EPP Registry/Registrar Toolkit Installation mini-Howto

조병철

jokun (at) mail.com

이 문서는 EPP에 대한 기본적인 설명과 툴킷의 설치 등에 대한 내용을 담고 있습니다.

고친 과정

고침 1.1.0 2002-05-14 고친이 jokun

오타 수정 및 내용정리

고침 1.0.1 2002-05-11 고친이 jokun

EPP에 대한 개요와 활용부분 추가

고침 1.0 2002-04-29 고친이 jokun

최초 작성

1. 머리글

이 문서는 EPP 레지스트리(Registry)와 통신을 위한 epp-rtk를 설치하는 법을 안내하는 문서입니다. 참고로 EPP(Extensible Provisioning Protocol)는 현재 .name, .info, .pro, .biz 의 domain name 레지스트리에서 사용하고 있습니다. epp-rtk는 설치를 경험해 본 바에 따르면 OS와 gcc, glibc 버젼에 민감한 것 으로 여겨지며, 실제로 레드햇 7.1에서는 설치에 어려움을 겪었으나 레드햇 7.2에 서는 쉽게(?) 진행되었습니다.

1.1. 저작권 정보

1.2. 책임의 한계

본 저자는 문서의 내용이 야기할 수 있는 어떠한 결과에 대해서도 책임을 지지 않습니다. 본 문서에서 내포하고 있는 정보들 및 예제들은 여러분이 알아서 활용하십시오. 비록 최선을 다 했으나 이 문서는 틀린 점이나 오류가 있을 수도 있습니다. 만약 여러분이 틀린 점을 발견했다면 꼭 저에게 알려 주시기 바랍니다.

1.3. 피드백

이 문서에 대한 발전적인 제안이나 수정사항, 문제점 등에 대한 피드백은 언제든지 환영합니다. `<jokun (at) mail.com>`로 메일을 보내 주십시오.

1.4. 도움을 주신 분들

이 문서의 초안에 대해 상세한 지적을 해 주신 권순선(kldp.org 관리자) 님께 감사의 말씀을 올립니다.

2. EPP란?

EPP는 레지스트리(Registry)와 레지스트라(Registrar) 간의 통신을 위한 프로토콜입니다. 레지스트라라 함은 도메인에 있어서 주 등록기관인 레지스트리로 부터 도메인 등록에 대한 위임을 받음 협력업체를 말합니다. 레지스트라가 되기 위해서는 레지스트리로 부터 몇 가지 테스트 과정을 거치며 흔히 얘기 하는 OT&E 테스트가 이에 해당합니다. 이를 통과할 경우 레지스트라 는 레지스트리와의 통신 프로토콜을 통해 실시간으로 도메인 등록이 이루어집 니다. 현재 .com .net .org의 경우 베리사인이 레지스트리에 해당하며 전국에 많은 레지스트라가 존재합니다. 국내에도 약 10개의 업체가 베리사인의 레지 스트라로 등록되어 있습니다.

.com .net .org 뿐만 아니라 TLD(Top Level Domain)는 그 수가 매우 많습니다. 각 TLD에 따라 각 기 다른 최상위 등록기관인 레지스트리가 있으며 해당하는 등록 프로토콜 또한 상이한 것이 현실입니다.

EPP의 근본 동기는 이렇게 많은 레지스트리와 레지스트라 사이의 도메인 등록에 있어서 그 표준을 제시하는 데 있습니다. 매번 새로운 레지스트리가 생길때마다 중복적인 프로토콜을 만드는 데 드는 노력을 줄이고자 하는 것입니다. 실제로 최근에 신규로 오픈하는 도메인인 .biz, .info, .pro, .name 등은 모두 EPP를 채택하고 있으며 신규로 레지스트라가 되고자 하는 곳에서는 EPP-RTK를 사용하여 애플리케이션을 만들고 있습니다. 대표적인 TLD인 com, net, org의 최상위 등록기관인 베리사인에서도 EPP를 개발해 오고 있습니다.

2.1. EPP 개요

EPP는 오브젝트와 액션의 2가지 중요한 특징을 가집니다. domains, contacts, hosts (nameservers) 와 같은 오브젝트는 EPP 레지스트리에 저장되는 객체이며 check, create, update, delete 의 액션은 레지스트리와 오브젝트를 변하게 합니다. EPP 레지스트리와 레지스트라는 상호간에 XML 포맷의 메세지를 통해 커뮤티케이션을 합니다.

