

ISBN:

ID 페더레이션을 위한 메타데이터 관리 시스템 구축 및 사용

일자	2015년 11월 16일
부서	슈퍼컴퓨팅본부 첨단연구망응용지원실
작성자	이경민, 조진용, 장희진, 공정욱

차 례

1. 개요	3
2. Janus	3
2.1 Janus 설치	3
2.2 Janus 사용	5
2.3 Metadata feed 설정	8
3. Jagger	13
3.1 Jagger 설치	13
3.2 Jagger 사용	16
3.3 메타데이터 서명	21
4. Trouble Shooting	24

SAML 2.0 메타데이터 관리 시스템 구축

1. 개요

- 이 문서는 SAML 2.0 메타데이터 관리 프로그램인 Janus 및 Jagger의 구축 및 사용 방법을 기술한다.
- Janus는 Uninnet에서 배포하는 오픈소스 메타데이터 관리 프로그램이다. simpleSAMLphp에 built on되어 사용된다. 특정 IdP나 SP를 차단할 수 있는 블랙리스트 기능 및 Attribute Manipulation 기능, 메타데이터 생성, 관리, 배포등의 기능을 지원한다. hub & spoke 방식의 페더레이션을 사용할 시 유용하게 사용될 수 있다.
- Jagger는 heanet에서 배포하는 오픈소스 메타데이터 관리 프로그램이다. 메타데이터 생성, 관리, 배포 및 서명 기능을 제공하며 깔끔하고 정리가 잘된 UI를 가지고 있다. Janus가 가지고 있는 BlackList, Manipulation 기능은 없으나 janus에 비해 전체적인 완성도가 높다. full-mesh 방식의 페더레이션에서 메타데이터 관리를 위해 유용하게 사용될 수 있다.

2. Janus

2.1 Janus 설치

- 설치 환경
 - Centos 6.7 x64
 - simpleSAMLphp (>= 1.7.0)
 - Apache (>= 2.2)
 - PHP (>=5.3)
 - MySQL (>= 5.1)
 - Composer
- Janus 설치

본 기술 문서에는 **Janus Version 1.20(2015/05/13)**을 설치한다.

Janus를 설치하기 위해서는 simpleSAMLphp(이하 SSP)가 필요하다. 최신 버전의 SSP를 다운로드 받아 설치 한다. 본 문서에서는 SSP의 설치는 생략하며 정보가 필요시 다음 URL을 참조한다. <https://simplesamlphp.org/>

Janus를 다운로드 한 후 SSP의 module 폴더에 복사한다.

```
# git clone https://github.com/janus-ssp/janus.git
# cp janus /SSP_PATH/modules/ -rf
```

Composer를 설치한다. Composer가 이미 설치 되어 있을 경우 생략한다.

```
# curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin
--filename=composer
```

Composer가 자동적으로 /usr/local/bin/composer란 이름으로 설치된다. composer 명령어를 입력하여 composer 헬프가 나오는지 확인한다.

```
[root@janus test3]# composer

Composer version 1.0-dev (a066171d0c023ad6429881a7692e46f10e080f99) 2015-11-12 13:44:17

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi                    Force ANSI output
  --no-ansi                 Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
  --profile                 Display timing and memory usage information
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  -v|vv|vvv, --verbose      Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  about      Short information about Composer
  archive    Create an archive of this composer package
```

config-dist에 있는 설정 템플릿 파일을 config 디렉토리로 복사한다.

```
# cp /SSP_PATH/modules/janus/app/config-dist/*.yaml /SSP_PATH/modules/janus/app/config/
```

MySQL 데이터베이스를 생성한다. 본 문서에서는 janus 설정파일에 있는 기본 이름인 janus_db를 데이터베이스 이름으로 사용한다.

```
// DB 생성
mysql > create databases janus_db;
// 사용자 생성
mysql > grant all privileges on janus_db.* to USER_ID@'HOST' IDENTIFIED BY 'PW';
mysql > flush privileges;
```

parameters.yml 파일을 열어 관리자 및 DB 정보를 입력한 후 bin 디렉토리 안의 migrate.sh를 호출한다. 이때 bin 디렉토리안이 아닌 janus 디렉토리 (/SSP_PATH/modules/janus)에서 ./bin/migrate.sh 명령어를 입력하여 실행한다. bin 디렉토리 안에서 migrate.sh를 실행할 경우 경로 관련 에러가 발생한다.

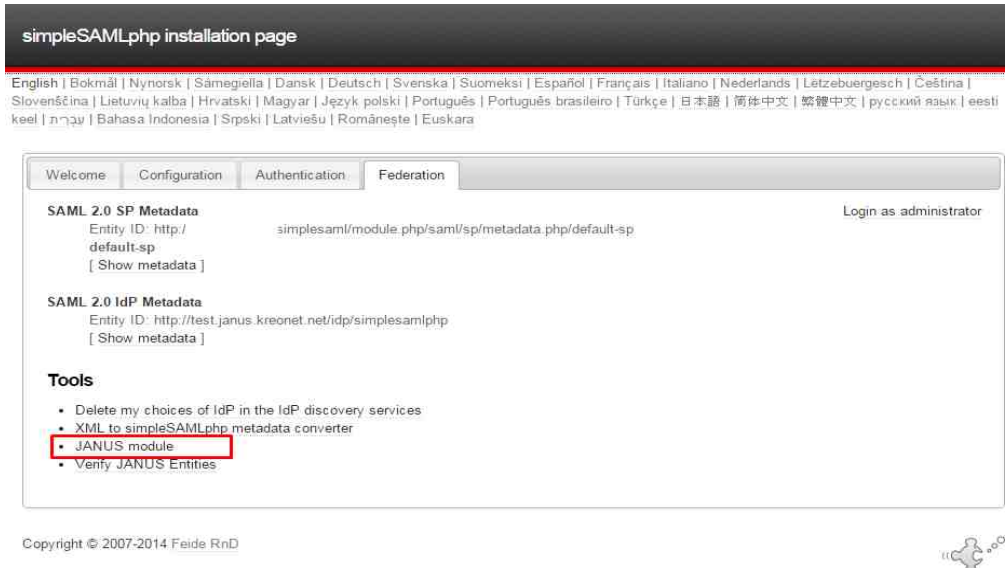
```
# cd /SSP_PATH/modules/janus
# ./bin/migrate.sh
```

timezone 관련 에러가 발생할 경우 php.ini 파일에 다음 라인을 추가한 후 migrate.sh를 다시 실행한다.

```
# vim /etc/php.ini
[Date]
date.timezone = Asia/Seoul
```

2.2 Janus 사용

- 웹 브라우저를 이용해 설치한 SSP로 접속한 후 Federation 탭으로 이동한다. Tools에서 JANUS module을 클릭한다.



