CAIRIS-SAFAX Integration Manuals

EMANUELE CELORIA

22-02-2018

Contents

1 Developer Manual

The purpose of this manual is to describe the code developed to implement the integration between CAIRIS and SAFAX tools. The project is going to be described package by package and file by file.

1.1 "cairis"module

Folder named "cairis"represents the *CAIRIS module* implemented "ex novo"on SAFAX. This is a web service developed in JAVA and packaged as a WAR file; it represents the core part of the final integration. In the subfolder *src* it's possible to find all the source files (.java) where the code for the implementation of the *CAIRIS module* is present, in particular they are organized in 4 packages:

1. /nl/tue/sec/cairis/db
2. /nl/tue/sec/cairis/engine
3. /nl/tue/sec/cairis/impl
4. /nl/tue/sec/cairis/util
5. /nl/tue/sec/cairis/ws

1.1.1 /nl/tue/sec/cairis/db

In this package 2 files are present:

DBAbstraction.java: In *DBAbstraction* class, methods supporting the connection to the MySql SAFAX database (db) are present, together with the implementation of the upload, select, update and delete statements, which are used when performing the queries. The class has been declared *public* in order to be visible also outside its package, in such a way, for the implemented methods, to be invoked by other external classes. The methods defined inside *DBAbstraction* class are:

- **private static void connect**: It is a private static method, so it can be invoked just within this class and without having to create an instance of the class. It creates a *DBJerseyConfig* class' instance which is used to retrieve all the configuration parameters needed to successfully perform the connection, such as: db host, db port, db name, db username, db password. These parameters are retrieved from the *config.db.properties* file.

  - parameters: no input parameters.
  - returns: no return value (void method).

- **private static boolean disconnect**: Performs the closure of the db connection.
  - parameters: no input parameters
  - returns: returns true if the closure of the db connection has been successfully performed, false otherwhise.

- **public static int insertStatement**: It is a public method, so can be invoked also outside this class. By calling the *connect()* method, it retrieves the configuration parameters needed to open a connection towards the db and then it creates and executes the statement used to insert data inside the db. After that, the disconnect method is invoked and the connection is closed.
  - parameters:
    - String query: which is the query to be performed on the db.
    - List<String>variables: which is a list of all the parameters to insert inside the query, ordered as they appear in the query.
  - returns: returns an integer value greater than 0 if the statement has been executed correctly, 0 otherwise.

- **public static boolean updateStatement**: By calling the *connect()* method, it retrieves the configuration parameters needed to open a connection towards the db and then it creates and executes the statement used to upload data inside the db. After that, the disconnect method is invoked and the connection is closed.
  - parameters:
    - String query: which is the query to be performed on the db.
    - List<String>variables: which is a list of all the parameters to insert inside the query, ordered as they appear in the query.
  - returns: returns true if the statement has been executed correctly, false otherwise.

- **public static boolean deleteStatement**: By calling the *connect()* method, it retrieves the configuration parameters needed to open a connection towards the db and then it creates and executes the statement used to delete data inside the db. After that, the disconnect method is invoked and the connection is closed.
  - parameters:

        - String query: which is the query to be performed on the db.
        - List<String>variables: which is a list of all the parameters to insert inside the query, ordered as they appear in the query.
    - returns: returns true if the statement has been executed correctly, false otherwise.

- **public static ArrayList<String>selectRecords**: By calling the *connect()* method, it retrieves the configuration parameters needed to open a connection towards the db and then it creates and executes the statement used to select multiple tuples inside the db. After that, the disconnect method is invoked and the connection is closed.

    - parameters:
        - String query: which is the query to be performed on the db.
        - List <String>variables: which is a list of all the parameters to insert inside the query, ordered as they appear in the query.
    - returns: returns a list of String values which represent the tuples that have been selected from the db, *null* otherwise.

- **public static String selectRecord**: By calling the *connect()* method, it retrieves the configuration parameters needed to open a connection towards the db and then it creates and executes the statement used to select a single tuple inside the db. After that, the disconnect method is invoked and the connection is closed.

    - parameters:
        - String query: which is the query to be performed on the db.
        - List<String>variables: which is a list of all the parameters to insert inside the query, ordered as they appear in the query.
    - returns: returns a String value which represent the tuple that has been selected from the db, *null* otherwise.

- **public static ArrayList<String> selectColumns**: By calling the *connect()* method, it retrieves the configuration parameters needed to open a connection towards the db and then it creates and executes the statement used to extract, from each selected tuple, the attributes composing it. After that, the disconnect method is invoked and the connection is closed.

    - parameters:
        - String query: which is the query to be performed on the db.
        - List<String>variables: which is a list of all the parameters to insert inside the query, ordered as they appear in the query.

  - returns: returns a list of String value which represent the attributes of the tuples that have been selected from the db, *null* otherwise.

DBFns.java : In *DBFns* class, methods that perform queries to the SAFAX database are defined. The class has been defined *public* in order to be visible also outside its package and let the *public* methods defined in the class to be invoked. Furthermore, methods have been defined *static*, so that can be invoked without the need of creating an instance of the class. The methods defined inside *DBFns* class are:

- **public static int errorlog**: Insert the error in SAFAX *ext_errorlog* table.
  - parameters:
    - String transactionID: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
    - String logHead: which represents an explicative header explaining the type of error.
    - String logMsg: which is just an additional message that can be added in order to give more infos about the error.
    - int level: which is an integer value representing the error log level (4 = Invalid Errors, 3 = Network Errors, 2 = DB Errors, 1 = Parsing Errors, 0 = Every other Error).
    - String component: which represents the SAFAX component experiencing the error.
  - returns: An integer value representing the result of the *insertStatement* method in *DBAbstraction* class. Which means an integer value greater than 0 if the execution of the query was successful, 0 otherwise.

- **public static String getDemoIDfromTransactionID**: Retrieves the *demoid* from the *sfx_transaction* table.
  - parameters:
    - String transactionID: it's the SAFAX *transaction-id* associated to the transaction performed.
  - returns: An String value representing the *demoid* .

- **public static ArrayList<String> getCAIRISCredentials**: Retrieves, from the *sfx_cairis_credentials* table, the CAIRIS credentials associated to the specified *demoid*.

  - parameters:
    * String demoID: it's the identifier of a demo.
  - returns: A list of String values representing, respectively, the ccid (identifier of the tuple), the CAIRIS username and the CAIRIS password; *null* otherwise.

- **public static String getCAIRISDB**: Retrieves, from the *sfx_ cairis_ db* table, the CAIRIS database associated to the specified CAIRIS account.
  - parameters:
    * String ccid: it's the identifier of the tuple representing a CAIRIS account.
  - returns: A String value representing the CAIRIS database associated to the CAIRIS account identified by the *ccid*.

The followings are methods created with the purpose of being used by the *CAIRIS module* to perform some tests retrieving the risk values from the local *sfx_ cairis_ risks* db. They could be useful either in preliminary configuration phases, just to have a simple overview of the risk retrieval mechanism, or if the CAIRIS' APIs are not working (so, actually, in the normal configuration of SAFAX, these methods are never invoked).

- **public static String getHighestRiskValuefromResource**: Retrieves the highest risk value given a resource.
  - parameters:
    * String resource: which represents the asset which we want to retrieve the risk of.
  - returns: A string which is the result of the selectRecord method in DBAbstraction class.
- **public static String getHighestRiskValuefromResourceGivenThreat**: Retrieves the highest risk value given a resource and a threat.
  - parameters:
    * String resource: which represents the asset which we want to retrieve the risk of.
    * String threat: which represents the threat associated to the asset.
  - returns: A string which is the result of the *selectRecord* method in *DBAbstraction* class.

- **public static String getHighestRiskValuefromResourceGivenEnvironment**: Retrieves the highest risk value given a resource and an environment.
  - parameters:
    - String resource: which represents the asset which we want to retrieve the risk of.
    - String environment: which represents the environment within which we want to retrieve the risk.
  - returns: A string which is the result of the *selectRecord* method in *DBAbstraction* class.
- **public static String getHighestRiskValuefromResourceGivenThreatandEnvironment**: Retrieves the highest risk value given a resource, a threat and an environment.
  - parameters:
    - String resource: which represents the asset which we want to retrieve the risk of.
    - String threat: which represents the threat associated to the asset.
    - String environment: which represents the environment within which we want to retrieve the risk.
  - returns: A string which is the result of the *selectRecord* method in *DBAbstraction* class.

1.1.2 /nl/tue/sec/cairis/engine

In this package, just 1 file is present:

CairisEngine.java : The *CairisEngine* class is a public class, where the method, in which resides the core part of the implementation engine of the *CAIRIS module*, is implemented. The class allows the adoption of two authentication methods implemented. Since, as default, the Basic HTTP authentication method is enabled, in order to use the simple authentication method, you should just uncomment the line which follows the comment string “SIMPLE SESSION_ID AUTHENTICATION" and comment the part of the code between the comment strings “HTTP BASIC AUTHENTICATION" and “End HTTP BASIC AUTHENTICATION". The method defined inside *CairisEngine* class is:

- **public static int coreExecute**: It’s the core logic that resides behind the functioning of the *CAIRIS module.* It invokes methods to retrieve CAIRIS credentials and database, to perform authentication on CAIRIS and to retrieve the risk.

  - parameters:
    * String CairisURL: Risk API URL.
    * String transID: it's the SAFAX *transaction-id* representing the transaction performed.
    * String authURL: CAIRIS API to contact in order to obtain the *session-id*.
    * String CAIRISdb: the name of the CAIRIS db to contact.
  - returns: A positive integer value (≥ 0) representing the value of risk, retrieved by CAIRIS, -1 otherwise.

1.1.3 /nl/tue/sec/cairis/impl

In this package 3 files are present:

AuthenticationMethods.java : The *AuthenticationMethods* class is a public class, where methods that support the two kinds of authentication supported by CAIRIS are implemented. The methods defined inside *AuthenticationMethods* class are:

- **public static ClientResponse httpBasicAuth**: Allows to perform http basic authentication, by contacting the proper CAIRIS api, passed as parameter, through POST method, and giving it username and password in order to obtain a valid *session-id*.
  - parameters:
    * String username: username used to authenticate on CAIRIS.
    * String passwd: password used to authenticate on CAIRIS.
    * String authURL: CAIRIS API to contact in order to obtain the *session-id*.
  - returns: An object of type *ClientResponse* which represents the response to the invocation of the *authURL*.
- **public static ClientResponse simpleAuth**: Allows to directly retrieve the risk value from the CAIRIS API, passed as parameter by means of a standard and predefined authentication (*session_id*=test) supported by CAIRIS.
  - parameters:
    * String cairisURL: the complete URL to invoke in order to retrieve the risk from CAIRIS.