EPP는 통신 레이어에 독립적이며 EPP 메세지를 다양한 전송 계층(plain TCP, TCP+TLS, SMTP, BEEP) 위에서 사용할 수 있습니다. EPP는 확장가능한 프로토콜 이기 때문에 필요에 따라 오브젝트 타입과 액션을 추가적으로 정의하여 기본 메시지 이외에 부가적인 기능을 확장하여 사용하기 용이합니다.

EPP는 세션을 유지하는 경우와 그렇지 않은 경우를 모두 사용 가능합니다. 세션을 유지하고자 할 때에는 레지스트라는 한 번의 로그인을 거친 후 epp_Session을 통해 해당 세션을 계속 유지하며 적절한 액션을 취할 수 있습니다. 세션을 유지하는 경우는 매 액션마다 인증과정이 거쳐야 할 필요가 없으므로 세션을 유지하지 않는 경우보다 부하가 더 적습니다.

세션을 유지하지 않을 때에는 아래와 같이 로그인 정보를 가지는 creds를 EPP command를 사용할 때 마다 명시해 주어야 합니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?> <epp
xmlns="urn:iana:xml:ns:epp-1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:iana:xml:ns:epp-1.0 epp-1.0.xsd">
<command> <creds> <clID>ClientX</clID>
<pw>foo-BAR2</pw> <newPW>bar-FOO2</newPW>
<options> <version>1.0</version>
<lang>en</lang> </options> </creds> <login>
<svcs> <domain:svc xmlns:domain="urn:iana:xml:ns:domain"
xsi:schemaLocation="urn:iana:xml:ns:domain domain.xsd"/>
<contact:svc xmlns:contact="urn:iana:xml:ns:contact"
xsi:schemaLocation="urn:iana:xml:ns:contact contact.xsd"/>
<host:svc xmlns:host="urn:iana:xml:ns:host"
xsi:schemaLocation="urn:iana:xml:ns:host host.xsd"/> </svcs>
</login> <unspec/> <clTRID>ABC-12345</clTRID>
</command> </epp>
```

2.2. EPP 오브젝트와 액션

EPP-RTK는 기본적으로 아래와 같은 3개의 오브젝트와 7개의 액션을 제공합니다.

```
Objects:
    domain
    host
    contact
Actions:
    check
    info
    create
    update
```

```
delete
transfer
renew
```

필요에 따라 별도의 addon 방식으로 별도의 패키지를 설치하여 새로운 오브젝트와 액션을 사용할 수 있습니다. .name 도메인의 경우는 기본적인 오브젝트 이외에 nameWatch, EmailFwd 등의 새로운 오브젝트를 추가적으로 사용합니다. 다시 말해서, 레지스트리마다 별도의 새로운 기능이 있기 마련이며 이에 맞는 addon이 필요합니다.

3. 설치

epp-rtk는 현재 sourceforge.net의 공개 프로젝트로 진행중에 있으며 프로젝트 경로는 다음과 같습니다.

epp-rtk sourceforge.net (http://epp-rtk.sourceforge.net)

```
Project UNIX name: epp-rtk
Registered: 2001-05-04 14:48
```

이 프로젝트에 올려진 꾸러미(package)가 현재 liberty-rtk-addon, epp-rtk-c-sample, epp-rtk-cpp, domtools, epp-rtk-java, name-rtk-addon-, us-rtk-addon 등의 7개가 있으며 우리는 .name 등록을 위해 epp-rtk-cpp, domtools, name-rtk-addon을 사용합니다.

3.1. Xerces 설치

Xerces는 C++과 JAVA 2가지 언어를 지원합니다. 우리는 이중에서 C++버젼을 선택합니다.

먼저 설치를 위해 패키지를 다운로드 합니다.

xerces-c1_7_0-linux-7.2.tar.gz (http://xml.apache.org/dist/xerces-c/stable/xerces-c1_7_0-linux7.2.tar.gz).

 # cd /home/dotName # tar xzvf xerces-c1_7_0-linux-7.2.tar.gz

나중에 설치할 epp-rtk가 /usr/local/xerces/lib/xerces.so를 참조하므로 심볼릭 링크를 만들어줍니다.

 # ln -sf /home/dotName/xerces-c1_7_0-linux-7.2/libxerces-c1_7_0.so /usr/local/xerces/lib/lib/xerces.so

3.2. domtools 설치

domtools-1.02.tar.gz (http://prdownloads.sourceforge.net/epp-rtk/domtools-1.02.tar.gz).

 # tar xzvf domtools-1.02.tar.gz

위와 같이 압축을 푼 후에 configure.in 파일을 열어 xerces의 헤더 파일의 경로를 xerces가 설치된 경로에 맞게 추가합니다.

 # vi configure.in

