local network rather than any specific host.

user:networkDomain

(See previous section.)

user:networkAddr

(See previous section.)

user:networkMask

(See previous section.)

user:broadcastAddr

(See previous section.)

user:defaultGateway

(See previous section.)

user:defaultDNS

(See previous section.)

user:addressPool

(See previous section.)

user:netbiosServer

(See previous section.)

dhcp:defLease

Default lease for DHCP-granted addresses.

dhcp:maxLease

Maximum lease for DHCP-granted addresses.

dhcp:dhcpHostFQDN

Fully qualified domain name of DHCP server for this network.

dhcp:updateStyle

Dynamic DNS update style for this network.

Class user:HostSystem

DHCP parameters relating a specific host. These are used to generate fixed IP addresses for hosts identified by their MAC (Ethernet) address. Access rules are then enforced on the IP address.

user:hostName

Local name of the host (see previous section).

dnsm:hostDomainName

Fully qualified domain name of the host.

dhcp:hostMACAddress

A host MAC (Ethernet hardware) address for the given host.

5. Vocabulary for network device configuration - This section introduces a [TOC](#) vocabulary for describing network device configuration. The terms presented here are oriented toward Cisco IOS configuration file commands. Additional terms may be introduced for different network devices.

The original intent of this demonstration was to infer a range of RDF statements that correspond closely to the required output, and then generate the configuration files in a simple syntactic transformation process with minimal logic. In the present implementation, there is less inference and more logic used in the output generation phase than was intended. As a consequence, the network device configuration vocabulary actually used is quite limited. (I still think that the original design approach is preferable, but to realize that, greater inferential power needs to be brought to bear on the problem.)

Vocabularies used:

user:

a new vocabulary for describing network user characteristics.

Namespace <http://id.ninebynine.org/wip/2002/user/>.

ical:

[RDF schema for iCalendar\[10\]](#): for describing Internet access schedules.

Namespace

<http://www.ilrt.bris.ac.uk/discovery/2001/06/schemas/ical-full/hybrid.rdf#>.

ndev:

a new vocabulary for describing network device characteristics.

Describing network devices:

Class ical:VCALENDAR

Specifies a time schedule for an access rule.

user:rangeName

Specifies an alphanumeric name for the time range that can be used as an identifier in generated configuration data.

ical:VEVENT-PROP

Specifies a REC-VEVENT resource that describes a time interval that is included in the schedule specified by the current VCALENDAR value.

Class ical:REC-VEVENT

Specify a recurring time interval.

ndev:dayName

Day(s) of week for which this interval applies. One of: "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "weekday", "weekend".

ndev:timeStart

Time for start of interval, as hh:mm, 24-hour clock.

ndev:timeEnd

Time for end of interval, as hh:mm, 24-hour clock.

Note: in accordance with cwm name-munging algorithms, hyphens contained in identifiers are coded as double underbars ("__") in Notation3 data.

6. Rules for creating configuration data - These rules are evaluated using Tim [TOC](#) Berners-Lee's [cwm](#) program[12]. It may also be possible to evaluate these using Jos De Roo's faster RDF rule engine, [Euler](#)[13].

The rules are intended to take the input network and access policy description and generate additional facts closely corresponding to the network configuration data that is used by the network configuration and access control mechanisms. The idea is that the resulting facts are simply converted to the appropriate format for use by the network systems. (In practice, to get an initial demonstration prepared quickly, this goal is only partially realized.)

The rules are coded in Notation3 in the format recognized by [cwm](#)[12]. Running cwm with the following files as input:

- [users.n3](#) and
- [configrules.n3](#)

yields this output file:

- [genconfig.n3](#)

Roughly, the rules perform the following functions:

- Construct a full domain name for each host.
- Apply some default DHCP parameter values.
- Construct DNS configuration values for each host, combining network-wide and host-specific information.
- [\[\[\[TODO: Process access policy descriptions to a set of statements corresponding closely to the various configuration files to be generated. This will involve reformulating the given access policies in a logical form that maps directly to the configuration file formats.\]\]\]](#)
- Process time interval data to a form that corresponds closely to the required access configuration data format.