  - returns: An object of type *ClientResponse* which represents the response to the invocation of the *cairisURL*.

CAIRISRetrievalMethods.java: The *CAIRISRetrievalMethods* class is a public class implementing the risk's retrieval method, used when HTTP Basic authentication is previously used and a valid *session-id* has been correctly retrieved. The method defined inside *CAIRISRetrievalMethods* class is:

- **public static ClientResponse getRisk**: Allows to retrieve the risk value from the CAIRIS API passed as parameter after that a valid *session-id* have been negotiated. The *session-id* is extracted from the response, received by CAIRIS, parsed and attached to the *CairisURL*.
  - parameters:
    - ClientResponse resp: it's the response, received from CAIRIS, to the HTTP basic authentication request.
    - String CairisURL: the complete URL to invoke in order to retrieve the risk from CAIRIS.
    - String CAIRISdbURL: the CAIRIS database URL API to contact in order to open the desired db.
    - String dbname: is the name of the CAIRIS database to open.
    - String transID: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
  - returns: An object of type *ClientResponse* which represents the response to the invocation of the *CairisURL*.

- **private static ClientResponse openCAIRISdb**: Allows to open the CAIRIS database passed as parameter. The method has been implemented as *private*, in such a way to be visible only within the class.
  - parameters:
    - String sessionCairis: the *session-id* retrieved form CAIRIS and to include in the database open API.
    - String CAIRISdbURL: the CAIRIS database URL API to contact in order to open the desired db.
    - String dbname: is the name of the CAIRIS database to open.
    - String transID: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.

  - returns: An object of type *ClientResponse* which represents the response to the invocation of the *CAIRISdbURL*.

EvaluationMethods.java : The *EvaluationMethods* class is a public class implementing the threshold based risk's evaluation method. If, in the future, a new evaluation method will be implemented, it could be defined in this class. The method defined inside *EvaluationMethods* class is:

- **public static boolean evaluateThreshold**: Allows to compare the risk, passed as parameter, against a threshold, passed as parameter too.
  - parameters:
    - int risk: it's the risk that have been retrieved from CAIRIS.
    - int threshold: it's the threshold against which we compare the risk.
    - transID: it's the SAFAX transaction-id associated to the transaction performed, that allows to call methods for writing logs.
  - returns: True if the risk is below the threshold, false otherwise.

1.1.4 /nl/tue/sec/cairis/util

In this package 4 files are present:

CairisUtil.java: The *CairisUtil* class is a public class where utility functions can be implemented. First of all, the COMPKEY parameter is defined. In particular it is defined as *private*, since there is no reason why it should be accessible from outside this class, *final* because it is a value that will never change and *static* so that can be directly called without creating an instance of the class. The method defined inside *CairisUtil* class is:

- **public static String writeLog**: This method is used to contact the PAP component which is, among other things, responsible of storing and showing the logs relative to the evaluation mechanism of a XACML request, showing them in the Account Activity section of SAFAX application. The data are passed in the URL by means of a *Form* instance, the parameters passed in the *Form* object are: *clientcode* (a unique string that identifies the component which is writing the logs), *transactionid* (the SAFAX transaction-id), *level* (the log level), *message* (the message to write in the logs), *serviceid* (the denomination of the component: *nl:tue:sec:cairis*).

  - parameters:
    * String transactionid: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
    * int level: it's the log level (3 = Network Messaging, 2 = Internal messages to the component, 1 = Interface level, 0 = Every other, -1 = Info, -2 = Warning, -3 = Error, -4 = Super Critical Errors).
    * String message: it's the message to write in the log.
  - returns: A string representing the entity response if the PAP URL has been correctly contacted (code status = 200), an empty string otherwise.

DataUtil.java: The *DataUtil* class is a public class where JSON parsing methods are defined. The methods defined inside *DataUtil* class are:

- **public static JSONObject MapToJSON**: This method converts a *LinkedHashMap* into a *JSONObject*.
  - parameters:
    * LinkedHashMap<String, String> lmap: the map to be converted.
  - returns: a *JSONObject* object.
- **public static JSONArray MapToJSON**: This method converts an array of *LinkedHashMap* into a JSON Object.
  - parameters:
    * ArrayList<LinkedHashMap<String, String>> lmap: the array of maps to be converted.
  - returns: a *JSONArray* array of *JSONObject* objects.
- **public static ArrayList<LinkedHashMap<String,String>> JSONArrayToMap**: This method converts an array of JSON objects into an array of maps.
  - parameters:
    * JSONArray jArray: the JSON array to be converted.
  - returns: an *ArrayList* of *LinkedHashMap* maps.
- **public static LinkedHashMap<String,String> JSONToMap**: This method converts a JSON object into a map.

  - parameters:
    * JSONObject json: the JSON object to be converted.
  - returns: a *LinkedHashMap* map.

- **public static JSONArray StringTOJSONArray**: This method converts a string into a JSON array.
  - parameters:
    * String jsonString: the string to be converted.
  - returns: a *JSONArray* array.

- **public static ArrayList<String> convertToList**: This method converts a sequence of strings into an array of strings.
  - parameters:
    * String...args: the sequence of strings to be converted.
  - returns: an *ArrayList* array of String objects.

- **public static JSONObject getJSONFromInt**: This method converts an integer into a JSON object.
  - parameters:
    * int response: the integer value to be converted.
  - returns: a *JSONObject* object.

  -

- **public static JSONObject getJSONFromDouble**: This method converts a double into a JSON object.
  - parameters:
    * double response: the double value to be converted.
  - returns: a *JSONObject* object.

- **public static JSONObject getJSONFromString**: This method converts a string into a JSON object.
  - parameters:
    * String response: the string to be converted.
  - returns: a *JSONObject* object.

- **public static JSONObject getJSONFromBool**: This method converts a boolean value into a *JSONObject*.
  - parameters:

    - Boolean response: the boolean value to be converted.
  - returns: a *JSONObject* object.

- **public static Response buildResponse**: This method builds a response to an invocation of an API, with a *JSONString* object.
  - parameters:
    - JSONObject json: the *JSONObject* to pass in the response.
  - returns: a Response object.

- **public static Response buildResponse**: This method builds a response to an invocation of an API, with a *JSONString* object.
  - parameters:
    - JSONArray json: the JSON array to pass in the response.
  - returns: a Response object.

DBJerseyConfig.java: The *DBJerseyConfig* class is a public class used to retrieve the configuration parameters in order to successfully connect to SAFAX MySql db. The methods defined inside *DBJerseyConfig* class are:

- **public String getDBHost**: Allows to get the db host.
  - parameters: no input parameters.
  - returns: a string indicating the db host.

- **public String getDBPort**: Allows to get the db port.
  - parameters: no input parameters.
  - returns: a string indicating the db port.

- **public String getDB**: Allows to get the db name.
  - parameters: no input parameters.
  - returns: a string indicating the db name.

- **public String getDBUser**: Allows to get the db username.
  - parameters: no input parameters.
  - returns: a string indicating the db username.

- **public String getDBPassword**: Allows to get the db password.
  - parameters: no input parameters.
  - returns: a string indicating the db password.

- **public String getPropValues**: Allows to read and get the configuration parameters from the file where they are specified (*config.db.properties* file).
    - parameters: no input parameters.
    - returns: a string value that list all the parameters read from the file.

LogUtil.java: The *LogUtil* class is a public class which implements log methods. The methods defined inside *LogUtil* class are:

- **public static void errorlog**: it just invokes errorlog method in *DBFns* class.
    - parameters:
        - String transactionid: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
        - String header: represents an explicative header explaining the type of error.
        - String message: just an additional message that can be added in order to give more infos about the error.
        - int level: an integer value representing the error log level (4 = Invalid Errors, 3 = Network Errors, 2 = DB Errors, 1 = Parsing Errors, 0 = Every other Error).
    - returns: no return value (void method).
- **public static void writeLog**: it just invokes *writeLog* method in *CairisUtil* class.
    - parameters:
        - String transactionid: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
        - String message: just an additional message that can be added in order to give more infos about the error.
        - int level: an integer value representing the error log level (4 = Invalid Errors, 3 = Network Errors, 2 = DB Errors, 1 = Parsing Errors, 0 = Every other Error).
    - returns: no return value (void method).

1.1.5 /nl/tue/sec/cairis/ws

In this package 1 file is present:

CairisService.java: The *CairisService* class is a public class where CAIRIS UDFs are specified. The core part of the implementation of these UDFs has been defined in the *CairisEngine* file, within *nl.tue.sec.cairis.engine* package. Indeed, each one of the UDFs, after a preliminary self-configuration step, invokes the *coreExecute* method, which follows a standard procedure to retrieve the final risk value from CAIRIS and thus can exploit a unified code. In this class, 3 *private final static* parameters are defined: *CAIRISHOME*, which represents the URL of the CAIRIS demo, *authURL*, which represents the URL of the API to contact in order to obtain a session in CAIRIS, and *CAIRISdb*, which represents the URL to contact in order to manage the relation with a CAIRIS database. This parameters are defined *private* because they are going to be used only within this class and *final*, since they will never change their value. The UDFs implemented in *CairisService* class are:

- **public Response asset**
  - urn:bu:udf:cairis:risk:level:asset : It's the UDF that should be able to retrieve the highest risk for an asset.
  - Path: api/risk_level/asset/{asset_name}/{threshold}/{transID}
  - Method: GET
  - Parameters:
    - asset_name: it's the name of the resource which we want to retrieve the risk of.
    - threshold: it's the threshold against which evaluate the resource's risk.
    - transID: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
  - Returns: a JSON Response encapsulating a boolean value, true if risk is below threshold, false otherwise.
- **public Response asset_threat**
  - urn:bu:udf:cairis:risk:level:asset:threat:type : It's the udf that should be able to retrieve the highest risk for an asset, given a threat type.
  - Path: api/risk_level/asset/threat_type/{asset_name}/{threat_name}/{threshold}/{transID}
  - Method: GET

  - Parameters:
    - asset_name: it's the name of the resource which we want to retrieve the risk of.
    - threat_name: it's the name of the threat associated to the asset.
    - threshold: it's the threshold against which evaluate the resource's risk.
    - transID: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
  - Returns: a JSON Response encapsulating a boolean value, true if risk is below threshold, false otherwise.