simpleSAMLphp installation page

English | Bokmål | Nynorsk | Sámeigiella | Dansk | Deutsch | Svenska | Suomi | Español | Français | Italiano | Nederlands | Lëtzebuergesch | Čeština | Slovenščina | Lietuvių kalba | Hrvatski | Magyar | Język polski | Português | Português brasileiro | Türkçe | 日本語 | 简体中文 | 繁體中文 | русский язык | eesti keel | עברית | Bahasa Indonesia | Srpski | Latviešu | Românește | Euskara

Welcome | Configuration | Authentication | **Federation**

SAML 2.0 SP Metadata Entity ID: http://... Login as administrator

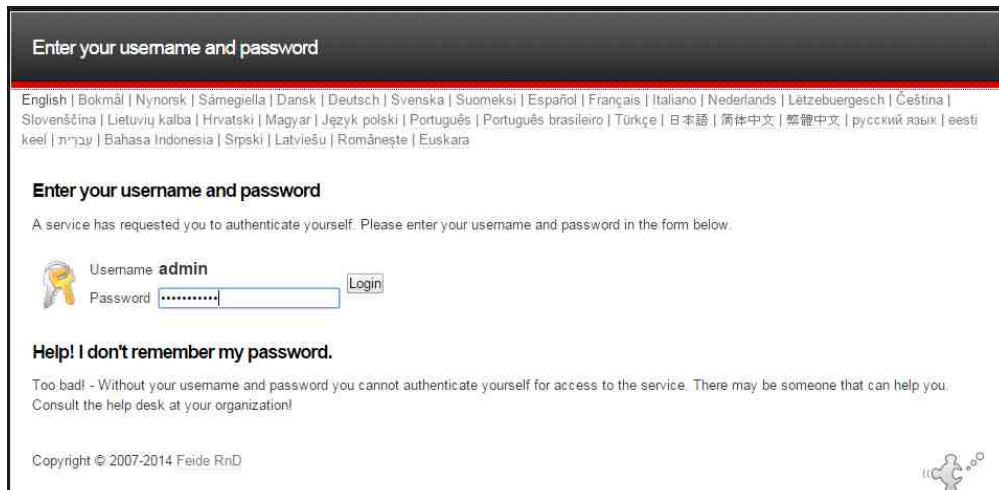
SAML 2.0 IdP Metadata Entity ID: http://test.janus.kreonet.net/idp/simplesamlphp

Tools

- Delete my choices of IdP in the IdP discovery services
- XML to simpleSAMLphp metadata converter
- JANUS module**
- Verify JANUS Entities

Copyright © 2007-2014 Feide RnD

- Admin 패스워드를 입력하고 로그인 한다.



Enter your username and password

English | Bokmål | Nynorsk | Sámeigiella | Dansk | Deutsch | Svenska | Suomi | Español | Français | Italiano | Nederlands | Lëtzebuergesch | Čeština | Slovenščina | Lietuvių kalba | Hrvatski | Magyar | Język polski | Português | Português brasileiro | Türkçe | 日本語 | 简体中文 | 繁體中文 | русский язык | eesti keel | עברית | Bahasa Indonesia | Srpski | Latviešu | Românește | Euskara

Enter your username and password

A service has requested you to authenticate yourself. Please enter your username and password in the form below.

Username **admin** Password [masked] Login

Help! I don't remember my password.

Too bad! - Without your username and password you cannot authenticate yourself for access to the service. There may be someone that can help you. Consult the help desk at your organization!

Copyright © 2007-2014 Feide RnD

- Janus의 대시보드 페이지가 나타난다.



Dashboard for admin

User | **Connections** | Inbox

Create connection

Search

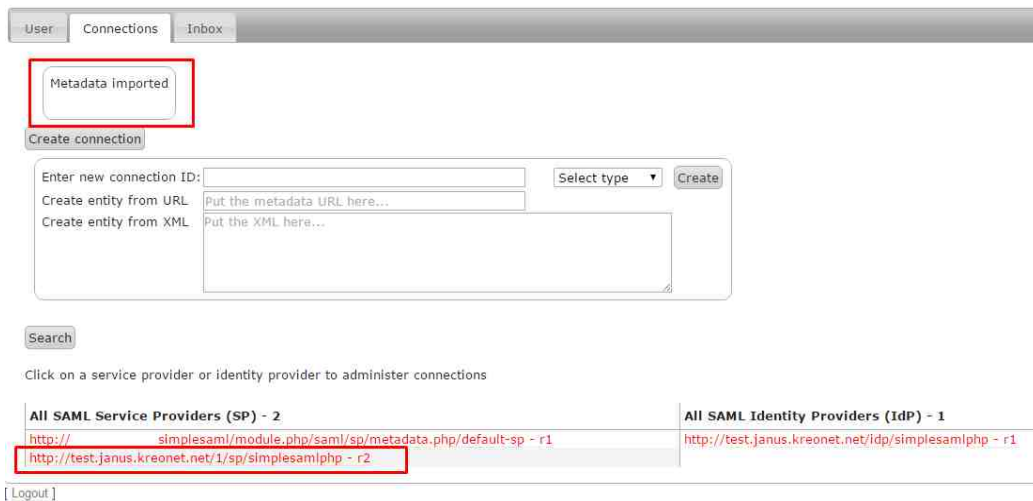
Click on a service provider or identity provider to administer connections

All SAML Service Providers (SP) - 1	All SAML Identity Providers (IdP) - 1
http://.../simplesaml/module.php/saml/sp/metadata.php/default-sp - r1	http://test.janus.kreonet.net/idp/simplesamlphp - r1

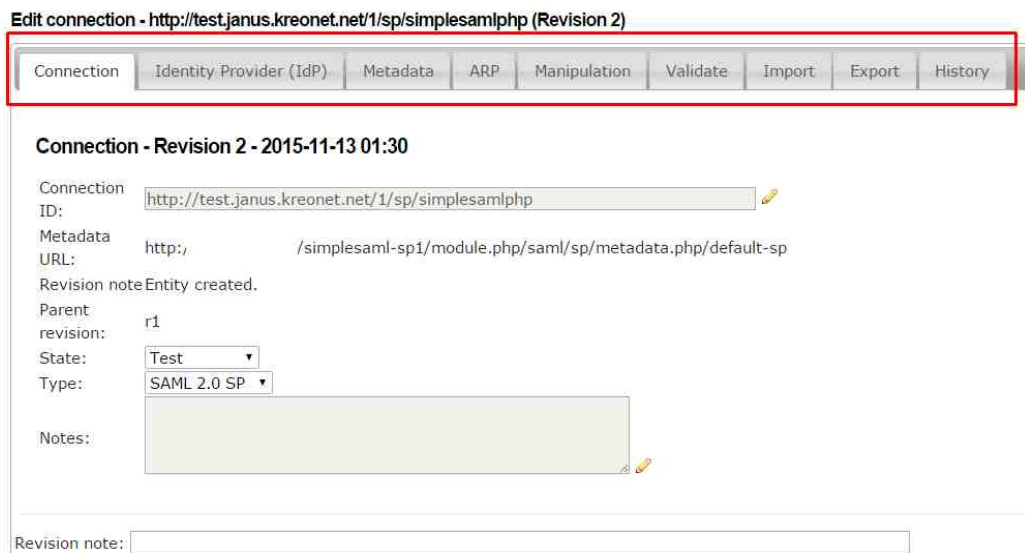
[Logout]

- Create connection 버튼을 눌러 메타데이터를 추가할 수 있다.
- “Enter new connection ID”에 메타데이터의 Entity 이름을 입력한다.
- “Select type”을 클릭해 IdP 혹은 SP를 선택한다.
- 메타데이터를 웹에서 받아올 경우 “Create entity from URL” 입력칸에 메타데이터 배포주소를 입력한다.
- 메타데이터를 XML 형태로 입력할 경우 “Create entity from XML” 부분에 입력한다.
- Create 버튼을 클릭하면 다음과 같이 IdP 혹은 SP가 등록된다.