7. Templates for creating configuration files - The final creation of configuration [TOC](#) files from RDF configuration data is performed using my Semafor RDF-driven query and

report generation software [14], and in particular the Python program N3GenReport.py. This software is described in RDF for little languages[15].

The output files generation template is coded in Notation3 in a format recognized by the N3GenReport.py program. Running N3GenReport.py with the following files as input:

- configfiles.n3
- genconfig.n3

yields these output files:

- dhcpd.conf
- named-localnet.conf
- ios-accesslists.conf

which are the required configuration data for DHCP, DNS and the IOS firewall router respectively.

8. Conclusions - The results from this exercise have not been exactly as expected, though the deviations from expectations have not been so great as to invalidate the goals. It was expected that this simple access control scenario was well within the capabilities of current RDF tools to process with very little difficulty. In practice, the inference needed to map between the logic of different access control schemes has not been so easy.

The first implemented design [17] suffered a flaw with respect to RDF formal semantics because of some confusion between describing access permissions and determining whether a given access is permitted. This has resulted in addition of the material in section A note about access permission logic.

It has been expected that this project would demonstrate the following features of using RDF in this way:

1. RDF would provide a common metadata format to support heterogeneous devices and systems.
2. Benefits would accrue from leveraging existing ontology and standards work (foaf, calendar).
3. Free extension of existing data schemas would allow specific requirements to be handled.
4. Use of "off-the-shelf" (or "off-the-net") applications would make the demonstration quick and easy to implement.
5. By doing most of the processing in the form of RDF inference, semantic transformations would be separated from syntactic transformations, and the syntactic transformations would be very easy to apply.
6. A common input syntax for all aspects of the process would simplify the creation and acquisition of various kinds of input data.
7. Extension to more demanding scenarios would be easily achieved because of easy extensibility of RDF metadata.

To a fair extent, these expected results have been achieved, but the work proved more difficult than expected because some of the required logic was not easily coded as cwm inference rules. More work is needed to determine if this is due to fundamental limitations of cwm rules, or lack of skill in applying them.

Items 1-3 above were fully realized. RDF's model allowing free extensibility proved to be very valuable, by making it easy to build on existing data definitions.

Items 4-6 were realized to some extent, but the complexity of logic required to get the desired results meant that:

- It may be that "off-the-shelf" (or "off-the-net") software is helpful only so far as the required task is within its capabilities. That RDF is a very general purpose language

for representation of information does not necessarily mean that any RDF inference engine is an equally general-purpose tool for processing that information. Despite this, the demonstration was still constructed quite quickly using existing software, though the result was less functional than expected.

- The separation of semantic and syntactic transformation was reduced by difficulties in achieving the desired semantic transformations. I remain convinced that a sufficiently powerful inference system, capable of dealing with inferences over complex information, would allow this goal to be largely realized. I feel that formal structures for representing disjunction as well as conjunction are needed to support sufficiently powerful inference processes. Such structures can be simulated within a closed environment (such as cwm). Maybe the existing tools will yield better results on further exploration.
- Use of a common syntax certainly has many advantages: it avoids the need to write new parsing software (an activity that XML alone does not completely eliminate), but it does mean that some of the input data must be prepared in a format that is sometimes obscure to read. There has been discussion of tools to map expressions in some defined syntax directly into RDF. My report generation software could well benefit from such a transformation.

With respect to item 7, extensibility of the final result is not as easy as hoped. I think this is due to the cumulative effect of the shortcomings noted above, and that easy extensibility would follow from a cleaner separation between semantic and syntactic transformations.

8.1 Some lessons learned

The following lessons were noted concerning some practical aspects of writing inference rules for this demonstration:

- Rule debugging can be tricky and tedious. It's not always obvious why a rule is not matched. Rules are more compact than many other forms of programming, but not always easier to write. It is questionable if simple rules are indeed the most useful form of expression of some application logic.
- Rule design: use smaller rules, generate intermediate values. I don't know if this is a useful heuristic, but many of the difficulties encountered would have been easier to spot if simpler rules and more of them had been used.
- Could possible use more powerful inference capabilities. This idea is picked up in the next subsection.
- Notation3 makes a very convenient configuration file format. Whatever its faults may be for expressing complex logic, it is very easy to code the information contained in a wide variety of application configuration file formats directly in Notation3.