- **public Response asset_environment**
  - urn:bu:udf:cairis:risk:level:asset:environment : It's the udf that should be able to retrieve the highest risk for an asset, in a specific environment.
  - Path: api/risk_level/asset/environment/{asset_name}/{environment}/{threshold}/{transID}
  - Method: GET
  - Parameters:
    - asset_name: it's the name of the resource which we want to retrieve the risk of.
    - environment: it's the name of the environment in which we want to retrieve the resource's risk.
    - threshold: it's the threshold against which evaluate the resource's risk.
    - transID: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
  - Returns: a JSON Response encapsulating a boolean value, true if risk is below threshold, false otherwise.

- **public Response asset_threat_environment**
  - urn:bu:udf:cairis:risk:level:asset:threat:type:environment : It's the udf that should be able to retrieve the highest risk for an asset, in a specific environment, given a threat type.
  - Path: api/risk_level/asset/threat_type/environment/{asset_name}/{threat_name}/{environment}/{threshold}/{transID}
  - Method: GET
  - Parameters:

    * asset_name: it's the name of the resource which we want to retrieve the risk of.
    * threat_name: it's the name of the threat associated to the asset.
    * environment: it's the name of the environment in which we want to retrieve the resource's risk.
    * threshold: it's the threshold against which evaluate the resource's risk.
    * transID: it's the SAFAX *transaction-id* associated to the transaction performed, that allows to call methods for writing logs.
  - Returns: a JSON Response encapsulating a boolean value, true if risk is below threshold, false otherwise.

1.1.6 The other "cairis"subfolders

In subfolder *resources* it's possible to find the *config.db.properties* file, which provides the configuration's informations needed to successfully interact with the SAFAX database. In subfolder *build* it's possible to find all the compiled files (.class) of the project. Finally, in *WebContent* subfolder it's possible to find the libraries included in the project and the *web.xml* file, which is the standard deployment descriptor for the Web application that the Web service is part of. It declares filters and servlets used by the service.

1.2 "sfx"module

Folder named "sfx"represents the Web Service providing the graphical part of the project. Here HTML and Javascript files, needed to support the functioning of the application, are contained. This Web Service was already present in the original project, so only the changes in the files that have been updated, to support the goal of this thesis work, will be reported here:

1.2.1 WebContent/main.html

In this file, the HTML code has been updated in such a way to include, in the GUI, the *Risk* tab, where the user can insert and update the CAIRIS Settings. By searching for the word "CAIRIS ", inside the file, it is possible to see the updates that have been made. Mostly, they consist in a form where to insert username, password and database informations.

1.2.2 WebContent/js/sfxmain.js

In this file, the javascript function called by the Save Settings button in *main.html* has been implemented. The function sends an Ajax request to the *sfxservice* module, contacting the *sfxservice/demo/edit/cairis/* API and passing as parameters the informations inserted in the form by the user, corresponding to the CAIRIS credentials and database informations. Then, it displays a message showing whether the result of the operations has been successful or not.

1.3 "sfxservice"module

Folder named "sfxservice"represents the Web Service providing the interface between the web component of the project and the engine part, developed in java. This Web Service was already present in the original project, so only the changes in the files that have been updated, to support the goal of this thesis work, will be reported here:

1.3.1 package nl.tue.sec.safax.sfxbe.db

DBFns.java: In *DBFns* class, methods that perform queries to the SAFAX database are defined. The class has been defined *public* in order to be visible also outside its package and let the *public* methods defined in the class to be invoked. Furthermore, methods have been defined *static*, so that can be invoked without the need of creating an instance of the class. The methods newly implemented are:

- **public static String setCairisCredentials:** The method sets into the *sfx_cairis_credentials* table the username and the password for the specific demo; if they already exist, it updates them.
  - parameters:
    - String cairisuname: the CAIRIS username.
    - String cairispwd: the CAIRIS password.
    - String demoid: the identifier of the demo.
  - returns: the identifier of the tuple for the set/updated CAIRIS account, *null* otherwise.

- **public static int setCairisDB:** The method sets into the *sfx_cairis_db* table the database to associate to a CAIRIS account specified in the *sfx_cairis_credentials* table. Whenever a database is added or updated, it becomes the default one.

  - parameters:
    - String dbname: the CAIRIS database.
    - String cairisid: the identifier of the tuple of the CAIRIS account to which associate the database.
  - returns: an integer value representing the identifier of the tuple for the set/updated CAIRIS database, 0 if something in the process goes wrong.

1.3.2 package nl.tue.sec.safax.sfxbe.impl

DemoHandlerImpl.java: The *DemoHandlerImpl* class is a public class where the implementation engine of methods in *DemoHandler.java* file is implemented. The method newly implemented is:

- **public String editCairis:** The method allows to invoke the database functions supporting the setting or the update of the CAIRIS credentials and database informations.
  - parameters:
    - String demoid: the demo identifier.
    - String uname: the SAFAX username.
    - String cairisusr: the CAIRIS username.
    - String cairispwd: the CAIRIS password.
    - String cairisdb: the CAIRIS database.
    - String projectid: the SAFAX project identifier.
    - String sessionid: the SAFAX user session identifier.
  - returns: a string value representing the result of the operation.

1.3.3 package nl.tue.sec.safax.sfxbe.ws

DemoHandler.java: The *DemoHandler* class is a public class where the SAFAX API called by the javascript functions through Ajax request, are defined. The API newly implemented is:

- **public Response editCairis**
  - Path: /edit/cairis/{demoid}/{projectid}
  - Method: POST
  - URL parameters:
    - String demoid: the demo identifier.

    * String projectid: the SAFAX project identifier.
  - Path parameters:
    * String cairisusr: the CAIRIS username.
    * String cairispwd: the CAIRIS password.
    * String cairisdb: the CAIRIS database.
  - Cookie parameters:
    * String sessionid: the SAFAX user session identifier.
  - returns: a JSON Response encapsulating a string value representing the result of the operation, carried on by the *editCairis* method in *DemoHandlerImpl* class.

1.4 db_risk_tables folder

In the folder “db_risk_tables”are present the scripts to create the SAFAX database tables needed in the project.

- **safax_sfx_cairis_risks.sql**

  The script creates a SAFAX risk table, named sfx_cairis_risks, that simulates the risks retrieved by CAIRIS and that can be used to test the *CAIRIS module* if, for example, the CAIRIS service is not available. In order to achieve this, a table with these columns is needed:

  - resourceid (PrimaryKey, not null, Datatype: INT)
  - resourcename (not null, Datatype: VARCHAR)
  - threatname (Default: null, Datatype: VARCHAR)
  - environmentname (Default: null, Datatype: VARCHAR)
  - riskvalue (not null, Datatype: INT)

  In order to successfully perform the test, you should uncomment the line of codes in *CairisService* class which are between the comment strings “TEST”and “End TEST”. Then, you should comment the part of code where the *coreExecute* method is invoked.

- **safax_sfx_cairis_credentials.sql**

  The script creates a table, named sfx_cairis_credentials, that contains the CAIRIS credentials associated to each demo in the application. In order to achieve this, a table with these columns is needed:

  - ccid (PrimaryKey, not null, unsigned, zero fill, auto increment, Datatype: INT)

  - cairisuname (not null, Datatype: VARCHAR)
  - cairispwd (not null, Datatype: VARCHAR)
  - demoid (not null, unsigned, zero fill, Datatype: INT)

- **safax_sfx_cairis_db.sql**

  The script creates a table, named sfx_cairis_db, that contains the CAIRIS databases (active and not active) associated with each credential. In order to achieve this, a table with these columns is needed:

  - cdid (PrimaryKey, not null, unsigned, zero fill, auto increment, Datatype: INT)
  - dbname (not null, Datatype: VARCHAR)
  - ccid (not null, unsigned, zero fill, Datatype: INT)
  - isactive (Default: 1, Datatype: TINYINT(1))

1.5 example_policies_requests folder

In the folder "example_policies_requests" are present the XACML risk policies and requests that can be used to benefit from this Risk-based authorization mechanism, each one tests a different CAIRIS API:

- **risk_policy_asset.xml**: This is an example of Risk policy, which asks for a *resource-id* and an *action-id*. The two attributes' values are compared to the attributes with same *attribute-id* in the request according to *urn:oasis:names:tc:xacml:1.0:function: string-equal* function, which is just a string comparator function. In this way, the applicability of the policy is verified. In the condition of the rule, instead, *urn:bu:udf:cairis:risk:level:asset* UDF is called. The function takes as parameters the attributes specified inside, which in this case are the *resource-id* attribute specified in the request and an integer value representing the threshold. If the result of the condition in the first rule will be TRUE, then the effect will be PERMIT, if FALSE, then the second rule's effect will be DENY. (See figure 1).

- **risk_request_asset.xml**: This is an example of Risk request, which wants to perform a certain action over a specified resource, so *resource-id* and *action-id* have to be specified. (See figure 2).

- **risk_policy_asset_env.xml**: This is an example of Risk policy, which asks for an *resource-id*, an environment-id and an action-id. The three attributes' values are compared to the attributes with same *attribute-id* in the request according to *urn:oasis:names:tc:xacml:1.0:function:string-equal* function, which is just a string comparator function. In this way, the applicability of the policy is verified. In the condition of the rule, instead, *urn:bu:udf:cairis:risk:level:asset:environment* UDF is called. The function takes as parameters the attributes specified inside, which in this case are the *resource-id* attribute and the *environment-id* attribute specified in the request and an integer value representing the threshold. If the result of the condition in the first rule will be TRUE, then the effect will be PERMIT, if FALSE, then the second rule's effect will be DENY. (See figure 3).