Dashboard for admin

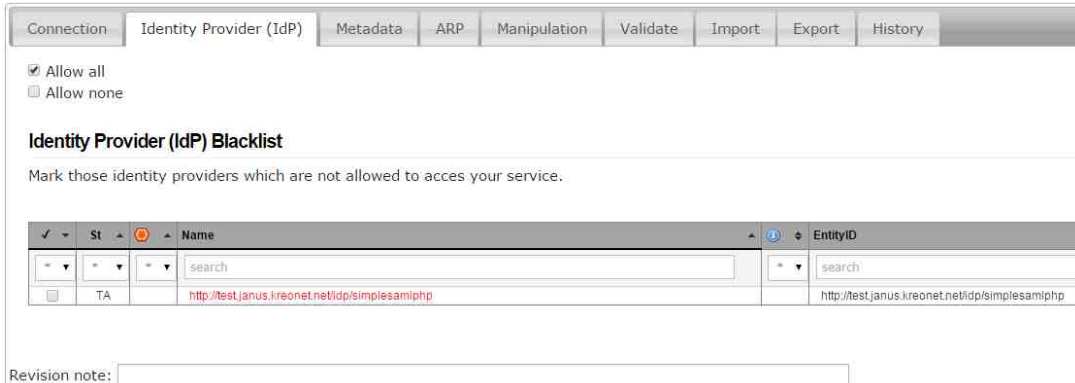


- 추가된 IdP 및 SP를 클릭하면 메타데이터 관리가 가능하다. 상단의 탭을 클릭해 기능을 사용한다.



- Identity Provider Blacklist(SP 기준) 및 Manipulation 기능은 hub and spoke 기반의 페더레이션에서 사용할 수 있다. Blacklist 기능은 SP일 경우 특정 IdP를 통한 서비스 사용

을 차단할 수 있으며 IdP는 특정 SP의 인증 요청을 차단할 수 있다.



- Manipulation은 IdP와 SP의 속성을 조작할 수 있도록 한다. php코드로 작성해야 한다. 아래의 예는 EPPN(eduPersonPrincipalName)에서 UID와 schacHomeOrganization을 추출하기 위해 manipulation 코드이다.

Edit connection - <http://test.janus.kreonet.net/1/sp/simplesamlphp> (Revision 2)

- Export 탭에서 Export metadata를 클릭하면 메타데이터를 확인할 수 있다. XML, JSON 등 다양한 형태의 메타데이터를 제공한다. 단, 메타데이터를 URL 형태로 배포 할 수는 없다. URL 형태로 배포하기 위해서는 별도의 메타데이터 feed 설정이 필요하다.

Metadata export

Here you can see your metadata

Metadata is displayed in JSON, XML format (SAML) or simpleSAMLphp's own dictionary format.

Show/Hide JSON

```
{
  "SingleLogoutService": [
    {
      "Location": "http://localhost/simplesaml-sp1/module.php/saml/sp/saml2-logout.php",
      "Binding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
    }
  ],
  "AssertionConsumerService": [
    {
      "Location": "http://localhost/simplesaml-sp1/module.php/saml/sp/saml2-acs.php/d",
      "index": 0,
      "Binding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    }
  ],
  "certData": {
    "metadata-set": "saml20-sp-remote",
    "NameIDFormat": "urn:oasis:names:tc:SAML:2.0:nameid-format:transient",
    "name": {
      "en": "http://test.janus.kreonet.net/1/sp/simplesamlphp"
    },
    "eid": "g"
  }
}
```

Show/Hide PHP

Show/Hide XML

[Back - Dashboard](#)

2.3 Metadata feed 설정

- Metadata feed

Janus를 통해 URL형태로 메타데이터를 배포하기 위해서는 metadataexport 기능을 이용해야 한다. metadataexport는 다음 주소로 접속하면 이용할 수 있다.

http://JANUS-URL/modules.php/janus/metadataexport.php?id='FEED_ID'

Janus의 code google 페이지를 참조하면 mdexport 설정을 통해 다수의 FEED ID를 생성하여 여러 종류의 FEED(ex : IdP만 배포, SP만 배포 등)를 생성할 수 있다고 설명하고 있으나 version 1.20에서는 mdexport 설정을 수행해도 오직 Production(FEED_ID = prod) 상태의 메타데이터만 배포할 수 있었다. 다양한 메타데이터를 배포할 수 있도록 일부 설정 파일을 수정한다.

1. 새로운 FEED를 생성하기 위해 다음 파일을 수정한다.

/SSP_PATH/modules/janus/src/Janus/ServiceRegistry/Bundle/CoreBundle/DependencyInjection/Configuration.php

Configuration.php 파일에 아래 내용을 입력한다. 입력된 내용은 IdP 및 SP를 배포하기 위한 설정이다. BOLD 처리된 부분을 수정하여 다른 FEED를 생성하는 것도 가능하다.

```

$mdExportBuilder
    ->arrayNode('feeds_sp')->children()
        ->arrayNode('sp')->children()
            ->arrayNode('types')
            ->prototype('scalar')->end()
        ->end()
    ->arrayNode('states')
        ->prototype('scalar')->end()
    ->end()
    ->scalarNode('mime')->end()
    ->arrayNode('exclude')
        ->prototype('scalar')->end()
    ->end()
    ->scalarNode('postprocessor')->end()
    ->scalarNode('entitiesDescriptorName')->end()
    ->scalarNode('filename')->end()
    ->scalarNode('maxCache')->end()
    ->scalarNode('maxDuration')->end()
    ->arrayNode('sign')->children()
        ->booleanNode('enable')->end()
        ->scalarNode('privatekey')->end()
        ->scalarNode('privatekey_pass')->end()
        ->scalarNode('certificate');

$mdExportBuilder
    ->arrayNode('feeds_idp')->children()
        ->arrayNode('idp')->children()
            ->arrayNode('types')
            ->prototype('scalar')->end()
        ->end()
    ->arrayNode('states')
        ->prototype('scalar')->end()
    ->end()
    ->scalarNode('mime')->end()

