com.fr.decision.fun.HttpAuthorizeProvider

-
-
-
-
-
-
-
-
-

3 LDAPHTTPHTTPsso10.09.08.0http10.0HTTPHttpAuthorizeProvider

18.09.0http10.0

2http

310.0http

HttpAuthorizeProvider.java

```
package com.fr.decision.fun;

import com.fr.stable.fun.mark.Mutable;

/**
 * http
 */
public interface HttpAuthorizeProvider extends Mutable {

    String MARK_STRING = "HttpAuthorizeProvider";

    int CURRENT_LEVEL = 1;

    /**
     * RSA
     */
    Scope scope();

    /**
     *
     * @param username
     * @param inputPassword
     * @param savedPassword
     * @param hashPassword hash
     * @return truefalse
     */
    boolean authorize(String username, String inputPassword, String savedPassword, String hashPassword);

    /**
     *
     * @return truefalse
     */
    boolean authorize(String uuid, String returnMessage);

    enum Scope {
        /**
         * httpauthorize
         */
        REPLACE,
        /**
         * http
         */
        CHECK
    }
}
```

HttpPassport.java

```
package com.fr.decision.authorize.impl;

...

public class HttpPassport extends AbstractPassport {

    private static final String PARAMETER_NAME = "data";

    ...

    @Override
    public boolean checkTicket(String username, String inputPassword, String savedPassword, String
hashPassword) {
        Set<HttpAuthorizeProvider> providers = ExtraDecisionClassManager.getInstance().getArray
(HttpAuthorizeProvider.MARK_STRING);
        for (HttpAuthorizeProvider provider : providers) {
            if (HttpAuthorizeProvider.Scope.REPLACE == provider.scope()) {
```

```
                boolean result = provider.authorize(username, inputPassword, savedPassword, hashPassword);
                if (result) {
                    return true;
                }
            }
        }
        String publicKey = getDecryptedPublicKey();
        if (StringUtils.isNotBlank(publicKey)) {
            String message;
            String uuid = UUID.randomUUID().toString();
            try {
                HashMap<String, String> para = new HashMap<String, String>();
                Key key = SecurityToolbox.string2PublicKey(publicKey);
                String data = SecurityToolbox.encrypt(
                        requestText(username, inputPassword, savedPassword, hashPassword, uuid),
                        key
                );
                para.put(PARAMETER_NAME, data);
                message = SecurityToolbox.decrypt(HttpToolbox.get(getUrl(), para), key);
            } catch (Exception e) {
                throw new RuntimeException("Http Request Failed:" + e.getMessage());
            }

            boolean result = checkMessage(uuid, message);
            if (result) {
                return true;
            }
            for (HttpAuthorizeProvider provider : providers) {
                if (provider.scope() == HttpAuthorizeProvider.Scope.CHECK) {
                    result = provider.authorize(uuid, message);
                    if (result) {
                        return true;
                    }
                }
            }
        }
        return false;
    }

    private boolean checkMessage(String uuid, String message) {
        if (StringUtils.isEmpty(message)) {
            return false;
        }
        JSONObject data = new JSONObject(message);
        return data.optBoolean(SUCCESS_MARK) && uuid.equals(data.optString(UUID_FIELD));
    }

    private String requestText(String username, String password, String savedPassword, String hashPassword,
String uuid) throws Exception {
        Map<String, String> map = new HashMap<String, String>();
        map.put("username", username);
        map.put("password", password);
        map.put("savedPassword", savedPassword);
        map.put("hashPassword", hashPassword);
        map.put(UUID_FIELD, uuid);
        ObjectMapper mapper = new ObjectMapper();
        return mapper.writeValueAsString(map);
    }

    ...
}
```

FR	10.0		

BI	5.1		
BI	5.1.2		
BI	5.1.3		

plugin.xml

```
<extra-decision>
    <HttpAuthorizeProvider class="your class name"/>
</extra-decision>
```

HttpPassport PassportcheckTickethttp

authorizeHttpPassportScopeREPLACEauthorize(String username, String inputPassword, String savedPassword, String hashPassword)true,httpPassportProviderHttpAuthorizeProvider#authorizefalsehttpScopeCHECKauthorize(String username, String inputPassword, String savedPassword, String hashPassword)httpauthorize(String uuid, String returnMessage)

demo-http-authorize-provider

open-JSD-7814

demo-auth-http