- **risk_request_asset_env.xml**: This is an example of Risk request, which wants to perform a certain action over a specified resource, in a specific context, so *resource-id*, *environment-id* and *action-id* have to be specified. (See figure 4).

- **risk_policy_asset_threat.xml**: This is an example of Risk policy, which asks for an *resource-id*, a *threat-id* and an *action-id*. The three attributes' values are compared to the attributes with same attribute-id in the request according to *urn:oasis:names:tc:xacml:1.0:function:string-equal* function, which is just a string comparator function. In this way, the applicability of the policy is verified. In the condition of the rule, instead, *urn:bu:udf:cairis:risk:level:asset:threat:type* UDF is called. The function takes as parameters the attributes specified inside, which in this case are the *resource-id* attribute and the *threat-id* attribute specified in the request and an integer value representing the threshold. If the result of the condition in the first rule will be TRUE, then the effect will be PERMIT, if FALSE, then the second rule's effect will be DENY. (See figure 5).

- **risk_request_asset_threat.xml**: This is an example of Risk request, which wants to perform a certain action over a specified resource, given a certain threat, so *resource-id*, *threat-id* and *action-id* have to be specified. (See figure 6).

- **risk_policy_asset_threat_env.xml**: This is an example of Risk policy, which asks for an *resource-id*, a *threat-id*, an *environment-id* and an *action-id*. The four attributes' values are compared to the attributes with same *attribute-id* in the request according to *urn:oasis:names:tc:xacml:1.0:function:string-equal* function, which is just a string comparator function. In this way, the applicability of the policy is verified. In the condition of the rule, instead, *urn:bu:udf:cairis:risk:level:asset:threat:type: environment*

UDF is called. The function takes as parameters the attributes specified inside, which in this case are the *resource-id* attribute, the *threat-id* attribute and the *environment-id* attribute specified in the request and an integer value representing the threshold. If the result of the condition in the first rule will be TRUE, then the effect will be PERMIT, if FALSE, then the second rule's effect will be DENY. (See figure 7).

- **risk_request_asset_threat_env.xml**: This is an example of Risk request, which wants to perform a certain action over a specified resource, given a certain threat, in a specific context, so *resource-id*, *threat-id*, *environment-id* and *action-id* have to be specified. (See figure 8).

2 User Manual

The purpose of this manual is to enhance the utility of the SAFAX Installation Guide [1] and the SAFAX User Manual [2], adding the necessary steps in order to install, better understand and make use of the new *CAIRIS module* implemented for this thesis work. This manual has been organized in two sections, the first one is more like an installation guide, which has been thought to help users who are going to install locally SAFAX from scratch, the second one is a user guide, which instead has the purpose of giving to the user the necessary knowledge to appropriately use the tool.

2.1 Installation Guide

All the references to chapters, sections, pages that are made in this Installation Guide refers to the SAFAX Installation Guide [1]. Pay attention that not every single step described in [1] is repeated in this guide, but only the improvements, changes and of course the additional steps necessary in order to make use of SAFAX tool with the integration supporting the interaction with the CAIRIS tool.

2.1.1 Tomcat installation

In *Prerequisites >Tomcat > Windows section* (pag. 7).

In the following steps, it is explained how to download and install Tomcat. Tomcat is needed both for the development and execution of SAFAX, being used in this project as a servlet container for Jersey web services.

1. Create a custom directory (It is suggested to create the custom directory in the workspace of the user account on Windows system).
2. Download the 32-bit or 64-bit zip file from https://tomcat.apache.org/download-70.cgi.
3. Extract the zip file in the custom directory (make sure after the extraction of tomcat zip that no spaces are present in the path, in order to avoid problems).

2.1.2 Safax repository download via SVN

In *Development Environment* > SVN section (pag. 13).

In this section it is explained how to download SAFAX repository and all its components by using SVN service.

A new folder named *cairis* will be now available, which contains the CAIRIS service. It is a component which allows to have a risk-based access control mechanism by contacting CAIRIS application as an external trust service and relying on it for risk parameters retrieval.

2.1.3 MySQL

In *Development Environment* > MySQL section (pag. 19).

In the following steps it is described how to create with MySQL Workbench a new user account, which is the one that the module of SAFAX will use when contacting the database.

Use MySQL Workbench to create a new database and importing SAFAX database schema:

1. Click *Create a new schema in the connected server* button (9).
2. Name the schema: safax.
3. Choose *latin1_swedish_ci* for the Collation (10).
4. click *Open SQL Script* to import the SAFAX initial database. The script to use when importing tables can be found in the SAFAX SVN server.
5. Execute the script to create tables in the SAFAX database.

6. Create a user named vdJCh9F3 and assign the schema safax to this user, set a password for the user (the same password that the module are going to use when contacting the database) (11).
7. Assign to the user the necessary privileges needed to manage the tables (at least: SELECT, INSERT, UPDATE, DELETE, EXECUTE) (12).

2.1.4 Execution Environment

In *Execution Environment* (pag. 20)

In the following steps it is described how to correctly setup the execution environment of SAFAX.

Prerequisites

The following applications are needed to install the SAFAX development environment:

- JDK
- Tomcat
- MySQL
- MongoDB

See Section "Prerequisite" (pag. 5) for the installation of these applications.

Service Deployment Windows and Ubuntu

1. **Some execution environment configuration**:
   (a) (Only for Windows systems) Setup CATALINA_HOME environmental variable.
      i. Right-click on *My Computer* and select *Properties*.
      ii. Click on *Advanced system settings* > *Advanced* > *Environmental Variables*.
      iii. Under *System variables*, click on *New*.
      iv. In the Variable name enter: "CATALINA_HOME".
      v. In the Variable value enter the path of the Tomcat's installation folder.

vi. Under *System Variables*, select variable *Path* and click on *Edit*.

vii. In the Variable value append: “;%CATALINA_HOME%\bin”.

(b) (Only for Ubuntu systems) create a setenv.sh file in the bin folder of tomcat with the following content:

2. **Update Tomcat users**
   Go to *tomcat-installed-folder/conf/tomcat-users.xml* and add the following lines:

   <role rolename=“tomcat”/>
   <role rolename=“manager-gui”/>
   <role rolename=“manager-script”/>
   <role rolename=“manager-jmx”/>
   <role rolename=“manager-status”/>
   <role rolename=“admin-gui”/>
   <role rolename=“admin-script”/>
   <user username=“tomcat”password=“tomcat”roles=“tomcat”/>
   <user username=“both”password=“tomcat”roles=“tomcat,role1”/>
   <user username=“role1”password=“tomcat”roles=“role1”/>
   <user username=“admin”password=“admin”roles=“manager-gui, manager-script, manager-jmx,manager-status, admin-gui, admin-script”/>

3. **Change Tomcat port to 80**
   Go to *tomcat-installed-folder/conf/server.xml* and update the following lines:

   <Connector port=“80”protocol=“HTTP/1.1”
   connectionTimeout=“20000”redirectPort=“8443”/>

4. **Deploy SAFAX web services to Tomcat server by using one of the two options**:

   (a) Copying web application archive files (.war) into Tomcat webapps folder (*tomcat-installed-folder/webapps*). .war files can be found in SAFAX SVN server.

   (b) Using Tomcat’s manager application:

   i. First start Tomcat
   Windows:
   - opening command prompt.
   - typing cd Tomcat-directory-name.
   - typing .\bin\startup.bat (or %CATALINA_HOME%\bin\startup.bat).
   - typing .\bin\shutdown.bat (when later you will shutdown the server).

  - typing sudo .\catalina.bat (to debug while tomcat is running).

  Ubuntu:
  - opening a terminal.
  - typing cd /tomcat-installed-directory.
  - typing sudo ./bin/startup.sh.
  - typing sudo ./bin/shutdown.sh (when later you will shutdown the server).
  - typing sudo ./catalina.sh run (to debug while tomcat is running).

ii. Go to the Manager App (http://localhost/manager/html).

iii. Login by using the username and password mentioned in tomcat-users.xml file (user: admin, passwd: admin).
  - Scroll down to the section “Deploy”.
  - Upload the .war files and click “Deploy”.

Starting MongoDB application

Windows

1. create the following folder: C:/data/db
2. open a terminal and type: cd C:/Program Files/MongoDB/Server/3.2/bin.
3. run mongoDB application by typing on the terminal: mongod.
4. by default, mongoDB server will start at port 27017.

Linux

1. open a terminal.
2. start MongoDB service by typing: sudo service mongod start.
3. check if MySQL is running by checking the contents of the log file at: /var/log/mongodb/mongod.log for a line saying: [initandlisten] waiting for connections on port <port>.

2.2 User Guide

All the references to chapters, sections, pages that are made in this User Guide refers to the SAFAX User Manual [2]. Pay attention that not every single step described in [2] is repeated in this guide, but only the improvements, changes and of course the additional steps necessary in order to make use of SAFAX tool with the integration of the CAIRIS service.

2.2.1 Advanced Functionalities

In *Chapter 3 > Advanced functionalities* (pag. 32)

In this section, all the advanced features that SAFAX offers for policy evaluation are described. The following new sub-chapter (3.8) should be added:

Risk Service

1. Functional Description:
   The risk service to support a Risk-based access control mechanism is provided by the CAIRIS service. According to this kind of access control, the subject is authorized to perform an action over a resource only if the risk related is below a certain threshold. The idea is to implement a dynamic access control mechanism which goes beyond traditional access control systems (mostly role-based) which have quite a static approach when applying policies.

2. Caution and Warning:
   Not applicable

3. Formal Description:
   On the home page of SAFAX, click to create a new project, then create a new Demo. In the XACML tab upload your risk policy and your request paying attention to be coherent with the standard format that supports the different scenarios.

   It is possible to have 4 different kinds of scenarios:

   Scenario 1: We ask to evaluate the risk of accessing a certain resource. For this kind of scenario the request should contain a *resource-id* attribute (which represents the asset we are asking to access in a certain mode) and of course an *action-id* attribute (which represents the mode according to which we want to access the asset, i.e. read, write,..). The policy should include, in the *Condition*, the threshold against which the risk will be compared. The UDF which has to be used in order to evaluate the condition is: *urn:bu:udf:cairis:risk:level:asset*.