```

```
->arrayNode('exclude')
  ->prototype('scalar')->end()
->end()
->scalarNode('postprocessor')->end()
->scalarNode('entitiesDescriptorName')->end()
->scalarNode('filename')->end()
->scalarNode('maxCache')->end()
->scalarNode('maxDuration')->end()
->arrayNode('sign')->children()
  ->booleanNode('enable')->end()
  ->scalarNode('privatekey')->end()
  ->scalarNode('privatekey_pass')->end()
  ->scalarNode('certificate');
```

2. 생성된 FEED로 가져올 메타데이터의 조건을 설정한다.

```
/SSP_PATH/modules/janus/app/config/config_janus_core.yml
```

config_janus_core.yml 파일의 mdexport: postprocessor: feeds: 아래에 다음 내용을 추가한다. type과 state를 변경하여 배포하기 위한 메타데이터를 설정할 수 있다.

```
mdexport:
  postprocessor:
    filesystem:
      .....
      FTP:
      .....
  feeds_sp:
    sp:
      types:
        - saml20-sp
      states:
        - testaccepted
      mime: application/samlmetadata+xml
      exclude:
        - 'https://example.org/saml/metadata.xml'
      postprocessor: janus:FileSystem
      entitiesDescriptorName: 'FederationMetadata'
      filename: FederationMetadata.xml
      maxCache: 172800 # 24 hour cache time
```

```

maxDuration: 604800
sign:
  enable: true
  privatekey: /PATH/private.key
  privatekey_pass: null
  certificate: /PATH/cert.crt

feeds_idp:
  idp:
    types:
      - saml20-idp
    states:
      - testaccepted
  mime: application/samlmetadata+xml
  exclude:
    - 'https://example.org/saml/metadata.xml'
  postprocessor: janus:FileSystem
  entitiesDescriptorName: 'FederationMetadata'
  filename: FederationMetadata.xml
  maxCache: 172800 # 24 hour cache time
  maxDuration: 604800
  sign:
    enable: true
    privatekey: /PATH/private.key
    privatekey_pass: null
    certificate: /PATH/cert.crt

```

3. metadataexport.php 파일에 새롭게 추가한 FEED를 입력한다.

```
/SSP_PATH/modules/janus/www/metadataexport.php
```

metadataexport.php 파일의 58번째 라인에 다음 내용을 입력한다.

```

$md_feeds_sp = $janus_config->getArray('mdexport.feeds_sp');
$md_feeds_idp = $janus_config->getArray('mdexport.feeds_idp')

#%md_feeds에 새롭게 생성한 feeds를 병합한다.
$md_feeds = array_merge($md_feeds, $md_feeds_sp, $md_feeds_idp);

```

다음 URL 주소로 접근하여 메타데이터 배포가 정상적으로 이루어지는지 확인한다.

Production 상태의 메타데이터 다운로드

http://SERVER_ADDR/simplesaml/module.php/janus/metadataexport.php?id=prod

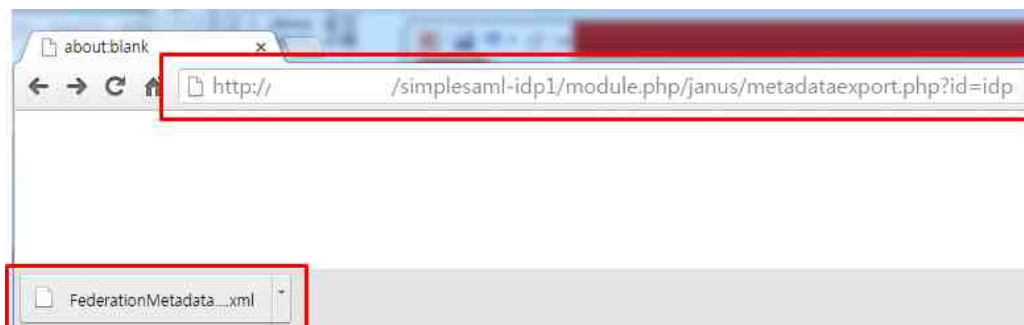
SP만 다운로드

http://SERVER_ADDR/simplesaml/module.php/janus/metadataexport.php?id=sp

IdP만 다운로드

http://SERVER_ADDR/simplesaml/module.php/janus/metadataexport.php?id=idp

※ 메타데이터 다운로드 확인



※ 다운로드된 메타데이터 내용 일부

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" Name="FederationMetadata" cacheDuration="PT172800S" validUntil="2015-11-19T17:47:34Z" ID="pfxcl#98477-1ab8-0a96-eae6-de0219c1a3ef">
  <Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
      <ds:Reference URI="#pfxcl#98477-1ab8-0a96-eae6-de0219c1a3ef">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <ds:DigestValue>7r/3/XuolY30uK5MH6J8k4189kk=/ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>VdUrkGLUSZUHTWefH+twdr2bDndXvtLbxU574Tm1fDZA1cultcaW0UEMLMab4rcw1AV5Dwqz4JueE768zj01M1BULLrsejGe5ztVh7qDN5xJG3f13JGaubC2reqASrvV1/cjPX2W1/ZDw1201bUA
    </ds:SignatureValue>
  </Signature>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:KeyInfo>
        <ds:Signature>
          <ds:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
            <ds:KeyDescriptor use="signing">
              <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                  <ds:X509Certificate>
                    MIICCTCAV4ODCC+bnEzvoZDANBgkqhkiG9w0BAQFADEMDQwCQ/DYQDCEwJUUJECMAwGA1UECHMFSEIYBjEwHwYJKoZIhvcNAQkGQzZ2ZkvcG1IbnQzGjAYBgNVBAMT
                    </ds:X509Certificate>
                  </ds:X509Data>
                </ds:KeyInfo>
              </ds:KeyDescriptor>
            </ds:KeyDescriptor>
            <ds:KeyDescriptor use="encryption">
              <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                  <ds:X509Certificate>
                    MIICCTCAV4ODCC+bnEzvoZDANBgkqhkiG9w0BAQFADEMDQwCQ/DYQDCEwJUUJECMAwGA1UECHMFSEIYBjEwHwYJKoZIhvcNAQkGQzZ2ZkvcG1IbnQzGjAYBgNVBAMT
                    </ds:X509Certificate>
                  </ds:X509Data>
                </ds:KeyInfo>
              </ds:KeyDescriptor>
            </ds:KeyDescriptor>
            <ds:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="http://.../simplesaml-
            sp/module.php/saml/sp/saml2-logout.php/default-sp"/>
          </ds:SPSSODescriptor>
          <ds:NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:transient
          </ds:NameIDFormat>
          <ds:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="http://.../simplesaml-
          sp/module.php/saml/sp/saml2-acss.php/default-sp" index="0"/>
        </ds:EntityDescriptor>
      </ds:EntityDescriptor>
    </EntityDescriptor>
  </EntityDescriptor>
```

3. Jagger

3.1 Jagger 설치

- 설치 환경

- Centos 6.7 x64
- Httpd(Apache) (>= 2.2)
- PHP (>=5.5)
- MySQL (>= 5.1)
- Codeigniter framework (>= 3.0)
- Composer

- Jagger 설치

본 기술 문서에서는 **Jagger Version v1.5.0**을 설치한다.

본 기술문서에서 Apache / Mysql / PHP의 설치는 생략한다. 사전에 설치환경과 부합하는 버전의 프로그램들이 설치되어 있어야 한다. Jagger 웹 페이지는 Codeigniter 프레임워크를 사용한다. Codeigniter는 PHP 기반의 웹 프레임워크이다. 일반적으로 필요한 프로그래밍 라이브러리 및 보안 옵션(XSS 필터링, CSRF 보호 등)을 제공하며 MVC(Model-View-Controller) 기반의 웹 사이트를 구축하는데 도움을 준다. 먼저 Codeigniter 프레임워크를 다운로드 한다.