```
CPPFLAGS="-l/usr/local -l/home/dotName/xerces-c1_7_0-linux7.2/include
-l/home/dotName/xerces-c1_7_0-linux7.2/include/xercesc -Wall"
```

이제 final.sh 스크립트를 실행하면 automake 부터 해서 make install 까지 모두 진행됩니다.

 # ./final.sh

3.3. epp-rtk 설치

 # tar xzvf epp-rtk-cpp-0.3.0-src.tar.gz

domtools를 설치할 때와 마찬가지로 configure.in 파일을 열어 아래와 같이 xerces 헤더 파일의 경로를 추가해 준다. 단, xerces의 경로가 2개 추가된 것에 주의하기 바랍니다.

 # vi configure.in

```
CPPFLAGS="-l/usr/include -l/usr/local/include/
-l/home/dotName/xerces-c1_7_0-linux7.2/include -l"
```

이제 ./configure를 실행하도록 합니다. 새로 생성된 파일중에 config.cache를 열어 ac_cv_c_const=no라고 되어 있는 것을 yes로 변경하도록 합니다.

 # vi config.cache

```
- ac_cv_c_const=${ac_cv_c_const=no}
+ ac_cv_c_const=${ac_cv_c_const=yes}
```

OT&E 테스트 시에는 SSL 통신을 사용하지 않으므로 testTools.cc 파일에 epp_TransportSSL로 정의된 것을 epp_TransportTCP로 변경합니다.

이제 final.sh를 실행하여 설치를 하도록 합니다.

 # ./final.sh

3.4. name-rtk-addon 설치

epp-rtk-cpp를 설치하는 것과 거의 동일하므로 좀 전에 진행하던 것처럼 진행합니다.

 # tar xzvf name-rtk-addon-0.2.0.tar.gz

 # vi configure.in

```
 CPPFLAGS="-l/usr/include -l/usr/local/include/
-l/home/dotName/xerces-c1_7_0-linux7.2/include
-l/home/dotName/xerces-c1_7_0-linux7.2/include/xercesc
-l/usr/local/include/epprtk"
```

 # vi config.cache

```
- ac_cv_c_const=${ac_cv_c_const=no}
- ac_cv_c_const=${ac_cv_c_const=yes}
```

 # ./final.sh

이것으로서 필요한 파일의 설치는 모두 완료되었습니다.

4. 테스트

epp-rtk의 소스 디렉토리를 보면 login, hello, domaincheck, ... 등의 실행파일이 생성된 것을 확인할 수 있습니다.

 # ./hello -n=[EPP_Host] -p=[Port]

이것을 실행한 후 아래와 같이 greeting 메세지가 보이면 정상적으로 설치된 것입니다.

```
<greeting>
<svID>Verisign GRS EPP Server Stub</svID>
<svDate>2002-04-28T13:36:58.0847Z</svDate>
<svcMenu>
<version>1.0</version>
<lang>en</lang>
<objURI>http://www.nic.name/epp/emailFwd-1.0</objURI>
<objURI>urn:ietf:params:xml:ns:domain-1.0</objURI>
<objURI>urn:ietf:params:xml:ns:contact-1.0</objURI>
<objURI>http://www.nic.name/epp/defReg-1.0</objURI>
<objURI>http://www.nic.name/epp/nameWatch-1.0</objURI>
<objURI>urn:ietf:params:xml:ns:host-1.0</objURI>
<svcExtension>
<extURI>http://www.nic.name/epp/persReg-1.0</extURI>
</svcExtension>
</svcMenu>
</greeting>
```

이제 libdata.a libeprtk.a libnpssl.a libtransport.a 의 파일을 사용하여 EPP 도메인 등록을 위한 프로그래밍을 시작합니다.

5. EPP 활용

이제 원하는 각 기능 별로 프로그램을 만듭니다. 도메인이 등록가능한 지 여부를 검사하는 DomainCheck 기능을 구현한다고 할 때, 등록에 필요한 기본적인 샘플 루틴은 epp-rtk 소스 디렉토리 하위의 src 디렉토리에 main_domaincheck.cc를 참고합니다.

도메인 가능 여부를 요청하는 부분입니다.

