

Brekeke SIP Server

Version 2

Authentication Plug-in Developer's Guide

Brekeke Software, Inc.

Version

Brekeke SIP Server v2 Authentication Plug-in Developer's Guide

Revised September, 2010

Copyright

This document is copyrighted by Brekeke Software, Inc.

Copyright ©2003-2010 Brekeke Software, Inc.

This document may not be copied, reproduced, reprinted, translated, rewritten or readdressed in whole or part without expressed, written consent from Brekeke Software, Inc.

Disclaimer

Brekeke Software, Inc. reserves the right to change any information found in this document without any written notice to the user.

Trademark Acknowledgement

- ◆ *LINUX is a registered trademark of Linus Torvalds in the United States and other countries.*
- ◆ *Red Hat is a registered trademark of Red Hat Software, Inc.*
- ◆ *Windows is a trademark or registered trademark of Microsoft Corporation in the United States and other countries.*
- ◆ *Mac is a trademark of Apple Computer, Inc., registered in the U.S. and other countries.*
- ◆ *Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.*
- ◆ *Other logos and product and service names contained in this document are the property of their respective owners.*

1.	WELCOME.....	3
2.	REQUIREMENTS.....	4
2.1.	User Directory.....	4
2.2.	Plug-in	4
3.	PROCESS FLOW	5
4.	CLASS AND INTERFACES	6
4.1.	UserDir Interface	6
4.2.	UserRecord Class	6
4.3.	Envrnmnt Class	7
4.4.	Logging Class.....	9
4.5.	LogLevel.....	10
5.	METHODS	11
5.1.	init	11
5.2.	close	11
5.3.	lookup.....	11
5.4.	append.....	12
5.5.	remove.....	13
5.6.	remove.....	13
5.7.	getCount()	13
6.	PLUG-IN INSTALLATION	14
7.	PLUG-IN SAMPLE CODE	15

1. Welcome

The Brekeke SIP Server authenticates REGISTER and INVITE requests from SIP user agents. The server queries a built-in user directory database and validates the query through a default authentication plug-in. A user directory is a user database that has a user name and password.

If you want to use another existing user directory service (a user database or authentication server) for authentication on the Brekeke SIP Server, it is possible by creating your own plug-ins.

In this document, we introduce the classes, interfaces, and methods necessary to develop authentication plug-ins.

2. Requirements

2.1. User Directory

At a minimum, the user directory must include a user name and password for each user. The Brekeke SIP Server only queries and does not add or delete user data. Therefore, you must create and delete users using another tool.

2.2. Plug-in

It is recommended that you use Java version 1.4 or later for developing a plug-in because the plug-in will be loaded as a Java class. The class for the plug-in must implement UserDir interface (please refer to 4.1).

The methods that should be implemented in the class are listed below:

init()	-	does initialization
close()	-	does closing
lookup()	-	does a user search
append()	-	adds a user
remove()	-	deletes a user
getCount()	-	gets a total number of users