```
# git clone git://github.com/bcit-ci/CodeIgniter.git
# cp -rf ./CodeIgniter /opt/codeigniter
```

jagger를 Apache의 Document Root 디렉토리에 다운로드 한다.

```
# git clone https://github.com/Edugate/Jagger.git
# cp -rf ./Jagger /var/www/rr3
```

Httpd 설정을 진행한다.

```
# vim /etc/httpd/conf.d/ssl.conf
Alias /rr3 /var/www/rr3
<Directory /var/www/rr3>
    RewriteEngine On
    RewriteBase /rr3
    RewriteCond $1 !^(Shibboleth|\.ssolindex|\.php|logos|signedmetadata|
                                flags|images|apps|schemas|fonts|styles|images|js|robots
                                |\.txt|pub|includes|cert)
    RewriteRule ^(.*)$ /rr3/index.php?/$1 [L]
</Directory>
<Directory /var/www/rr3/application>
    Order allow,deny
    Deny from all
</Directory>
```

php.ini의 설정을 다음과 같이 변경한 후 apache를 재시작한다.

```
# vim /etc/php.ini
memory_limit = 256M
max_execution_time = 60

# service httpd restart
```

Mysql 설정을 진행한다. 데이터베이스와 데이터베이스 사용자를 생성한다.

```
mysql> create database rr CHARACTER SET utf8 COLLATE utf8_general_ci;
mysql> grant all on rr.* to rr@'localhost' identified by 'PASSWORD';
mysql> flush privileges;
```

Jagger 디렉토리로 이동하여 install.sh 파일을 실행한다. Jagger 서비스에 필요한 부가 프로그램 및 라이브러리가 다운로드 되고 템플릿 설정파일이 생성된다.

```
# /var/www/rr3
# install.sh
```

```
[root@janus rr3]# ./install.sh
Script will create additional folders and downloads additional software. After script is finished please
run composer (https://getcomposer.org/) on composer.json - it will install Doctrine 2.4.x and Zend-ACL
logos2 directory doesnt exist....creating
done
--2015-11-15 06:52:13-- http://downloads.sourceforge.net/project/geshi/geshi/GeSHi%201.0.8.11/GeSHi-1.0.8
.11.tar.gz?r=http%3A%2F%2Fsourceforge.net%2Fprojects%2Fgeshi%2Ffiles%2Flatest%2Fdownload%3Fsource%3Dfiles&
as=1346371975&use_mirror=heanet
Resolving downloads.sourceforge.net... 216.34.181.59
Connecting to downloads.sourceforge.net[216.34.181.59]:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://heanet.dl.sourceforge.net/project/geshi/geshi/GeSHi%201.0.8.11/GeSHi-1.0.8.11.tar.gz [fol
lowing]
--2015-11-15 06:52:13-- http://heanet.dl.sourceforge.net/project/geshi/geshi/GeSHi%201.0.8.11/GeSHi-1.0.8
.11.tar.gz
Resolving heanet.dl.sourceforge.net... 193.1.193.66, 2001:770:18:aa40::c101:c142
Connecting to heanet.dl.sourceforge.net[193.1.193.66]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1025858 (1002K) [application/x-gzip]
Saving to: "/tmp/tmp.7UffikYEDA/GeSHi-1.0.8.11.tar.gz"

100%[=====>] 1,025,858 345K/s in 2.9s

2015-11-15 06:52:17 (345 KB/s) - "/tmp/tmp.7UffikYEDA/GeSHi-1.0.8.11.tar.gz" saved [1025858/1025858]

Done!!!
Now go to application/config
copy below config files and customize them:
=====
config-default.php -> config.php
config_rr-default.php -> config_rr.php
database-default.php -> database.php
email-default.php -> email.php
memcached-default.php -> memcached.php
=====
```

Jagger 설치를 위해 템플릿 설정파일을 복사하고 다음과 같이 변경한다.

```
# cd /var/www/rr3/application/config/
# cp config-default.php config.php
# cp database-default.php database.php
```

```
# cp config_rr-default.php config_rr.php
# cp email-default.php email.php

[ config.php ]
$config['base_url'] = 'https://SERVER_ADDRESS/rr3'

[ database.php ]
$db['default']['username'] = 'rr'; //User Name
$db['default']['password'] = 'PASSWORD';
$db['default']['database'] = 'rr'; //Database Name
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = '';

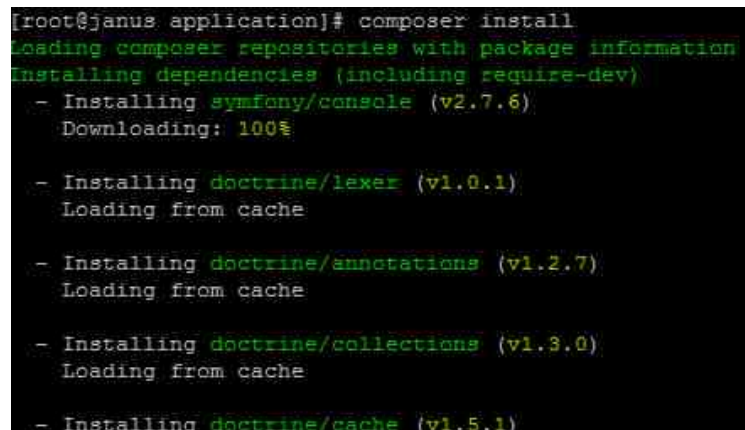
[ config_rr.php ]
$config['rr_setup_allowed'] = TRUE;
```

codeigniter 프레임워크에서 index.php 파일을 Jagger의 루트 디렉토리에 복사한 후 timezone과 codeigniter 프레임워크의 경로를 설정한다.

```
# cd /var/www/rr3
# cp /opt/codeigniter/index.php ./
# vim index.php
date_default_timezone_set('Asia/Seoul'); (index.php 추가)
$system_path = '/opt/codeigniter/system'; (index.php에서 변경)
```

composer 명령을 이용해 서비스에 필요한 추가 어플리케이션을 다운로드한다.