   Scenario 2: We ask to evaluate the risk of accessing a certain resource given a threat type.
   For this kind of scenario the request should contain a *resource-id* attribute (which represents the asset we are asking to access in a certain mode), a *threat-type* attribute (which represents the threat associated to the asset) and of course an *action-id* attribute (which represents the mode according to which we want to access the asset). The policy

should include, in the *Condition*, the threshold against which the risk will be compared. The UDF which has to be used in order to evaluate the condition is: *urn:bu:udf:cairis:risk:level:asset:threat:type*.

Scenario 3: We ask to evaluate the risk of accessing a certain resource in a certain context.
For this kind of scenario the request should contain a *resource-id* attribute (which represents the asset we are asking to access in a certain mode), an *environment* attribute (which represents the environment we are interested in, which is associated to the asset) and of course an *action-id* attribute (which represents the mode according to which we want to access the asset). The policy should include, in the *Condition*, the threshold against which the risk will be compared. The UDF which has to be used in order to evaluate the condition is: *urn:bu:udf:cairis:risk:level:asset: environment*.

Scenario 4: We ask to evaluate the risk of accessing a certain resource in a certain context given a threat type.
For this kind of scenario the request should contain a *resource-id* attribute (which represents the asset we are asking to access in a certain mode), a *threat-type* attribute (which represents the threat associated to the asset), an *environment* attribute (which represents the environment we are interested in, which is associated to the asset) and of course an *action-id* attribute (which represents the mode according to which we want to access the asset). The policy, should include, in the *Condition*, the threshold against which the risk will be compared. The UDF which has to be used in order to evaluate the condition is: *urn:bu:udf:cairis:risk:level:asset:threat: type:environment*.

Then go in the Risk tab (see figure 13) and set, for the specific demo, your CAIRIS settings: credentials (Username and Password) and Database. The username and password informations correspond to the account informations that the user is supposed to possess within the CAIRIS service. The database information, instead, represents the name of the database from which the user wants to retrieve the risk scores.
Click on *Save Settings* button to effectively set the informations.
A specific set of test credentials are available, in order to let a user, without an account on CAIRIS, to test the authorization mechanism: Username = test, Password = test, Database = ACME_Water (see figure 14).

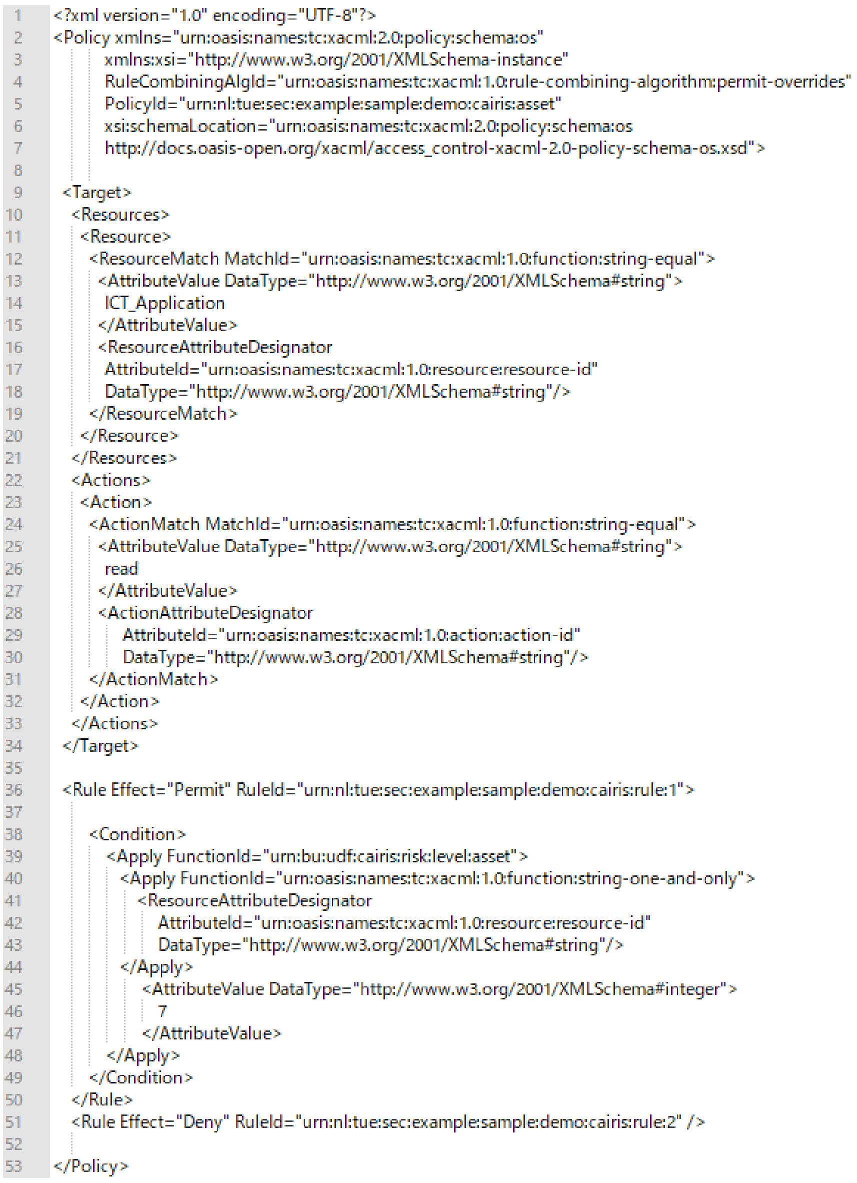
Then, after the uploads, click on run in order to evaluate your request. An example of risk response that is possible to see in the Account Activity page of SAFAX is shown in figure 15

4. Related

References

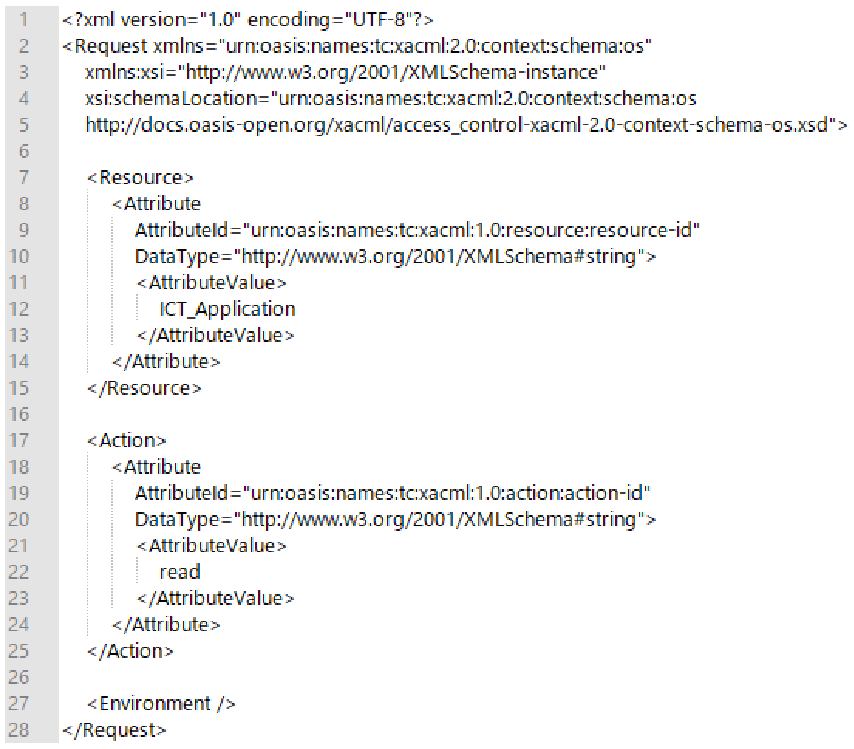
[1] *SAFAX Installation Guide.* URL: http://security1.win.tue.nl/safax/Documentation/safax_installation_guide.pdf.

[2] *SAFAX User Manual.* URL: http://security1.win.tue.nl/safax/Documentation/safax_user_manual.pdf.



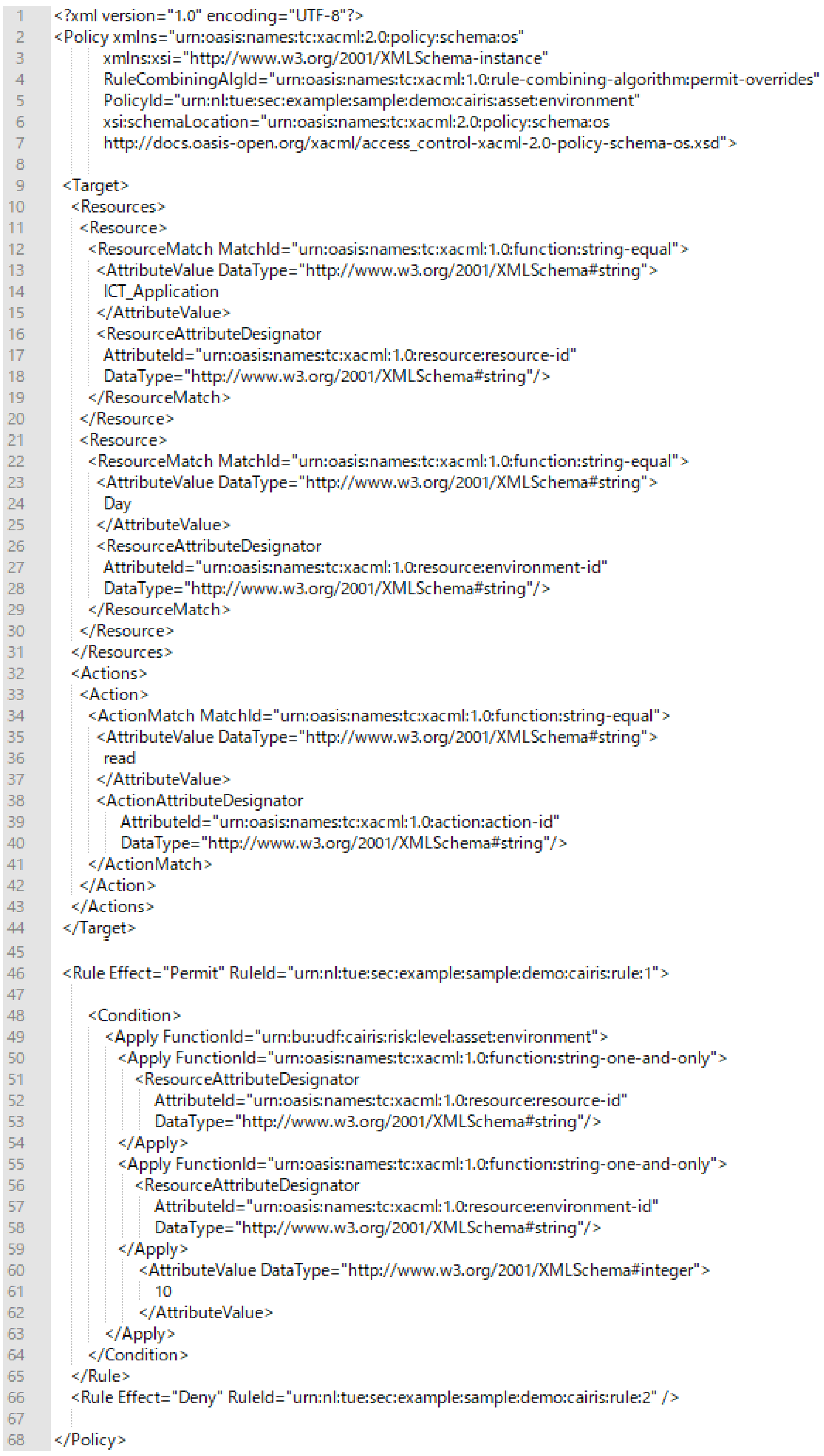
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides"
      PolicyId="urn:nl:tue:sec:example:sample:demo:cairis:asset"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
      http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd">