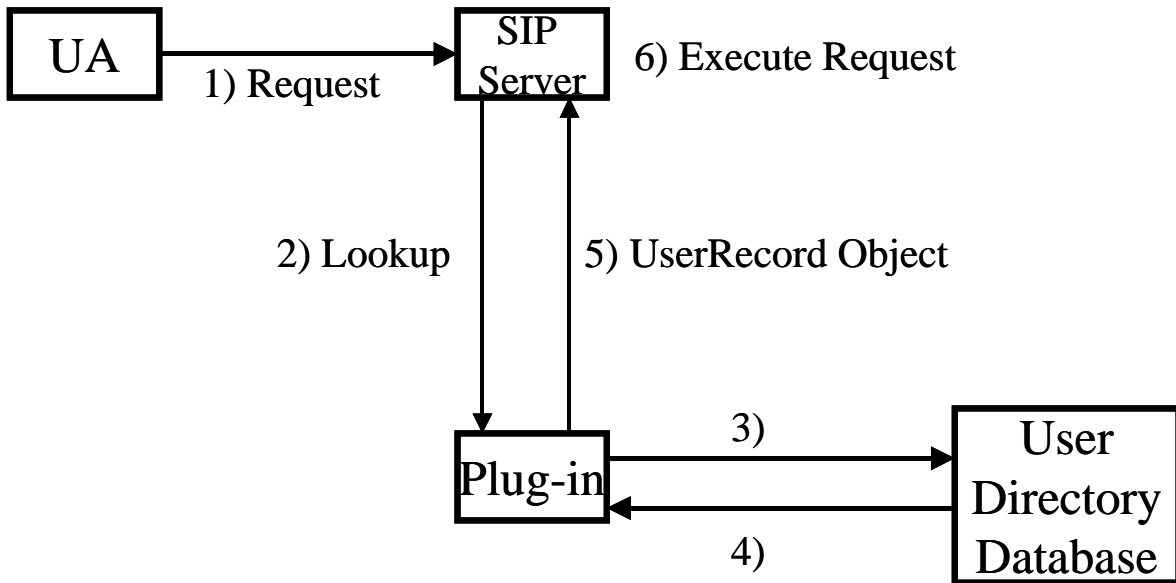
Please refer to Section 5, Methods, for additional details.

For compiling your program, please include

```
<sip_server_install_dir>\webapps\proxy\WEB-INF\lib\ondosip.jar
```

in the classpath.

3. Process Flow



- 1) A user agent sends a SIP request (REGISTER or INVITE) to the Brekeke SIP Server.
- 2) Brekeke SIP Server passes the user name, SIP method name, destination SIP-URI and Proxy-Authorization header field (INVITE) or Authorization header field (REGISTER) which are in the SIP request to the plug-in's lookup() method.
- 3) The plug-in queries the user directory for the record of the user name.
- 4) The user directory returns an appropriate record.
- 5) The plug-in puts the record to a UserRecord object (refer to 4.2, UserRecord Class) and returns the object to Brekeke SIP Server. If no record is found, null is returned.
- 6) Brekeke SIP Server compares the returned result with the authentication information in the SIP request if UserRecord object is returned from the plug-in.

4. Classes and Interfaces

This section explains the important classes and interfaces you need to develop plug-ins.

4.1. UserDir Interface

Package: com.brekeke.net.usrdir

A UserDir interface must be implemented in the plug-in. Please refer to Section 5, Methods, for additional details about the methods of UserDir interface.

4.2. UserRecord Class

Package: com.brekeke.net.usrdir

UserRecord is a class that holds a user's information. This object is returned from a lookup() method when Brekeke SIP Server calls the method.

The UserRecord class does not contain methods. It only contains member variables.

String	username	User name
String	password	Password in plain text
boolean	bAuthorized	A flag on whether authorized or not
long	uid	User ID (Optional)
long	gid	Group ID (Optional)
String	longname	Long user name (Optional)
String	email	e-mail address (Optional)
String	desc	Description of the user (Optional)
long	timeexpire	Expiration (Optional)
long	timemake	Date and time of when this record was created (Optional)
Object	ext	For extension (Optional)

All variables, except username, password, and bAuthorized, are optional. If Brekeke SIP Server needs to calculate an encrypted password, from the plain password, and authorize by comparing the calculated password with a header field (Proxy-Authorization header field, Authorization header field) in the request from user agent, set bAuthorized = false. The default

value of bAuthorized is false. If the authorization is done at plug-in and Brekeke SIP Server does not need to do an authorization, set bAuthorized = true.

4.3. Envrnmt Class

Package: com.brekeke.common

Envrnmt Class: a class that holds environment properties. The Envrnmt object is passed to plug-in as an argument of init() method.

Environment properties must be set in a property file in advance. Restart Brekeke SIP Server after you modify the property file.

The property file is shown below:

Brekeke SIP Server install directory/webapps/proxy/WEB-INF/work/sv/sv.properties

1) getStr

method: public String getStr(String key)

Searches for the String type property with the specified key in the properties. The method returns null if the property is not found.