```
# cd /var/www/rr3/application
# composer install
```



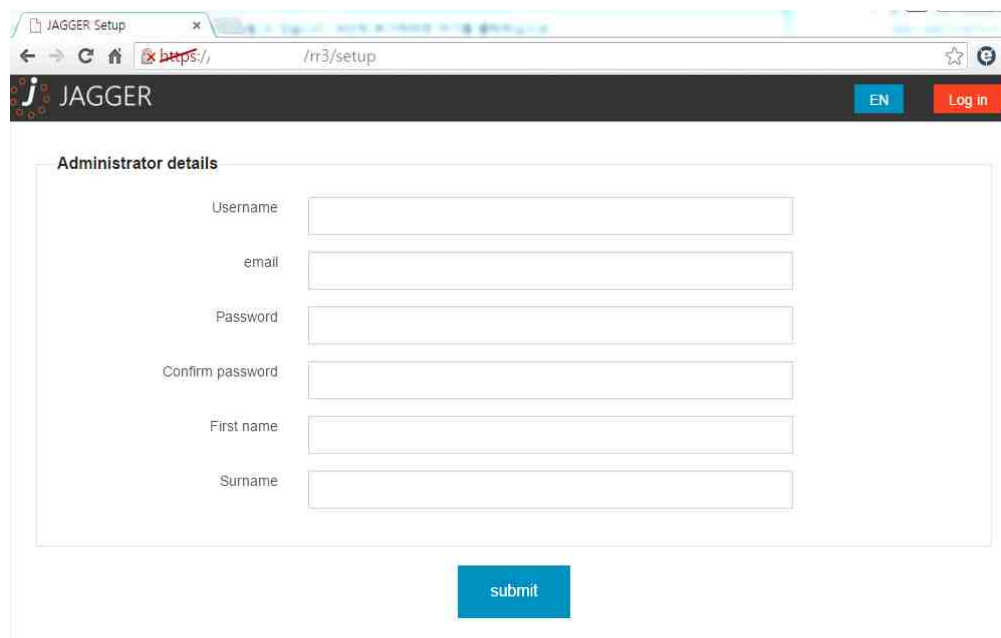
```
[root@janus application]# composer install
Loading composer repositories with package information
Installing dependencies (including require-dev)
- Installing symfony/console (v2.7.6)
  Downloading: 100%
- Installing doctrine/lexer (v1.0.1)
  Loading from cache
- Installing doctrine/annotations (v1.2.7)
  Loading from cache
- Installing doctrine/collections (v1.3.0)
  Loading from cache
- Installing doctrine/cache (v1.5.1)
```

Database 및 프록시 관련 설정을 진행한다.

```
# cd /var/www/rr3/application
# ./doctrine orm:schema-tool:create
# ./doctrine orm:generate-proxies
```

다음 URL로 접속하여 Jagger의 Admin 계정을 설정한다.