 <Target>
  <Resources>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      ICT_Application
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
  </Resources>
  <Actions>
   <Action>
    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      read
     </AttributeValue>
     <ActionAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
   </Action>
  </Actions>
 </Target>

 <Rule Effect="Permit" RuleId="urn:nl:tue:sec:example:sample:demo:cairis:rule:1">

    <Condition>
      <Apply FunctionId="urn:bu:udf:cairis:risk:level:asset">
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
         <ResourceAttributeDesignator
           AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
           DataType="http://www.w3.org/2001/XMLSchema#string"/>
       </Apply>
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
           7
         </AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Deny" RuleId="urn:nl:tue:sec:example:sample:demo:cairis:rule:2" />

</Policy>
```

Figure 1: risk_policy_asset.xml.



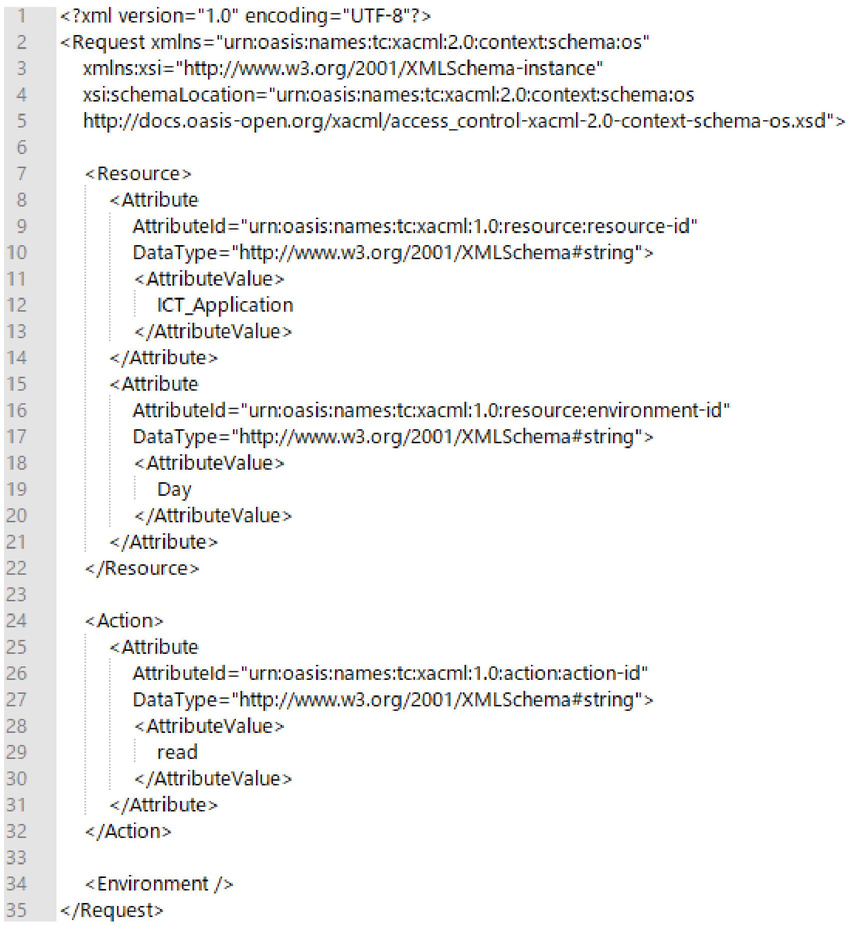
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
  http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">

  <Resource>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        ICT_Application
      </AttributeValue>
    </Attribute>
  </Resource>

  <Action>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        read
      </AttributeValue>
    </Attribute>
  </Action>

  <Environment />
</Request>
```

Figure 2: risk_request_asset.xml.



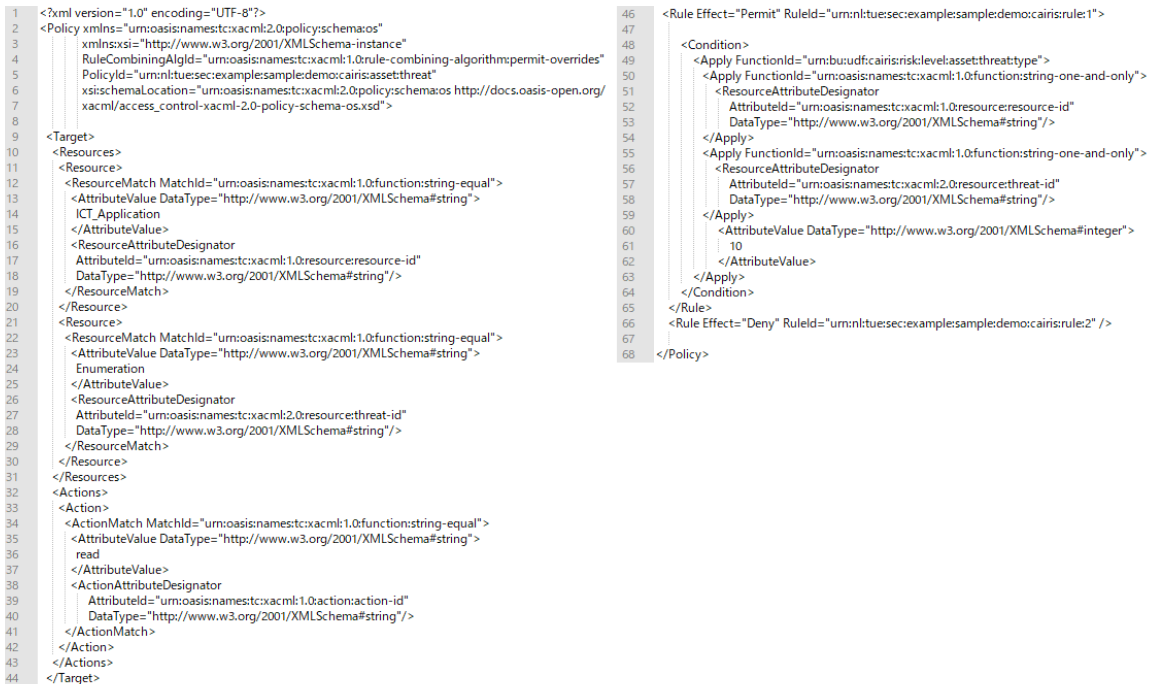
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides"
      PolicyId="urn:nl:tue:sec:example:sample:demo:cairis:asset:environment"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
      http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd">

 <Target>
  <Resources>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      ICT_Application
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      Day
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:environment-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
  </Resources>
  <Actions>
   <Action>
    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      read
     </AttributeValue>
     <ActionAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
   </Action>
  </Actions>
 </Target>

 <Rule Effect="Permit" RuleId="urn:nl:tue:sec:example:sample:demo:cairis:rule:1">

    <Condition>
      <Apply FunctionId="urn:bu:udf:cairis:risk:level:asset:environment">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:environment-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
            10
          </AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Deny" RuleId="urn:nl:tue:sec:example:sample:demo:cairis:rule:2" />

</Policy>
```

Figure 3: risk_policy_asset_env.xml.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
  http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">

    <Resource>
      <Attribute
        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
        DataType="http://www.w3.org/2001/XMLSchema#string">
        <AttributeValue>
          ICT_Application
        </AttributeValue>
      </Attribute>
      <Attribute
        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:environment-id"
        DataType="http://www.w3.org/2001/XMLSchema#string">
        <AttributeValue>
          Day
        </AttributeValue>
      </Attribute>
    </Resource>

    <Action>
      <Attribute
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string">
        <AttributeValue>
          read
        </AttributeValue>
      </Attribute>
    </Action>

    <Environment />
</Request>
```

Figure 4: risk_request_asset_env.xml.