Parameters: key – the property key.

Returns: The String value with the specified key value.

2) getStr

method: public String getStr(String key, String defStr)

Searches for the String type property with the specified key. The method returns the default value argument if the property is not found.

Parameters:

key – the property key.

defStr – a default value.

Returns: The String value with the specified key value.

3) getInt

method: public int getInt(String key, int defNum)

Searches for the Integer type property with the specified key. The method returns the default value argument if the property is not found.

Parameters:

key – the property key.

defNum – a default value.

Returns: The integer value with the specified key value.

4) getLong

method: public long getLong(String key, long defNum)

Searches for the Long type property with the specified key. The method returns the default value argument if the property is not found.

Parameters:

key – the property key.

defNum – a default value.

Returns: The long value with the specified key value.

4.4. Logging Class

Package: com.brekeke.common

Logging Class: a class to output logs

This object is passed to the plug-in through `init()` method as an argument. A log is output to a system log file.

1) `print`

method: `public void print(String str, LogLevel loglevel, int require)`

If the current log level meets the given required log level, then the string is output.

Parameters:

`str` - The String to be output

`loglevel` - Current log level

`require` - Required log level

2) `println`

method: `public void println(LogLevel loglevel, int require)`

If the current log level meets the given required log level, a new line character (`'\n'`) is output.

Parameters:

`loglevel` - Current log level

`require` - Required log level

3) `println`

method: `public void println(String str, LogLevel loglevel, int require)`

If the current log level meets the given required log level, then the string and a new line character (`'\n'`) is output.

Parameters:

`str` - The String to be output

`loglevel` - Current log level

`require` - Required log level

4.5. LogLevel

Package: com.brekeke.common

LogLevel: a class that contains log level values

This is used for logging. Level is set for console output (standard output) or file output.

Log level values are as follows:

Level	Value	Static field name
No Logging	0	LOG_LEVEL_SILENT
System	1	LOG_LEVEL_SYSTEM
Error	2	LOG_LEVEL_ERROR
Warning	4	LOG_LEVEL_WARNING
Exception	8	LOG_LEVEL_EXCEPTION
Status	16	LOG_LEVEL_STATUS
Detail	32	LOG_LEVEL_DETAIL
Debug	64	LOG_LEVEL_DEBUG
All	255	LOG_LEVEL_ALL

You can specify multiple log levels by logical add (OR). For example, if you want to output only System and Detail logs, the log level will be 34 (2 OR 32).

```
Log level = 2 OR 32
          = 34
```

1) LogLevel

Constructor: `public LogLevel(int levelConsole, int levelFile)`

Creates a new LogLevel object.

Parameters:

levelConsole - Log level for console output.

levelFile - Log level for file output.

5. Methods

This section explains all the methods that a plug-in must implement. These methods are declared in UserDir interface.

5.1. init

method: public void init(Envrnmt env, Logging log) throws Exception

init method is called when Brekeke SIP Server starts. Please include any kind of initialization regarding the plug-in such as connecting a database or setting log levels. If an exception occurs, Brekeke SIP Server aborts starting and exits.

Parameters:

env - Envrnmt object

log - Logging object

5.2. close

method: public void close() throws Exception

close method is called when Brekeke SIP Server is stopped. Please include any kind of endings regarding the plug-in such as disconnection with database.

5.3. lookup

method: public UserRecord lookup(String username, String method, String destination, String authinfo) throws Exception

lookup method is called when Brekeke SIP Server needs to do authentication. User name, SIP method name of the request, destination SIP-URI and Proxy-Authorization header field (INVITE), Authorization header field (REGISTER) are passed as arguments.

User name is supposed to be used as a search key for a user directory. SIP method name and destination URI can be used for the information to make a decision. Put the search result of the user into a UserRecord object as a return value. You must set a username value in UserRecord.

A plain password must be set in the password variable. If the password in the user directory is encrypted, it must be decrypted in this method.

If the authorization is done here, and Brekeke SIP Server does not need to do an authorization, you do not need to set the password however you must set `bAuthorized = true`.