```
epp_DomainCheckReq_ref l_req(new epp_DomainCheckReq());

l_req->m_cmd.ref(new epp_Command(NULL,
                 NULL,
                 epp_trid("ABC-12345")));

epp_string_seq_ref names(new epp_string_seq());

names->push_back("example1.com");
names->push_back("example2.com");
names->push_back("example3.com");

l_req->m_names = names;

epp_DomainCheck_ref l_obj(new epp_DomainCheck());

l_obj->setRequestData(*l_req);

test.processAction(l_obj);

epp_DomainCheckRsp_ref l_rsp = l_obj->getResponseData();
```

컴파일 시에는 이와 관련된 라이브러리를 지정해 주어야 합니다. DomainCheck의 경우 EPP-RTK를 설치할 경우 생기는 라이브러리가 필요하기 때문에 컴파일시에 옵션을 적절하게 지정할 필요가 있습니다.

-L./lib -lepprtk -ldata -ltransport -lnpssl

-L./lib를 지정하여 libepprtk.a 파일등이 있는 곳의 위치를 지정해야 합니다.

-DHAVE_DIRENT_H=1 -DSTDC_HEADERS=1 -DHAVE_DOM_DOM_HPP=1 -DHAVE_PARSERS_DOMPARSER_HPP=1 -DHAVE_UT IL_PLATFORMUTILS_HPP=1 -DHAVE_UTIL_XMLSTRING_HPP=1 -DHAVE_UTIL_XMLUNIDEFS_HPP=1 -DHAVE_UNISTD_H=1 -DHAV E_GETPAGESIZE=1 -DHAVE_MMAP=1 -DHAVE_MKDIR=1 -DHAVE_SELECT=1 -DHAVE_STRSTR=1

using namespace eppobject::domain;

using namespace eppobject::epp;

다음은 간단한 Makefile의 예제입니다.

```
CXX = c++
CFLAGS = --ansi -Wall -g -O2

USER_LIB = -lepprtk -ldata -ltransport -lnpssl -lnameaddon -lnamedata
-ldomtools
SYS_LIB = -ldomtools /usr/local/xerces/lib/libxerces.so -lpthread -lssl
-lcrypt -lcrypto -ldata
LIB_DIR = -L./lib -L/usr/local/lib -L/usr/local/lib/domtools
INCL = -I/usr/include -I/usr/local/include/
-I/home/dotName/xerces-c1_7_0-linux7.2/include -I/home/dotN
ame/xerces-c1_7_0-linux7.2/include/xercesc -I/usr/local/include/epprtk
-I/usr/local/include/name-rtk-addon

DEFS = -DHAVE_DIRENT_H=1 -DSTDC_HEADERS=1 -DHAVE_DOM_DOM_HPP=1
-DHAVE_PARSERS_DOMPARSER_HPP=1 -DHAVE_UT
IL_PLATFORMUTILS_HPP=1 -DHAVE_UTIL_XMLSTRING_HPP=1
-DHAVE_UTIL_XMLUNIDEFS_HPP=1 -DHAVE_UNISTD_H=1 -DHAV
E_GETPAGESIZE=1 -DHAVE_MMAP=1 -DHAVE_MKDIR=1 -DHAVE_SELECT=1 -
DHAVE_STRSTR=1

CXXCOMPILE = $(CXX) $(DEFS) $(INCL)
OBJECTS = main.o

all: epp_test

epp_test: $(OBJECTS)
     $(CXX) $(CFLAGS) -o epp_test $(OBJECTS) $(LIB_DIR) $(USER_LIB)
$(SYS_LIB)

.cc.o:
     $(CXXCOMPILE) -c $<

clean:
     rm -rf epp_test *.o
```

6. 참고자료

- EPP Article answers: Why the need for EPP? http://www.nwfusion.com/archive/2000/114159_12-11-2000.html?nf
- Article on EPP from XML Cover Pages October 04, 2001 http://xml.coverpages.org/epp.html
- Presentation by Scott Hollenbeck on EPP http://www.ietf.org/proceedings/00dec/slides/DOMREG-1/
- EPP HOWTO for EPP Registrars http://epp-rtk.sourceforge.net/epp-howto.html

- RTK and EPP Registry Compatibility Chart http://epp-rtk.sourceforge.net/comp_matrix.html
- EPP Public Sandbox, hosted by Tucows http://epp-test.openxrs.net/
- XML Trust Services http://www.verisign.com/developer/xml/epp.html
- Provisioning Registry Protocol(provreg) http://www.ietf.org/html.charters/provreg-charter.html