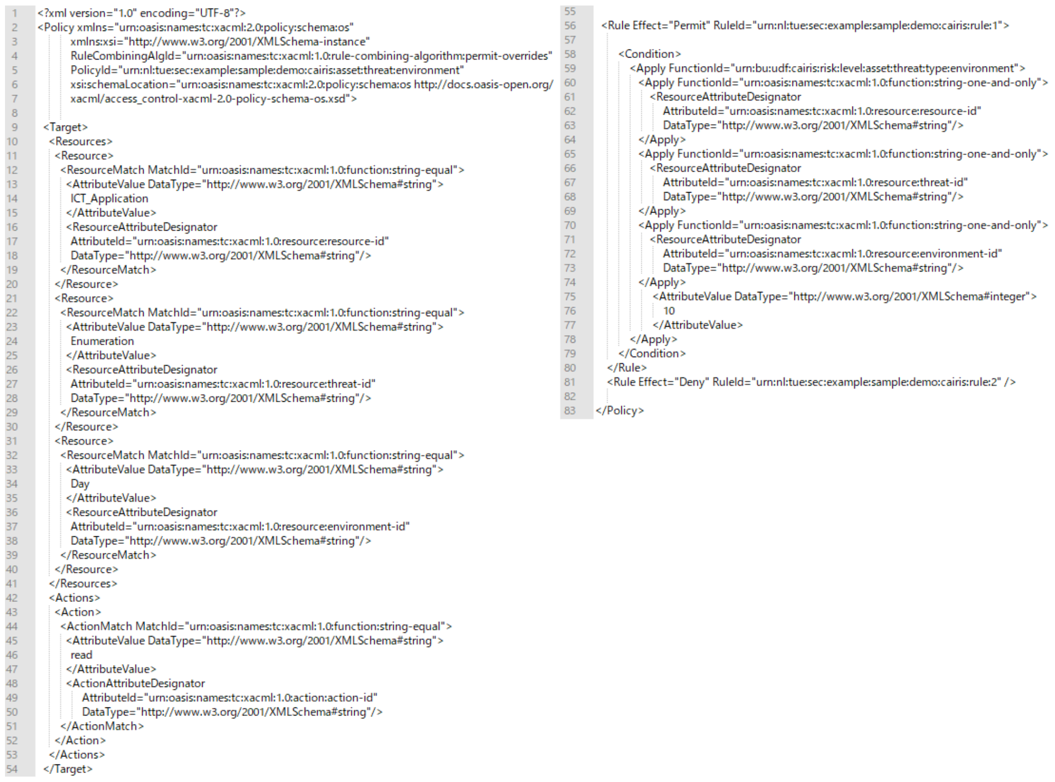
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides"
        PolicyId="urn:nl:tue:sec:example:sample:demo:cairis:asset:threat"
        xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-open.org/
        xacml/access_control-xacml-2.0-policy-schema-os.xsd">

 <Target>
  <Resources>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      ICT_Application
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      Enumeration
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:resource:threat-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
  </Resources>
  <Actions>
   <Action>
    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      read
     </AttributeValue>
     <ActionAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
   </Action>
  </Actions>
 </Target>
```

```xml
 <Rule Effect="Permit" RuleId="urn:nl:tue:sec:example:sample:demo:cairis:rule:1">

    <Condition>
      <Apply FunctionId="urn:bu:udf:cairis:risk:level:asset:threat:type">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <ResourceAttributeDesignator
             AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
             DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <ResourceAttributeDesignator
             AttributeId="urn:oasis:names:tc:xacml:2.0:resource:threat-id"
             DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
             10
          </AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Deny" RuleId="urn:nl:tue:sec:example:sample:demo:cairis:rule:2" />

</Policy>
```

Figure 5: risk_policy_asset_threat.xml.

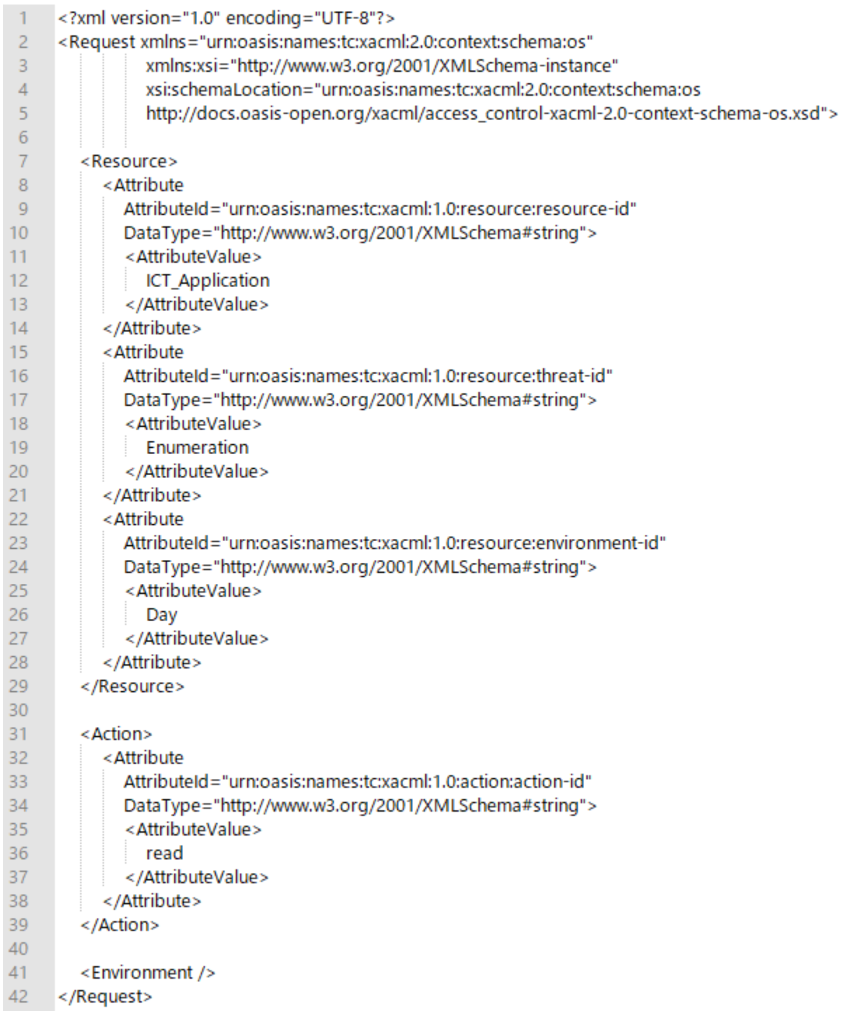
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
  http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">

  <Resource>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        ICT_Application
      </AttributeValue>
    </Attribute>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:2.0:resource:threat-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        Enumeration
      </AttributeValue>
    </Attribute>
  </Resource>

  <Action>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        read
      </AttributeValue>
    </Attribute>
  </Action>

  <Environment />
</Request>
```

Figure 6: risk_request_asset_threat.xml.



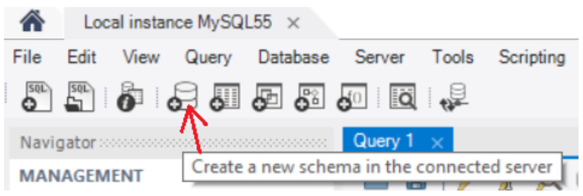
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides"
      PolicyId="urn:nl:tue:sec:example:sample:demo:cairis:asset:threat:environment"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-open.org/
      xacml/access_control-xacml-2.0-policy-schema-os.xsd">

 <Target>
  <Resources>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      ICT_Application
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      Enumeration
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:threat-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      Day
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:environment-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
  </Resources>
  <Actions>
   <Action>
    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      read
     </AttributeValue>
     <ActionAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
   </Action>
  </Actions>
 </Target>

 <Rule Effect="Permit" RuleId="urn:nl:tue:sec:example:sample:demo:cairis:rule:1">

    <Condition>
      <Apply FunctionId="urn:bu:udf:cairis:risk:level:asset:threat:type:environment">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:threat-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:environment-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
            10
          </AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Deny" RuleId="urn:nl:tue:sec:example:sample:demo:cairis:rule:2" />

</Policy>
```

Figure 7: risk_policy_asset_threat_env.xml.



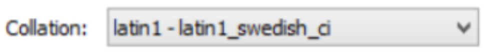
```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
        http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">

  <Resource>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        ICT_Application
      </AttributeValue>
    </Attribute>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:threat-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        Enumeration
      </AttributeValue>
    </Attribute>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:environment-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        Day
      </AttributeValue>
    </Attribute>
  </Resource>

  <Action>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>
        read
      </AttributeValue>
    </Attribute>
  </Action>