Return value is a `UserRecord` object. If you want to fail the authentication regardless of the search result, return `null`.

If an exception occurs, Brekeke SIP Server fails the authentication and does not execute the request.

Parameters:

- `username` - User name of the user who sent the request
- `method` - SIP method name of the request
- `destination` - Destination SIP-URI of the request
- `authinfo` - Proxy-Authorization header field (INVITE)
Authorization header field (REGISTER)

Returns:

- `UserRecord` object
- `null` for authorization failure

5.4. append

method: `public boolean append(UserRecord record)` throws Exception

Brekeke SIP Server does not call this `append` method. This is supposed to be called from other tools. When this method is called, add the given `UserRecord` object to the user directory.

Parameters: `record` – `UserRecord` object

Returns: `true` if successful
`false` if not successful

5.5. remove

method: public boolean remove(UserRecord record) throws Exception

Brekeke SIP Server does not call this remove method. This is supposed to be called from other tools. When this method is called, remove the given UserRecord object from the user directory.

Parameters:

record – UserRecord object

Returns: true if successful

false if not successful

5.6. remove

method: public boolean remove(String username) throws Exception

Brekeke SIP Server does not call this remove method. This is supposed to be called from other tools. When this method is called, remove the user info of given user name from the user directory.

Parameters: username – user name

Returns: true if successful

false if not successful

5.7. getCount()

method: public int getCount() throws Exception

When this method is called, return the number of users in the user directory.

Returns: the number of users in the user directory

6. Plug-in Installation

Please copy the developed plug-in to some appropriate directory, and add the directory in Java class path (CLASSPATH).

Also, please set the plug-in name to the Brekeke SIP Server's property net.usrdir.plugins. The property file is:

Brekeke SIP Server install directory/webapps/proxy/WEB-INF/work/sv/sv.properties

Examples:

Let us suppose the plugin-in SampleUserDir.class is in the directory /var/plugins/userdir and the package name of the plug-in class is com.domain.proxy.plugins

Add /var/plugins/userdir to the environment variable CLASSPATH.

```
CLASSPATH=/usr/java/jdk/lib:/var/plugins:/var/plugins/userdir
```

Set the plug-in to the environment property net.usrdir.plugins in the sv.properties file.

```
net.usrdir.plugins = com.domain.proxy.plugins.SampleUserDir
```

7. Plug-in Sample Code

Here is a sample of code that explains how to implement init and lookup method.

```
import com.brekeke.common.* ;
import com.brekeke.net.usrdir.* ;

public class SampleUserDir implements UserDir
{
    Envrmnt    env = null ;
    Logging    log = null ;
    LogLevel   loglevel = null ;

    // init
    public void init( Envrmnt env, Logging log ) throws Exception
    {
        this.env = env ;
        this.log = log ;

        // Log level setting
        // Log level is obtained from the environment properties net.userdir.loglevel.console
        // and net.userdir.loglevel.filein this example
        loglevel = new LogLevel( env.getInt( "net.userdir.loglevel.console",
                                           LogLevel.LOG_LEVEL_SYSTEM ),
                               env.getInt( "net.userdir.loglevel.file",
                                           LogLevel.LOG_LEVEL_EXCEPTION ) ) ;

        log.println( "SampleUserDir: start", loglevel, LogLevel.LOG_LEVEL_SYSTEM ) ;

        // Initialization of database connection, etc.

    }
}
```

```
// lookup
public UserRecord lookup( String username, String method, String destination, String
authinfo ) throws Exception
{

    // database inquiry for a user information by username key.

    if ( the user info not found ) {
        return ( null ) ;
    }

    if ( the user doesn't have the authority to execute the method ) {
        return ( null ) ;
    }

    if ( the user doesn't have the authority to access the destination SIP-URI ) {
        return ( null ) ;
    }

    UserRecord record = new UserRecord() ;
    record.username = username ;
    record.password = the plain password obtained from the database

    return ( record ) ;
}
}
```