```
https://SERVER_ADDRESS/rr3/setup
```



! 설정 완료 후 반드시 config_rr.php 파일의 설정을 `$config['rr_setup_allowed'] = FALSE;` 로 변경해야 한다. 설정 변경 후 setup 페이지 접속 시 다음과 같은 에러 페이지가 출력되어야 한다.



3.2 Jagger 사용

- 웹 브라우저를 통해 jagger 웹페이지로 이동한다. 우측 상단의 Login 버튼을 클릭하여 사용자 인증을 수행한다.

X

Log in with local account

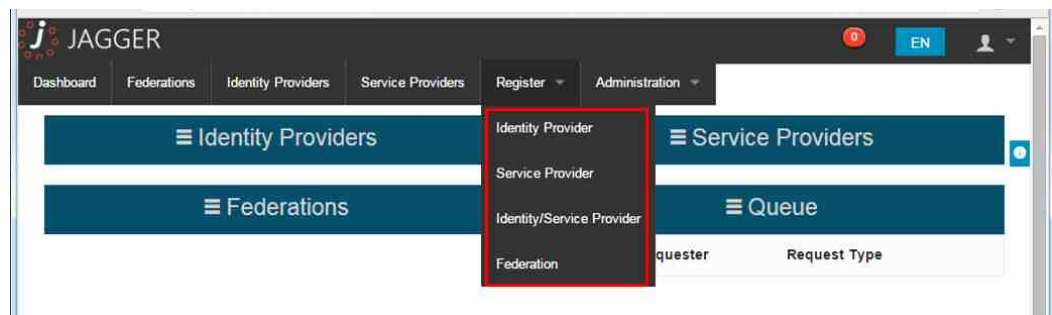
Username

Password

- 로그인 성공 시 Jagger의 대시보드 화면을 볼 수 있다. 최근 등록한 IdP, SP의 정보나 관리자의 승인을 기다리는 메타데이터 정보를 볼 수 있다.



- Register 탭을 클릭하여 IdP 및 SP를 등록한다.



- IdP를 등록하여 본다. Register 탭의 Identity Provider를 클릭한다.

- * Start over, Save draft, Register 버튼

* 필수 속성 체크

- 18 -

* 웹 GUI를 통한 속성 추가

General
Organization
Contacts
UI Information
UI Hints
SAML
Certificates

Name of organization
English (en) organization name Remove
Abkhaz (ab) Add in new language

Displayname of organization
English (en) displayname Remove
Abkhaz (ab) Add in new language

URL to information about organization
English (en) url information Remove
Abkhaz (ab) Add in new language

Start over Save draft Register

- 등록한 IdP / SP / Federation은 큐에 등록되며, 관리자의 승인(Accept)이 필요하다.

JAGGER
Dashboard
Federations
Identity Providers
Service Providers
Register
Administration

Identity Providers
Service Providers
Federations
Queue

Date	Requester	Request Type
2015-11-15 18:11:13	COREEN COREEN (coreen)	IDP - Create

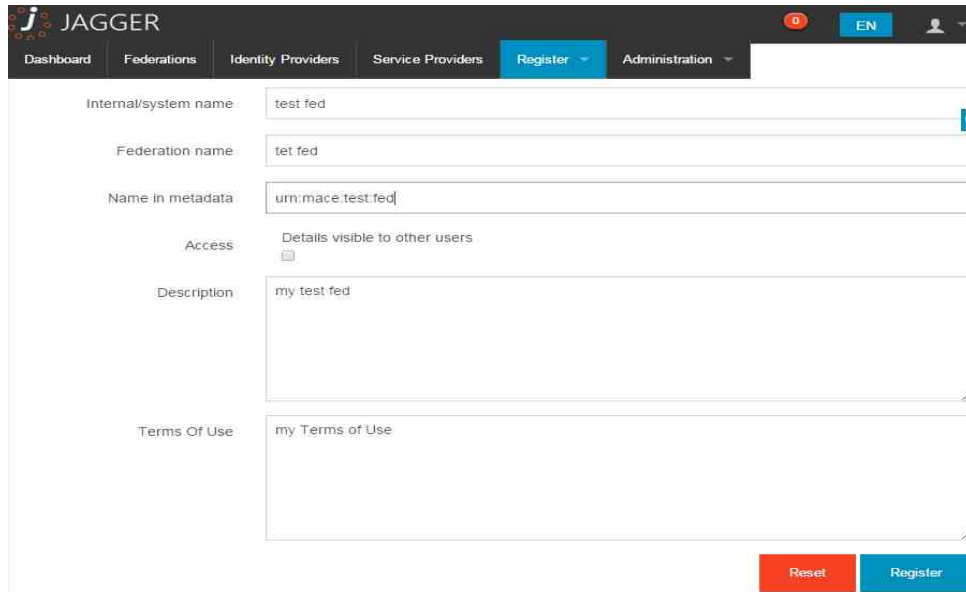
- 관리자 승인 페이지

JAGGER
Dashboard
Federations
Identity Providers
Service Providers
Register
Administration

Request awaiting for approval
Details
Requestor COREEN COREEN (coreen)
Source IP
Federation(s) to join None at the moment

Reject request Accept request

- Register 탭의 Federation을 클릭하여 페더레이션을 등록한다. IdP 등록과 동일하게 Queue에 등록되며 관리자가 승인(Accept)해 주어야 한다.



The image shows the 'Register' tab in the JAGGER interface. The form contains the following fields:

- Internal/system name: test fed
- Federation name: tet fed
- Name in metadata: urn:mace:test:fed
- Access: Details visible to other users (checkbox)
- Description: my test fed
- Terms Of Use: my Terms of Use

At the bottom right, there are 'Reset' and 'Register' buttons.

- Federation 탭에서 등록한 페더레이션 이름을 클릭하여 멤버(IdP / SP)를 추가한다.

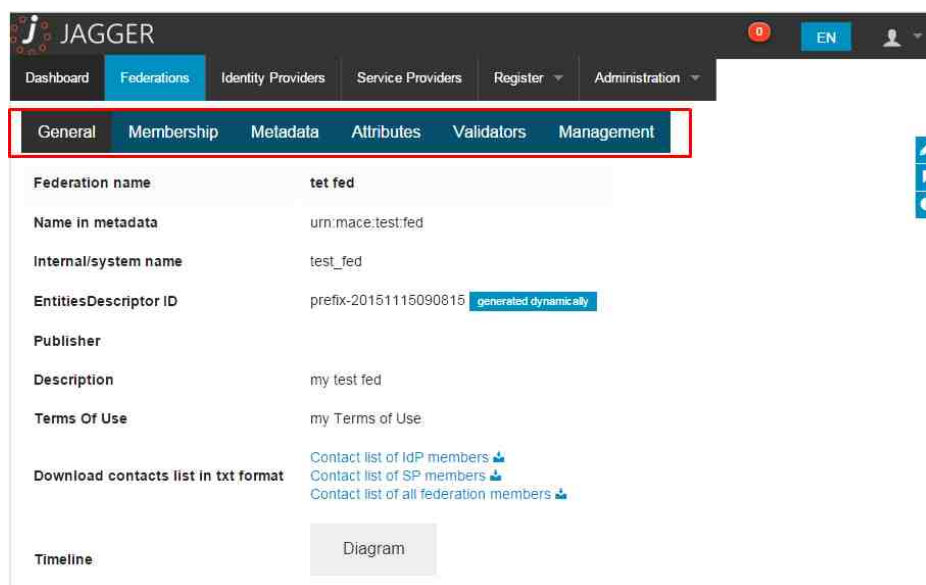


Name	Name in metadata	Description	#
tet fed	urn:mace:test:fed	my test fed	

The table shows a single entry for 'tet fed' with its metadata and description. A red box highlights the first row.

- 탭 메뉴를 통해 멤버를 추가하거나 메타데이터를 배포할 수 있다.

* 탭 메뉴



The image shows the 'Federation' tab selected in the JAGGER interface. The 'General' sub-tab is active, showing details for the 'tet fed' federation. A red box highlights the sub-tab menu.

Sub-tab menu:

- General
- Membership
- Metadata
- Attributes
- Validators
- Management

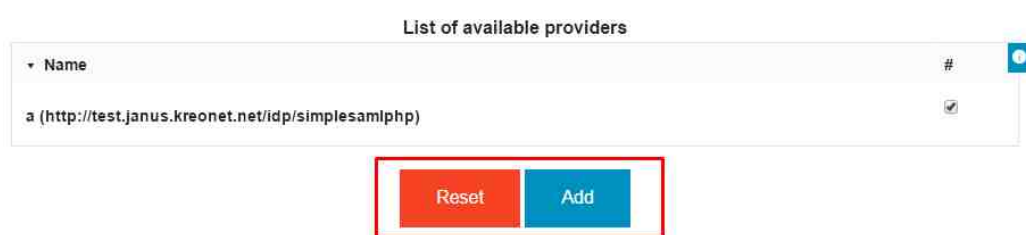
Details for 'tet fed':

- Federation name: tet fed
- Name in metadata: urn:mace:test:fed
- Internal/system name: test_fed
- EntitiesDescriptor ID: prefix-20151115090815 (generated dynamically)
- Publisher:
- Description: my test fed
- Terms Of Use: my Terms of Use
- Download contacts list in txt format:
 - Contact list of IdP members
 - Contact list of SP members
 - Contact list of all federation members
- Timeline: Diagram

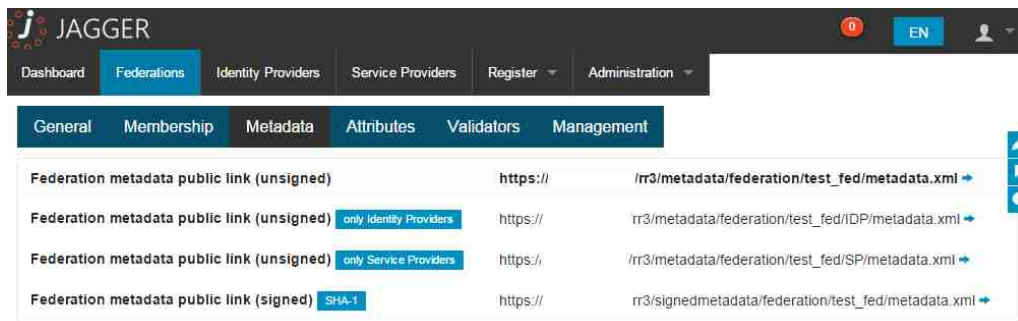
* IdP 및 SP 초대



* invitation 버튼을 클릭하여 Jagger에 등록된 IdP나 SP를 페더레이션에 등록 가능



- Metadata 탭을 클릭하여 메타데이터를 배포할 수 있다. IdP/SP를 통합하여 배포하거나 IdP와 SP를 분리하여 배포할 수 있다. 페더레이션 메타데이터 전체를 전자서명하여 배포할 수 있으며 서명을 위해서는 별도의 스크립트 실행이 필요하다. 메타데이터 서명은 3.3 메타데이터 서명 부분을 참고한다.



3.3 메타데이터 서명

메타데이터 서명을 위해서는 Java와 xmlsectool이 필요하다. 본 기술 문서에서 Java 설치에 생략한다. xmlsectool을 다운로드 받기 위해 wget을 사용한다.