  <Environment />
</Request>
```

Figure 8: risk_request_asset_threat_env.xml.



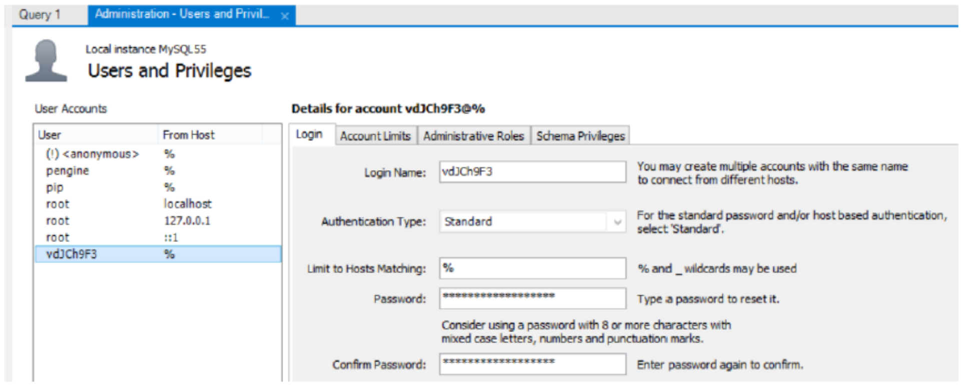
Create a new schema in the connected server

Figure 9: Create a new schema.



Collation: latin1 - latin1_swedish_ci

Figure 10: Collation.



Details for account vdJCh9F3@%

User	From Host
(!) <anonymous>	%
pengine	%
pip	%
root	localhost
root	127.0.0.1
root	::1
vdJCh9F3	%

Login Name: vdJCh9F3 — You may create multiple accounts with the same name to connect from different hosts.

Authentication Type: Standard — For the standard password and/or host based authentication, select 'Standard'.

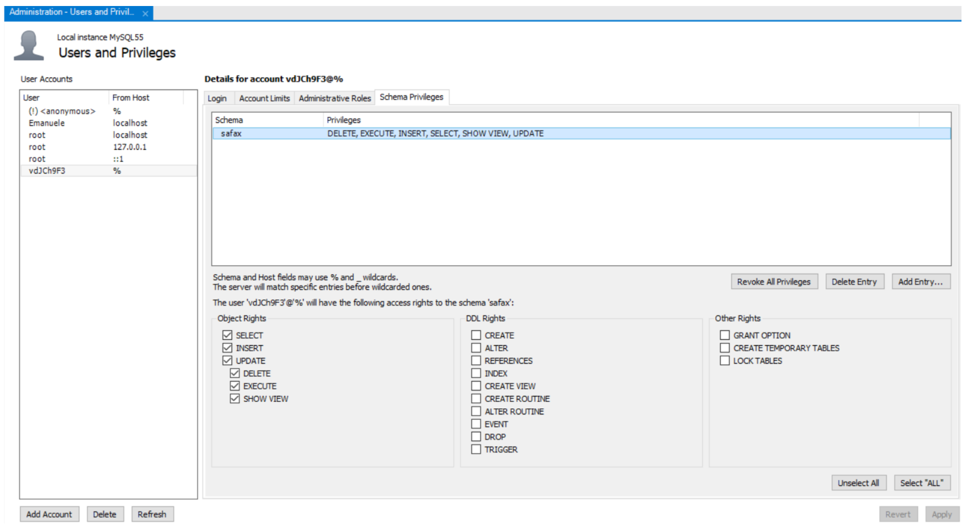
Limit to Hosts Matching: % — % and _ wildcards may be used

Password: ****************** — Type a password to reset it.

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: ****************** — Enter password again to confirm.

Figure 11: User creation.



Details for account vdJCh9F3@%

User	From Host
(!) <anonymous>	%
Emanuele	localhost
root	localhost
root	127.0.0.1
root	::1
vdJCh9F3	%

Schema	Privileges
safax	DELETE, EXECUTE, INSERT, SELECT, SHOW VIEW, UPDATE

Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.

The user 'vdJCh9F3'@'%' will have the following access rights to the schema 'safax':

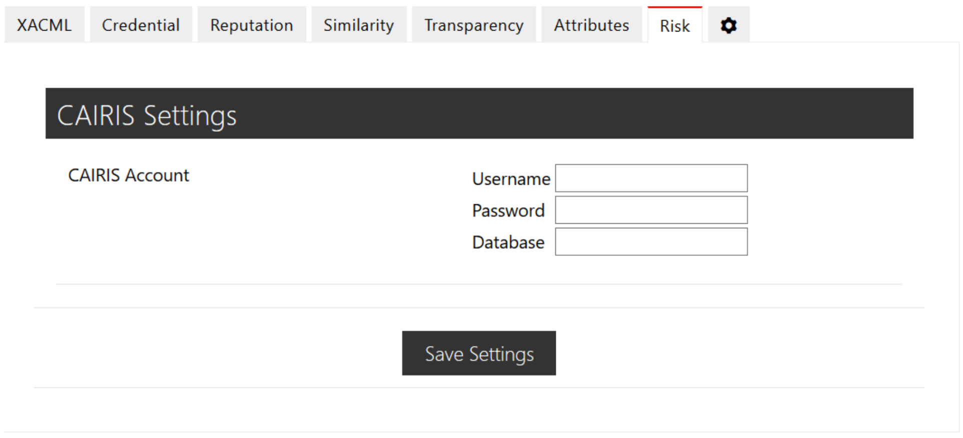
Object Rights
- ✓ SELECT
- ✓ INSERT
- ✓ UPDATE
- ✓ DELETE
- ✓ EXECUTE
- ✓ SHOW VIEW

DDL Rights
- ☐ CREATE
- ☐ ALTER
- ☐ REFERENCES
- ☐ INDEX
- ☐ CREATE VIEW
- ☐ CREATE ROUTINE
- ☐ ALTER ROUTINE
- ☐ EVENT
- ☐ DROP
- ☐ TRIGGER

Other Rights
- ☐ GRANT OPTION
- ☐ CREATE TEMPORARY TABLES
- ☐ LOCK TABLES

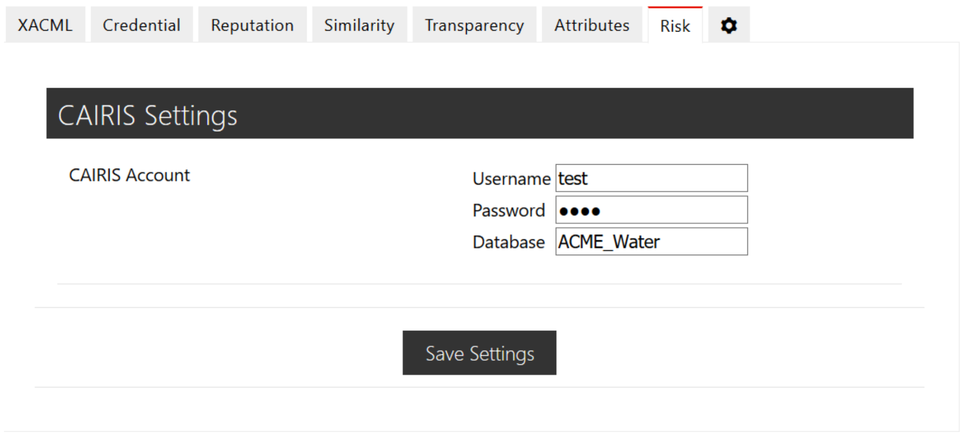
Figure 12: User permission assignment.

```
#!/bin/sh

# Java Heap Size = Place to store objects created by SAFAX web application.
# For a heavy service requested, insufficient Heap size will cause the polular
# java.lang.OutOfMemoryError

# Perm Gen Size = Place to store SAFAX web service loaded class deinition and metadata
# If a large code-based service is loaded, the insufficient Perm Gen size will cause the popular
# Java.Lang.OutOfMemoryError: PermGen
# -XX:MaxPermSize - Set the maximum PermGen Size
# discourages address map swapping by setting Xms and Xmx to the same value?

export CATALINA_OPTS="$CATALINA_OPTS -server -Xms128m"
export CATALINA_OPTS="$CATALINA_OPTS -Xmx1024m"
export CATALINA_OPTS="$CATALINA_OPTS -XX:MaxPermSize=512m"

# Check for application specific parameters at startup

#If [ -r "$CATALINA_BASE/bin/appenv.sh" ]; then
#. "$CATALINA_BASE/bin/appenv.sh"
#fi

echo "Using CATALINA_OPTS:"
for arg in $CATALINA_OPTS
do
echo ">> " $arg
done

echo ""

export JAVA_OPTS="-Djava.library.path=/usr/local/lib"
echo "Using JAVA_OPTS:"
for arg in $JAVA_OPTS

do
echo ">> " $arg
done
echo "________________________________________"
echo ""
```



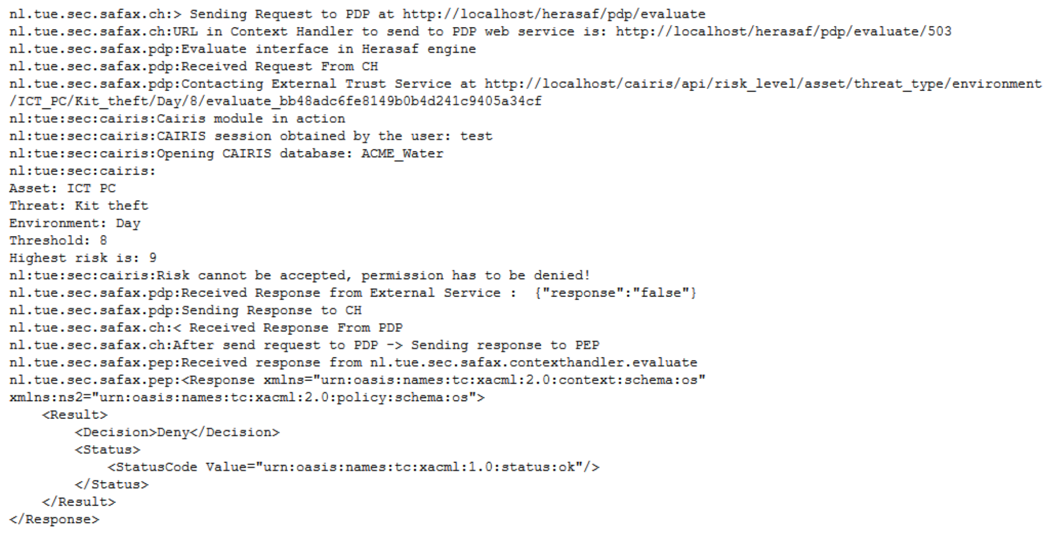
Username
Password
Database

Figure 13: CAIRIS Settings.



Username test
Password ●●●●
Database ACME_Water

Figure 14: CAIRIS Settings with *test* credentials.



```
nl.tue.sec.safax.ch:> Sending Request to PDP at http://localhost/herasaf/pdp/evaluate
nl.tue.sec.safax.ch:URL in Context Handler to send to PDP web service is: http://localhost/herasaf/pdp/evaluate/503
nl.tue.sec.safax.pdp:Evaluate interface in Herasaf engine
nl.tue.sec.safax.pdp:Received Request From CH
nl.tue.sec.safax.pdp:Contacting External Trust Service at http://localhost/cairis/api/risk_level/asset/threat_type/environment
/ICT_PC/Kit_theft/Day/8/evaluate_bb48adc6fe8149b0b4d241c9405a34cf
nl:tue:sec:cairis:Cairis module in action
nl:tue:sec:cairis:CAIRIS session obtained by the user: test
nl:tue:sec:cairis:Opening CAIRIS database: ACME_Water
nl:tue:sec:cairis:
Asset: ICT PC
Threat: Kit theft
Environment: Day
Threshold: 8
Highest risk is: 9
nl:tue:sec:cairis:Risk cannot be accepted, permission has to be denied!
nl.tue.sec.safax.pdp:Received Response from External Service :  {"response":"false"}
nl.tue.sec.safax.pdp:Sending Response to CH
nl.tue.sec.safax.ch:< Received Response From PDP
nl.tue.sec.safax.ch:After send request to PDP -> Sending response to PEP
nl.tue.sec.safax.pep:Received response from nl.tue.sec.safax.contexthandler.evaluate
nl.tue.sec.safax.pep:<Response xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:ns2="urn:oasis:names:tc:xacml:2.0:policy:schema:os">
    <Result>
        <Decision>Deny</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
```

Figure 15: Example of a risk response.