8.2 Ideas for better inference capabilities

This section explores some ideas for improved inference capabilities, noting that many of the shortcomings noted previously seem to stem from limitations in this area.

In trying to write the inference rules for this demonstration, it was felt that some or all of the following capabilities may have been useful:

- Ability to decompose and get at parts of string expressions. (e.g. decomposition of IP address literals.)
- Logical functions on numbers. (e.g. to allow calculation of IP local net broadcast address.)
- Capability to generate new unique symbols, as required. (For example, when generating Cisco IOS access-lists, arbitrary numbers are used to reference them.)
- List decomposition, honouring RDF semantics. Currently, to get at the members of

a list I have tried flattening the list, which loses some of the information encoded there. (Maybe adding sentinel values to the list elements would be more effective here?)

- List comprehension or building mechanisms. (e.g. is it possible to do an "obvious" task like constructing the reverse of a list using cwm inference rules?)
- The above discussion of list decomposition and building functions is motivated in part by a desire to infer new facts from some ordered information in a way that preserves the original ordering in the new facts.
- Some kinds of rule seem to be repeated for different types of information with similar structure. Is there value in having a rule-schema (or parameterized rule) mechanism?
- Mapping between different logical frameworks. (E.g. is it possible using simple inference rules to map a conjunctive normal form to disjunctive normal form, and vice versa? I see this as a step on the path to optimizing logical expressions, so that given some expression with a given logical outcome to recreate that expression using a minimum number of some different set of primitives. This seems to be at the heart of many of the problems of access rule transformation.)

It may be that all of these facilities can be simulated using simple cwm style rules. If so, it's not obvious how. Also, there's a question of efficiency: even for rules that can be expressed in cwm, they may be slow to execute, and general performance benefits may accrue from having them available as efficiently implemented built-in functions.

8.3 Further work

With respect to the target application of this demonstration, many areas for possible further work are suggested:

- Change the access control descriptions and rules to use the new iCalendar-based schema proposed by the SWAD-E calendaring workshop [11]. A seed version of this new schema is at <http://www.w3.org/2000/10/swap/pim/ical.n3> or <http://www.w3.org/2000/10/swap/pim/ical.rdf>.
- Fix up some problems with the IOS configuration file format.
- Extend the configuration data to create a full IOS configuration file, rather than just the access control elements.
- Automatically generate a SOA sequence number for DNS configuration data.
- Enhance the rules to handle multi-homed hosts.
- Pick up other DNS parameters from RDF input data. (Notation3 makes a very convenient file format for arbitrary configuration data.)
- Consider how to take advantage of web-based knowledge. For example, is there an easy way to take advantage of information published by device manufacturers? Rules, coded in RDF, for transformation from a common configuration database might be an example of this.
- Common abstraction for all network devices. The current work concentrates on mapping a given access policy to device-dependent information using device-specific rules. Is it possible to define a common information format (possibly with redundant information) that is directly transformable to a range of different devices. Then, just one set of transformation rules would serve for all devices. (By different devices, for the purposes of the immediate demonstration I mean devices with different access control models.)
- Generation of IPChains firewall configuration files. The work to map access control policy to access rules might be usefully applied to reliable configuration of freely available firewall software, such as IPChains on Linux. Configuring firewall software tends to be difficult and complicated: could semantic web tools simplify this process?
- User interface. The current work is completely lacking any form of friendly user interface, depending as it does on hand-edited text files for its input.

- Extension to other kinds of device.
- Automatic resource allocation. In its present form, the rules require that resource allocation (e.g. IP addresses) is defined in the input policy description. Could the work be generalized to allow some resources to be allocated automatically from some pool. A further step would be dynamic allocation, but that would require a run-time version of the RDF rule processing software.