```
# wget http://shibboleth.net/downloads/tools/xmlsectool/latest/xmlsectool-1.2.0-bin.zip
# unzip xmlsectool-1.2.0-bin.zip
```

다음 스크립트를 생성한다. 진하게 처리된 부분의 경로를 자신의 서버에 알맞게 수정한다.

```
# vim sign.sh
```

```
#!/bin/bash
export JAVA_HOME=/JAVA_PATH/bin
# optional args
G=$1
H=$2
XMLSECTOOLDIR="/PATH/xmlsectool-1.2.0"
SIGNCERT="/CERT_PATH/metadata-signer.crt"
SIGNKEY="/KEY_PATH/metadata-signer.key"
SIGNPASS="YOUR_STRONG_PASS_FOR_PRV_KEY"
RR3_PATH="/JAGGER_PATH/rr3"
RR3_URL="https://YOUR_SITE/rr3";
Y=`tempfile`
cd ${XMLSECTOOLDIR}
if [ $G == "provider" ]; then
    wget --no-check-certificate -O ${Y} ${RR3_URL}/${H}
else
    wget --no-check-certificate -O ${Y} ${RR3_URL}
fi
for i in `cat ${Y}`; do
    group=`echo $i|awk -F ";" '{ print $1 }'|tr -d ' '`
    name=`echo $i|awk -F ";" '{ print $2 }'|tr -d ' '`
    srcurl=`echo $i|awk -F ";" '{ print $3 }'|tr -d ' '`

    #tempofileoutput="/tmp/${name}"
    dstoutput="/JAGGER_PATH/rr3/signedmetadata/${group}/${name}"
    if [ ! -d "/JAGGER_PATH/rr3" ]; then
        exit 3
    fi
    if [ ! -d "$dstoutput" ]; then
        mkdir -p $dstoutput
    fi
    ${XMLSECTOOLDIR}/xmlsectool.sh --sign --certificate ${SIGNCERT}
--key ${SIGNKEY} --keyPassword ${SIGNPASS} ₩
    --outFile ${dstoutput}/metadata.xml --inUrl ${srcurl}
done
rm ${Y}
```

위의 스크립트를 사용시 SHA-1을 이용해 서명된다. SHA-2 서명을 수행하고 싶을 경우 signsh 파일을 다음과 같이 수정한다.

```
# vim sign.sh

.....
if [ ! -d "$dstoutput" ]; then
    mkdir -p $dstoutput
fi
${XMLSECTOOLDIR}/xmlsectool.sh --sign --digest SAH-256
--certificate ${SIGNCERT} --key ${SIGNKEY} --keyPassword ${SIGNPASS} \W
--outFile ${dstoutput}/metadata.xml --inUrl ${srcurl}
done
rm ${Y}
```

스크립트를 실행하면 메타데이터 서명이 이루어진다.

```
# ./sign.sh
```

주기적인 메타데이터 서명 및 로그 수집을 위해 cron과 logrotate 설정을 수행한다. 아래의 cron 설정은 매 6시간(0시, 6시, 12시, 18시)에 서명을 수행하기 위한 것이다.

```
# mkdir -p /var/log/jagger/sign/
# crontab -e
0 0,6,12,18 * * * /SCRIPT_PATH/sign.sh federation FEDERATION_NAME
>> /var/log/jagger/sign/message.log 2>&1
```

logrotate를 설정한다.

```
# vim /etc/logrotate.d/jagger_metadata_sign

/var/log/jagger/sign/*.log
{
    daily
    rotate 53
    missingok
    copytruncate
    compress
    delaycompress
    notifempty
    sharedscripts
    dateext
}
```

다음 스크린샷은 SHA-256은 서명된 페더레이션 메타데이터 중 일부이다.

```

<!--
Metadata was generated on: 2015-11-16 01:01 UTC
TERMS OF USE
-->
<?xml-stylesheet type="text/xsl" href="xsl/RequestInit.xsl" />
<md:EntitiesDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" xmlns="urn:oasis:names:tc:SAML:2.0:met
xmlns:init="urn:oasis:names:tc:SAML:profiles:SSO:request-init" xmlns:adattr="urn:oasis:names:tc:SAML:metadata
xmlns:shibed="urn:scac:shibboleth:metadata:1.0" xmlns:ukfedlabel="http://uk.federation.org.uk/2006/11/label" >
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" >
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc1411#" />
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc1411#" />
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha256" />
        <ds:DigestValue>
          <ds:DigestValue>
        </ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
      <ds:SignatureValue>
    </ds:SignatureValue>
  </ds:Signature>
  <ds:KeyInfo>
    <ds:KeyValue>
      <ds:RSAKeyValue>
        <ds:Modulus>
          <ds:Modulus>
        </ds:Modulus>
        <ds:Exponent>
          <ds:Exponent>
        </ds:Exponent>
      </ds:RSAKeyValue>
    </ds:KeyValue>
    <ds:X509Data>
      <ds:X509Certificate>
        <ds:X509Certificate>
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</md:EntitiesDescriptor>

```

4. Trouble Shooting

- composer 실행 시 php 관련 버전에서 발생

```

[root@janus application]# composer install
Loading composer repositories with package information
Installing dependencies (including require-dev)
Your requirements could not be resolved to an installable set of packages.

Problem 1
- lcobucci/jwt 3.1.0 requires php >=5.5 -> your PHP version (5.4.44) or "config.platform.php" value do
es not satisfy that requirement.

```

Jagger version 1.5.0에서 composer를 이용해 third-party 애플리케이션을 다운로드 할 경우 php 버전 5.5 이상을 요구한다. CentOS일 경우 remi 레포지토리를 설치하고 php 5.5를 설치한다.

```

# rpm -Uvh http://rpms.famillecollet.com/enterprise/remi-release-6.rpm
// 업데이트 시
# yum --enablerepo=remi,remi-php55 update php {OTHER LIBRARIES}
// 최초 설치 시
yum --enablerepo=remi,remi-php55 install php {OTHER LIBRARIES}

```


- Jagger를 사용해 IdP 및 SP 등록시 쓰기 권한 관련 에러 발생
Jagger의 application 디렉토리에 쓰기 권한을 부여한다.

```
chown -R apache.apache /JAGGER_PATH/rr3/application
```

- Jagger 웹 페이지에 접속시 PHP 에러 발생
Message : MemcachePool::set()....

방안 1. yum을 이용해 Memcached 패키지를 설치한다. 이미 설치되어 있을 경우 서비스를 실행시킨다.

```
* 설치
# yum install memcached
// remi repo를 사용하여 php를 설치하였을 시
# yum --enablerepo=remi,remi-php55 install memcached

* 서비스 실행
service memcached start
```

방안2. iptables 체크. iptables를 통해 memcached 서버 포트를 열어준다.

```
-A INPUT -s localhost -p tcp --dport 11211 -j ACCEPT
-A INPUT -s localhost -p udp --dport 11211 -j ACCEPT
```