9. Acknowledgements - This work has been conducted with support from the [TOC SWAD-E](#) project[7], as a strand of participation by [Rutherford Appleton Laboratory](#)[8].

Central to this work has been Tim Berners-Lee's program [Cwm](#)[12], which has been used to perform all the RDF inference.

This document has been authored in XML using the format described in RFC 2629 [\[4\]](#), and converted to HTML using the XML2RFC utility developed by Marshall Rose (<http://xml.resource.org/>).

- [1] [Lassila, O.](#) and [R. Swick](#), "[Resource Description Framework \(RDF\) Model and Syntax Specification](#)", W3C Recommendation [rdf-syntax](#), February 1999.
 - [2] [Brickley, D.](#) and [R. Guha](#), "[Resource Description Framework \(RDF\) Schema Specification 1.0](#)", W3C Candidate Recommendation [CR-rdf-schema](#), March 2000.
 - [3] [Dawson, F.](#) and [Stenerson, D.](#), "[Internet Calendaring and Scheduling Core Object Specification \(iCalendar\)](#)", RFC 2445, November 1998 ([TXT](#), [HTML](#), [XML](#)).
 - [4] [Rose, M.](#), "[Writing I-Ds and RFCs using XML](#)", RFC 2629, June 1999.
 - [5] [Godik, S.](#) and [T. Moses](#), "[OASIS eXtensible Access Control Markup Language \(XACML\)](#)", OASIS Committee Specification [cs-xacml-specification-1.0](#), November 2002.
 - [6] [Atarashi, R.](#), [Shimojo, T.](#), [Atarashi, Y.](#), [Miyake, S.](#), [Kitani, M.](#), [Baker, F.](#) and [M. Wasserman](#), "[XML Configuration Architecture](#)", [draft-atarashi-xmlconf-architecture-00](#) (work in progress), October 2002.
 - [7] [Brickley, D.](#) and [K. Sharp](#), "[European Semantic Web Advanced Development](#)", 2002.
 - [8] [Matthews, B.](#), "[Rutherford Appleton Laboratory](#)", June 2002.
 - [9] [Brickley, D.](#) and [L. Miller](#), "[Friend-of-a-friend \(FOAF\) RDF vocabulary](#)", 2002.
 - [10] [Miller, L.](#) and [M. Arick](#), "[iCalendar \(ICAL\) RDF vocabulary](#)", 2002.
 - [11] [Miller, L.](#), "[SWAD-Europe: Developer Workshop Report 2 - Semantic Web calendaring](#)", 2002.
 - [12] [Berners-Lee, T.](#), "[Cwm \(closed world machine\)](#)", September 2002.
 - [13] [De Roo, J.](#), "[Euler proof mechanism](#)", December 2002.
 - [14] [Klyne, G.](#), "[Notation3 query processor and report generator software](#)", December 2002.
 - [15] [Klyne, G.](#), "[RDF for little languages](#)", December 2002.
 - [16] [Klyne, G.](#), "[Framework for Security and Trust Standards](#)", December 2002.
 - [17] [Klyne, G.](#), "[Using RDF for Home Network Configuration \(flawed version of 15-Dec-2002\)](#)", December 2002.
-

2002-12-03:

- *Document initially created.*

2002-12-04:

- *Initial sketch of RDF vocabularies to use.*

2002-12-12:

- *Revise vocabulary descriptions based on running code.*
- *Add descriptions of rules and output file generation template.*
- *Add initial draft of conclusions and lessons learned.*

2002-12-15:

- *Noted logical problem with vocabulary definition. This document revision will be preserved as:*
☞ <http://www.ninebynine.org/SWAD-E/Scenario-HomeNetwork/HomeNetworkConfig-20021215.html>.

2002-12-16:

- *Reworked vocabulary to be consistent with RDF formal semantics, in particular the subgraph inference rule.*
- *Add a note about logic of access control descriptions.*
- *Make note to adopt new iCalendar-based RDF schema proposed by the SWAD-E calendaring workshop.*

2002-12-18:

- *Added CCLRC copyright notice.*

2002-12-22:

- *Spell-check content.*
- *Add diagram of process and files used.*